

# **phpDocumentor Manual**



# Contents

<a href="#">phpDocumentor Guide to Creating Fantastic Documentation</a>	1
<a href="#">  phpDocumentor Quickstart</a>	3
<a href="#">  Source code for sample1.php</a>	10
<a href="#">  Source code for sample2.php</a>	12
<a href="#">  Source code for sample3.php</a>	15
<a href="#">  phpDocumentor Tutorial</a>	16
<a href="#">  Documentable PHP Elements</a>	36
<a href="#">    phpDocumentor Tutorials</a>	41
<a href="#">    phpDocumentor Manual</a>	47
<a href="#">    phpDocumentor tags</a>	49
<a href="#">      @abstract</a>	51
<a href="#">      @access</a>	52
<a href="#">      @author</a>	54
<a href="#">      @category</a>	55
<a href="#">      @copyright</a>	56
<a href="#">      @deprecated</a>	57
<a href="#">      @example</a>	58
<a href="#">      @final</a>	60
<a href="#">      @filesource</a>	61
<a href="#">      @global</a>	62
<a href="#">      @ignore</a>	65
<a href="#">      @internal</a>	66
<a href="#">      @license</a>	67
<a href="#">      @link</a>	68
<a href="#">      @method</a>	69
<a href="#">      @name</a>	70
<a href="#">      @package</a>	71
<a href="#">      @param</a>	73
<a href="#">      @property</a>	76
<a href="#">      @return</a>	78
<a href="#">      @see</a>	80
<a href="#">      @since</a>	82
<a href="#">      @static</a>	83
<a href="#">      @staticvar</a>	85
<a href="#">      @subpackage</a>	86
<a href="#">      @todo</a>	88
<a href="#">      @tutorial</a>	89
<a href="#">      @uses</a>	91
<a href="#">      @var</a>	94
<a href="#">      @version</a>	95
<a href="#">    phpDocumentor Inline tags</a>	96
<a href="#">      inline {@example}</a>	97
<a href="#">      inline {@id}</a>	99

<a href="#">inline {@internal}</a>	101
<a href="#">inline {@inheritdoc}</a>	102
<a href="#">inline {@link}</a>	104
<a href="#">inline {@source}</a>	106
<a href="#">inline {@toc}</a>	109
<a href="#">inline {@tutorial}</a>	111
<a href="#"><b>Writing a New Converter</b></a>	113
<a href="#">Converter Default Template Variables</a>	117
<a href="#">Writing a Converter, Methods</a>	122
<a href="#"><b>Converter Manual</b></a>	128
<a href="#"><b>Using the PDFParser XML templating language</b></a>	131
<a href="#">Writing templates for the PDFdefault Converter</a>	133
<a href="#">Writing templates for the CHMdefault Converter</a>	134
<a href="#">Writing templates for the XMLDocBook Converter</a>	135
<a href="#">Writing templates for the HTMLSmarty Converter</a>	136
<a href="#">Writing templates for the HTMLframes Converter</a>	137
<a href="#"><b>Package Converters Procedural Elements</b></a>	143
<a href="#">Converter.inc</a>	143
<a href="#">Function adv_htmlentities</a>	144
<a href="#"><b>Package Converters Classes</b></a>	145
<a href="#">Class Converter</a>	145
<a href="#">Var \$all_packages</a>	145
<a href="#">Var \$class</a>	146
<a href="#">Var \$classes</a>	146
<a href="#">Var \$class_contents</a>	146
<a href="#">Var \$class_data</a>	146
<a href="#">Var \$class_elements</a>	147
<a href="#">Var \$curfile</a>	147
<a href="#">Var \$curpage</a>	147
<a href="#">Var \$define_elements</a>	147
<a href="#">Var \$elements</a>	147
<a href="#">Var \$function_elements</a>	148
<a href="#">Var \$global_elements</a>	148
<a href="#">Var \$highlightingSource</a>	148
<a href="#">Var \$leftindex</a>	148
<a href="#">Var \$outputformat</a>	149
<a href="#">Var \$package</a>	149
<a href="#">Var \$packagecategories</a>	149
<a href="#">Var \$package_elements</a>	150
<a href="#">Var \$package_output</a>	150
<a href="#">Var \$package_parents</a>	150
<a href="#">Var \$page</a>	151
<a href="#">Var \$page_contents</a>	151
<a href="#">Var \$page_data</a>	151
<a href="#">Var \$page_elements</a>	151
<a href="#">Var \$parseprivate</a>	152

<a href="#">Var \$path</a>	152
<a href="#">Var \$pkg_elements</a>	152
<a href="#">Var \$processSpecialRoots</a>	152
<a href="#">Var \$quietmode</a>	152
<a href="#">Var \$smarty_dir</a>	153
<a href="#">Var \$sort_absolutely_everything</a>	153
<a href="#">Var \$sort_page_contents_by_type</a>	153
<a href="#">Var \$sourcePaths</a>	153
<a href="#">Var \$subpackage</a>	153
<a href="#">Var \$targetDir</a>	154
<a href="#">Var \$templateDir</a>	154
<a href="#">Var \$templateName</a>	154
<a href="#">Var \$template_options</a>	154
<a href="#">Var \$title</a>	154
<a href="#">Var \$todoList</a>	155
<a href="#">Constructor Converter</a>	155
<a href="#">Method AttrToString</a>	155
<a href="#">Method Bolden</a>	156
<a href="#">Method Br</a>	156
<a href="#">Method checkState</a>	156
<a href="#">Method cleanup</a>	157
<a href="#">Method Convert</a>	157
<a href="#">Method convertClass</a>	157
<a href="#">Method convertConst</a>	158
<a href="#">Method convertDefine</a>	158
<a href="#">Method ConvertErrorLog</a>	158
<a href="#">Method convertFunction</a>	159
<a href="#">Method convertGlobal</a>	159
<a href="#">Method convertInclude</a>	160
<a href="#">Method convertMethod</a>	160
<a href="#">Method convertPage</a>	160
<a href="#">Method ConvertTitle</a>	161
<a href="#">Method ConvertTodoList</a>	161
<a href="#">Method convertTutorial</a>	161
<a href="#">Method convertVar</a>	162
<a href="#">Method Convert_RIC</a>	162
<a href="#">Method copyFile</a>	162
<a href="#">Method createParentDir</a>	163
<a href="#">Method EncloseList</a>	163
<a href="#">Method EncloseParagraph</a>	163
<a href="#">Method endClass</a>	163
<a href="#">Method endPage</a>	164
<a href="#">Method exampleProgramExample</a>	164
<a href="#">Method flushHighlightCache</a>	164
<a href="#">Method formatIndex</a>	165
<a href="#">Method formatLeftIndex</a>	165
<a href="#">Method formatPkgIndex</a>	165
<a href="#">Method formatTutorialTOC</a>	165
<a href="#">Method generateChildClassList</a>	166

<a href="#">Method generateFormattedClassTree</a>	166
<a href="#">Method getClassesOnPage</a>	167
<a href="#">Method getClassLink</a>	167
<a href="#">Method getConstLink</a>	167
<a href="#">Method getConverterDir</a>	168
<a href="#">Method getCurrentPageLink</a>	168
<a href="#">Method getCurrentPageURL</a>	168
<a href="#">Method getDefineLink</a>	168
<a href="#">Method getFileSourceName</a>	169
<a href="#">Method getFileSourcePath</a>	169
<a href="#">Method getFormattedConflicts</a>	169
<a href="#">Method getFormattedDescMethods</a>	169
<a href="#">Method getFormattedDescVars</a>	170
<a href="#">Method getFormattedImplements</a>	170
<a href="#">Method getFormattedInheritedConsts</a>	170
<a href="#">Method getFormattedInheritedMethods</a>	171
<a href="#">Method getFormattedInheritedVars</a>	171
<a href="#">Method getFormattedMethodImplements</a>	172
<a href="#">Method getFormattedOverrides</a>	172
<a href="#">Method getFunctionLink</a>	172
<a href="#">Method getGlobalLink</a>	173
<a href="#">Method getGlobalValue</a>	173
<a href="#">Method getHighlightState</a>	173
<a href="#">Method getId</a>	173
<a href="#">Method getIncludeValue</a>	174
<a href="#">Method getLink</a>	174
<a href="#">Method getMethodLink</a>	175
<a href="#">Method getPageLink</a>	176
<a href="#">Method getSortedClassTreeFromClass</a>	176
<a href="#">Method getSourceLink</a>	178
<a href="#">Method getState</a>	178
<a href="#">Method getTutorialId</a>	178
<a href="#">Method getTutorialLink</a>	179
<a href="#">Method getTutorialTree</a>	180
<a href="#">Method getVarLink</a>	180
<a href="#">Method hasSourceCode</a>	181
<a href="#">Method hasTutorial</a>	181
<a href="#">Method highlightDocBlockSource</a>	181
<a href="#">Method highlightSource</a>	181
<a href="#">Method highlightTutorialSource</a>	182
<a href="#">Method Italicize</a>	182
<a href="#">Method Kbdize</a>	182
<a href="#">Method ListItem</a>	183
<a href="#">Method newSmarty</a>	183
<a href="#">Method Output</a>	183
<a href="#">Method postProcess</a>	184
<a href="#">Method prepareDocBlock</a>	184
<a href="#">Method PreserveWhiteSpace</a>	184
<a href="#">Method ProgramExample</a>	185

<a href="#">Method returnLink</a>	185
<a href="#">Method returnSee</a>	185
<a href="#">Method Sampize</a>	186
<a href="#">Method setSourcePaths</a>	186
<a href="#">Method setTargetDir</a>	186
<a href="#">Method setTemplateBase</a>	187
<a href="#">Method setTemplateDir</a>	187
<a href="#">Method sortPageContentsByElementType</a>	187
<a href="#">Method sourceLine</a>	188
<a href="#">Method startHighlight</a>	188
<a href="#">Method TranslateEntity</a>	188
<a href="#">Method TranslateTag</a>	188
<a href="#">Method TutorialExample</a>	189
<a href="#">Method type_adjust</a>	189
<a href="#">Method unmangle</a>	189
<a href="#">Method vardump_tree</a>	190
<a href="#">Method Varize</a>	190
<a href="#">Method walk</a>	190
<a href="#">Method walk_everything</a>	191
<a href="#">Method writeExample</a>	191
<a href="#">Method writeFile</a>	192
<a href="#">Method writeSource</a>	192
<a href="#">Method rmdir</a>	193
<a href="#">Method setHighlightCache</a>	193
<a href="#">Method tutorial_path</a>	193
<a href="#"><b>CHMdefaultConverter.inc</b></a>	194
<a href="#"><b>Class CHMdefaultConverter</b></a>	194
Var \$base_dir	195
Var \$class_dir	195
Var \$current	195
Var \$currentclass	195
Var \$juststarted	196
Var \$KLinks	196
Var \$leftindex	196
Var \$name	196
Var \$outputformat	196
Var \$package_pages	196
Var \$page_dir	197
Var \$ric_set	197
Var \$sort_page_contents_by_type	197
Var \$wrote	197
Constructor CHMdefaultConverter	197
Method addHHP	198
Method addKLink	198
Method addSourceTOC	198
Method addTOC	199
Method convertClass	199
Method convertConst	199
Method convertDefine	200

<a href="#">Method ConvertErrorLog</a>	200
<a href="#">Method convertFunction</a>	200
<a href="#">Method convertGlobal</a>	201
<a href="#">Method convertInclude</a>	201
<a href="#">Method convertMethod</a>	201
<a href="#">Method convertPackagepage</a>	201
<a href="#">Method convertPage</a>	202
<a href="#">Method ConvertTodoList</a>	202
<a href="#">Method convertTutorial</a>	202
<a href="#">Method convertVar</a>	202
<a href="#">Method Convert_RIC</a>	203
<a href="#">Method copyMediaRecursively</a>	203
<a href="#">Method endClass</a>	203
<a href="#">Method endPage</a>	203
<a href="#">Method formatIndex</a>	203
<a href="#">Method formatLeftIndex</a>	204
<a href="#">Method formatPkgIndex</a>	204
<a href="#">Method formatTutorialTOC</a>	205
<a href="#">Method generateElementIndex</a>	205
<a href="#">Method generateFormattedClassTree</a>	205
<a href="#">Method generateFormattedClassTrees</a>	205
<a href="#">Method generateFormattedInterfaceTrees</a>	206
<a href="#">Method generateKLinks</a>	206
<a href="#">Method generatePkgElementIndex</a>	206
<a href="#">Method generatePkgElementIndexes</a>	207
<a href="#">Method generateTOC</a>	207
<a href="#">Method getCData</a>	207
<a href="#">Method getClassLink</a>	207
<a href="#">Method getConstLink</a>	207
<a href="#">Method getCurrentPageLink</a>	208
<a href="#">Method getDefineLink</a>	208
<a href="#">Method getExampleLink</a>	208
<a href="#">Method getFunctionLink</a>	208
<a href="#">Method getGlobalLink</a>	209
<a href="#">Method getId</a>	209
<a href="#">Method getIndexInformation</a>	209
<a href="#">Method getMethodLink</a>	210
<a href="#">Method getPageLink</a>	210
<a href="#">Method getPageName</a>	210
<a href="#">Method getRootTree</a>	211
<a href="#">Method getSourceAnchor</a>	211
<a href="#">Method getSourceLink</a>	211
<a href="#">Method getTutorialId</a>	211
<a href="#">Method getVarLink</a>	212
<a href="#">Method Output</a>	212
<a href="#">Method postProcess</a>	213
<a href="#">Method ProgramExample</a>	213
<a href="#">Method rcNatCmp</a>	213
<a href="#">Method rcNatCmp1</a>	213

<a href="#">Method returnLink</a>	214
<a href="#">Method returnSee</a>	214
<a href="#">Method setTargetDir</a>	214
<a href="#">Method setTemplateDir</a>	214
<a href="#">Method SmartyInit</a>	214
<a href="#">Method sourceLine</a>	215
<a href="#">Method TutorialExample</a>	215
<a href="#">Method unmangle</a>	215
<a href="#">Method writeExample</a>	215
<a href="#">Method writefile</a>	216
<a href="#">Method writeNewPPage</a>	216
<a href="#">Method writeSource</a>	216
<a href="#">HTMLframesConverter.inc</a>	217
<a href="#">HTMLSmartyConverter.inc</a>	218
<a href="#">Class HTMLframesConverter</a>	219
<a href="#">Var \$base_dir</a>	219
<a href="#">Var \$class_dir</a>	219
<a href="#">Var \$current</a>	219
<a href="#">Var \$currentclass</a>	219
<a href="#">Var \$juststarted</a>	220
<a href="#">Var \$leftindex</a>	220
<a href="#">Var \$name</a>	220
<a href="#">Var \$outputformat</a>	220
<a href="#">Var \$package_pages</a>	220
<a href="#">Var \$page_dir</a>	220
<a href="#">Var \$processSpecialRoots</a>	221
<a href="#">Var \$ric_set</a>	221
<a href="#">Var \$sort_page_contents_by_type</a>	221
<a href="#">Var \$wrote</a>	221
<a href="#">Constructor HTMLframesConverter</a>	221
<a href="#">Method convertClass</a>	222
<a href="#">Method convertConst</a>	222
<a href="#">Method convertDefine</a>	222
<a href="#">Method ConvertErrorLog</a>	223
<a href="#">Method convertFunction</a>	223
<a href="#">Method convertGlobal</a>	223
<a href="#">Method convertInclude</a>	224
<a href="#">Method convertMethod</a>	224
<a href="#">Method convertPackagepage</a>	224
<a href="#">Method convertPage</a>	224
<a href="#">Method ConvertTodoList</a>	225
<a href="#">Method convertTutorial</a>	225
<a href="#">Method convertVar</a>	225
<a href="#">Method Convert_RIC</a>	225
<a href="#">Method copyMediaRecursively</a>	226
<a href="#">Method endClass</a>	226
<a href="#">Method endPage</a>	226
<a href="#">Method formatIndex</a>	226
<a href="#">Method formatLeftIndex</a>	227

<a href="#">Method formatPkgIndex</a>	227
<a href="#">Method formatTutorialTOC</a>	227
<a href="#">Method generateElementIndex</a>	228
<a href="#">Method generateFormattedClassTree</a>	228
<a href="#">Method generateFormattedClassTrees</a>	228
<a href="#">Method generateFormattedInterfaceTrees</a>	228
<a href="#">Method generatePkgElementIndex</a>	229
<a href="#">Method generatePkgElementIndexes</a>	229
<a href="#">Method getCData</a>	229
<a href="#">Method getClassLink</a>	229
<a href="#">Method getConstLink</a>	230
<a href="#">Method getCurrentPageLink</a>	230
<a href="#">Method getDefineLink</a>	230
<a href="#">Method getExampleLink</a>	231
<a href="#">Method getFunctionLink</a>	231
<a href="#">Method getGlobalLink</a>	231
<a href="#">Method getId</a>	232
<a href="#">Method getIndexInformation</a>	232
<a href="#">Method getMethodLink</a>	232
<a href="#">Method getPageLink</a>	232
<a href="#">Method getPageName</a>	233
<a href="#">Method getRootTree</a>	233
<a href="#">Method getSourceAnchor</a>	233
<a href="#">Method getSourceLink</a>	234
<a href="#">Method getTutorialId</a>	234
<a href="#">Method getTutorialTree</a>	234
<a href="#">Method getVarLink</a>	234
<a href="#">Method makeLeft</a>	235
<a href="#">Method Output</a>	235
<a href="#">Method postProcess</a>	235
<a href="#">Method ProgramExample</a>	235
<a href="#">Method rcNatCmp</a>	235
<a href="#">Method rcNatCmp1</a>	235
<a href="#">Method returnLink</a>	236
<a href="#">Method returnSee</a>	236
<a href="#">Method setTargetDir</a>	236
<a href="#">Method SmartyInit</a>	236
<a href="#">Method sourceLine</a>	237
<a href="#">Method TutorialExample</a>	237
<a href="#">Method unmangle</a>	237
<a href="#">Method writeExample</a>	237
<a href="#">Method writeNewPPage</a>	237
<a href="#">Method writeSource</a>	238
<a href="#">Class HTMLSmartyConverter</a>	238
<a href="#">Var \$base_dir</a>	238
<a href="#">Var \$class_dir</a>	239
<a href="#">Var \$current</a>	239
<a href="#">Var \$currentclass</a>	239
<a href="#">Var \$juststarted</a>	239

<a href="#">Var \$leftindex</a>	239
<a href="#">Var \$name</a>	239
<a href="#">Var \$outputformat</a>	239
<a href="#">Var \$package_pages</a>	239
<a href="#">Var \$page_dir</a>	240
<a href="#">Var \$processSpecialRoots</a>	240
<a href="#">Var \$ric_set</a>	240
<a href="#">Var \$sort_page_contents_by_type</a>	240
<a href="#">Var \$wrote</a>	240
<a href="#">Constructor HTMLSmartyConverter</a>	240
<a href="#">Method convertClass</a>	241
<a href="#">Method convertConst</a>	241
<a href="#">Method convertDefine</a>	241
<a href="#">Method ConvertErrorLog</a>	242
<a href="#">Method convertFunction</a>	242
<a href="#">Method convertGlobal</a>	242
<a href="#">Method convertInclude</a>	243
<a href="#">Method convertMethod</a>	243
<a href="#">Method convertPackagepage</a>	243
<a href="#">Method convertPage</a>	244
<a href="#">Method ConvertTodoList</a>	244
<a href="#">Method convertTutorial</a>	244
<a href="#">Method convertVar</a>	244
<a href="#">Method Convert_RIC</a>	245
<a href="#">Method copyMediaRecursively</a>	245
<a href="#">Method endClass</a>	245
<a href="#">Method endPage</a>	245
<a href="#">Method formatIndex</a>	245
<a href="#">Method formatLeftIndex</a>	246
<a href="#">Method formatPkgIndex</a>	246
<a href="#">Method formatTutorialTOC</a>	247
<a href="#">Method generateElementIndex</a>	247
<a href="#">Method generateFormattedClassTree</a>	247
<a href="#">Method generateFormattedClassTrees</a>	247
<a href="#">Method generateFormattedInterfaceTrees</a>	248
<a href="#">Method generatePkgElementIndex</a>	248
<a href="#">Method generatePkgElementIndexes</a>	248
<a href="#">Method getCData</a>	249
<a href="#">Method getClassContents</a>	249
<a href="#">Method getClassLeft</a>	249
<a href="#">Method getClassLink</a>	249
<a href="#">Method getConstLink</a>	249
<a href="#">Method getCurrentPageLink</a>	249
<a href="#">Method getDefineLink</a>	250
<a href="#">Method getExampleLink</a>	250
<a href="#">Method getFunctionLink</a>	250
<a href="#">Method getGlobalLink</a>	250
<a href="#">Method getId</a>	251
<a href="#">Method getIndexInformation</a>	251

<a href="#">Method getMethodLink</a>	251
<a href="#">Method getPageContents</a>	252
<a href="#">Method getPageLeft</a>	252
<a href="#">Method getPageLink</a>	252
<a href="#">Method getPageName</a>	252
<a href="#">Method getRootTree</a>	252
<a href="#">Method getSourceAnchor</a>	253
<a href="#">Method getSourceLink</a>	253
<a href="#">Method getTutorialId</a>	253
<a href="#">Method getTutorialList</a>	253
<a href="#">Method getTutorialTree</a>	253
<a href="#">Method getVarLink</a>	253
<a href="#">Method makeLeft</a>	254
<a href="#">Method Output</a>	254
<a href="#">Method postProcess</a>	254
<a href="#">Method ProgramExample</a>	254
<a href="#">Method rcNatCmp</a>	254
<a href="#">Method rcNatCmp1</a>	255
<a href="#">Method returnLink</a>	255
<a href="#">Method returnSee</a>	255
<a href="#">Method setTargetDir</a>	255
<a href="#">Method SmartyInit</a>	256
<a href="#">Method sourceLine</a>	256
<a href="#">Method TutorialExample</a>	256
<a href="#">Method unmangle</a>	256
<a href="#">Method writeExample</a>	257
<a href="#">Method writeNewPPage</a>	257
<a href="#">Method writeRIC</a>	257
<a href="#">Method writeSource</a>	257
<a href="#">class.phpdocpdf.php</a>	258
<a href="#">ParserPDF.inc</a>	259
<a href="#">Define PHPDOCUMENTOR PDF EVENT CONTENT</a>	259
<a href="#">Define PHPDOCUMENTOR PDF EVENT FONT</a>	259
<a href="#">Define PHPDOCUMENTOR PDF EVENT NEWPAGE</a>	260
<a href="#">Define PHPDOCUMENTOR PDF EVENT PDFFUNCTION</a>	260
<a href="#">Define PHPDOCUMENTOR PDF EVENT TEXT</a>	260
<a href="#">Define PHPDOCUMENTOR PDF STATE CONTENT</a>	260
<a href="#">Define PHPDOCUMENTOR PDF STATE FONT</a>	260
<a href="#">Define PHPDOCUMENTOR PDF STATE NEWPAGE</a>	260
<a href="#">Define PHPDOCUMENTOR PDF STATE PDFFUNCTION</a>	260
<a href="#">Define PHPDOCUMENTOR PDF STATE TEXT</a>	260
<a href="#">PDFdefaultConverter.inc</a>	261
<a href="#">Class PDFdefaultConverter</a>	262
<a href="#">Var \$classpackage_pagenums</a>	262
<a href="#">Var \$curclasspackage</a>	262
<a href="#">Var \$curpagepackage</a>	262
<a href="#">Var \$leftindex</a>	262
<a href="#">Var \$name</a>	262
<a href="#">Var \$outputformat</a>	262

<a href="#">Var \$pagepackage_pagenums</a>	263
<a href="#">Var \$pdf</a>	263
<a href="#">Var \$ric_set</a>	263
<a href="#">Var \$smarty_dir</a>	263
<a href="#">Var \$sort_absolutely_everything</a>	263
<a href="#">Var \$sourcecode</a>	263
<a href="#">Constructor PDFdefaultConverter</a>	263
<a href="#">Method convertClass</a>	264
<a href="#">Method convertConst</a>	264
<a href="#">Method convertDefine</a>	264
<a href="#">Method convertDocBlock</a>	264
<a href="#">Method convertFunction</a>	264
<a href="#">Method convertGlobal</a>	265
<a href="#">Method convertInclude</a>	265
<a href="#">Method convertMethod</a>	265
<a href="#">Method convertPackagepage</a>	265
<a href="#">Method convertPage</a>	265
<a href="#">Method convertParams</a>	265
<a href="#">Method convertTutorial</a>	266
<a href="#">Method convertVar</a>	266
<a href="#">Method Convert_RIC</a>	266
<a href="#">Method generateFormattedClassTrees</a>	266
<a href="#">Method getCData</a>	266
<a href="#">Method getClassLink</a>	267
<a href="#">Method getConstLink</a>	267
<a href="#">Method getDefineLink</a>	267
<a href="#">Method getExampleLink</a>	268
<a href="#">Method getFileSourceName</a>	268
<a href="#">Method getFunctionLink</a>	268
<a href="#">Method getGlobalLink</a>	268
<a href="#">Method getGlobalValue</a>	269
<a href="#">Method getMethodLink</a>	269
<a href="#">Method getPageLink</a>	269
<a href="#">Method getPageName</a>	270
<a href="#">Method getRootTree</a>	270
<a href="#">Method getSourceAnchor</a>	270
<a href="#">Method getSourceLink</a>	270
<a href="#">Method getState</a>	270
<a href="#">Method getTutorialId</a>	270
<a href="#">Method getVarLink</a>	271
<a href="#">Method mystrnatcasecmp</a>	271
<a href="#">Method Output</a>	271
<a href="#">Method postProcess</a>	271
<a href="#">Method returnLink</a>	272
<a href="#">Method returnSee</a>	272
<a href="#">Method setTemplateDir</a>	272
<a href="#">Method TranslateTag</a>	272
<a href="#">Method unmangle</a>	272
<a href="#">Method writeExample</a>	273

<a href="#">Method writeSource</a>	273
<a href="#">Class PDFParser</a>	273
<a href="#">Constructor PDFParser</a>	273
<a href="#">Method getParserEventName</a>	274
<a href="#">Method parse</a>	274
<a href="#">Method setupStates</a>	274
<a href="#">Class phpdcpdf</a>	274
<a href="#">Var \$converter</a>	275
<a href="#">Var \$font_dir</a>	275
<a href="#">Var \$indents</a>	275
<a href="#">Var \$indexContents</a>	275
<a href="#">Var \$listType</a>	275
<a href="#">Var \$reportContents</a>	275
<a href="#">Var \$set_pageNumbering</a>	275
<a href="#">Var \$colorStack</a>	275
<a href="#">Var \$save</a>	275
<a href="#">Constructor phpdcpdf</a>	275
<a href="#">Method addMessage</a>	276
<a href="#">Method bullet</a>	276
<a href="#">Method dots</a>	276
<a href="#">Method ezNewPage</a>	276
<a href="#">Method ezOutput</a>	276
<a href="#">Method ezProcessText</a>	276
<a href="#">Method ezText</a>	277
<a href="#">Method getColor</a>	277
<a href="#">Method getYPlusOffset</a>	277
<a href="#">Method indent</a>	277
<a href="#">Method index</a>	277
<a href="#">Method IndexLetter</a>	277
<a href="#">Method orderedBullet</a>	278
<a href="#">Method rf</a>	278
<a href="#">Method setColorArray</a>	278
<a href="#">Method setHTMLColor</a>	278
<a href="#">Method setupTOC</a>	278
<a href="#">Method textColor</a>	278
<a href="#">Method validHTMLColor</a>	278
<a href="#">Method _ezText</a>	278
<a href="#">XMLDocBookpeardoc2Converter.inc</a>	280
<a href="#">XMLDocBookConverter.inc</a>	281
<a href="#">Class XMLDocBookConverter</a>	281
<a href="#">Var \$base_dir</a>	281
<a href="#">Var \$category</a>	282
<a href="#">Var \$class</a>	282
<a href="#">Var \$class_data</a>	282
<a href="#">Var \$class_dir</a>	282
<a href="#">Var \$current</a>	282
<a href="#">Var \$currentclass</a>	282
<a href="#">Var \$function_data</a>	282
<a href="#">Var \$juststarted</a>	283

<a href="#">Var \$leftindex</a>	283
<a href="#">Var \$local</a>	283
<a href="#">Var \$method_data</a>	283
<a href="#">Var \$name</a>	283
<a href="#">Var \$outputformat</a>	283
<a href="#">Var \$package_pages</a>	283
<a href="#">Var \$page</a>	284
<a href="#">Var \$page_data</a>	284
<a href="#">Var \$page_dir</a>	284
<a href="#">Var \$path</a>	284
<a href="#">Var \$sort_page_contents_by_type</a>	284
<a href="#">Var \$sourceloc</a>	284
<a href="#">Var \$template_options</a>	284
<a href="#">Constructor XMLDocBookConverter</a>	285
<a href="#">Method convertClass</a>	285
<a href="#">Method convertDefine</a>	286
<a href="#">Method ConvertErrorLog</a>	286
<a href="#">Method convertFunction</a>	286
<a href="#">Method convertGlobal</a>	287
<a href="#">Method convertInclude</a>	287
<a href="#">Method convertMethod</a>	287
<a href="#">Method convertPackagePage</a>	288
<a href="#">Method convertPage</a>	288
<a href="#">Method convertTutorial</a>	288
<a href="#">Method convertVar</a>	288
<a href="#">Method endClass</a>	289
<a href="#">Method endPage</a>	289
<a href="#">Method formatIndex</a>	289
<a href="#">Method formatLeftIndex</a>	289
<a href="#">Method formatPkgIndex</a>	290
<a href="#">Method generateElementIndex</a>	290
<a href="#">Method generateFormattedClassTree</a>	290
<a href="#">Method generateFormattedClassTrees</a>	291
<a href="#">Method generatePkgElementIndex</a>	291
<a href="#">Method generatePkgElementIndexes</a>	292
<a href="#">Method getCData</a>	292
<a href="#">Method getClassLink</a>	292
<a href="#">Method getDefineLink</a>	292
<a href="#">Method getFunctionLink</a>	293
<a href="#">Method getGlobalLink</a>	293
<a href="#">Method getId</a>	293
<a href="#">Method getLink</a>	294
<a href="#">Method getMethodLink</a>	294
<a href="#">Method getPageLink</a>	294
<a href="#">Method getPageName</a>	295
<a href="#">Method getRootTree</a>	295
<a href="#">Method getTutorialId</a>	295
<a href="#">Method getVarLink</a>	295
<a href="#">Method makeLeft</a>	296

<u>Method Output</u>	296
<u>Method postProcess</u>	296
<u>Method prepareDocBlock</u>	296
<u>Method rcNatCmp</u>	296
<u>Method rcNatCmp1</u>	296
<u>Method returnLink</u>	297
<u>Method returnSee</u>	297
<u>Method setTargetDir</u>	297
<u>Method setTemplateDir</u>	297
<u>Method SmartyInit</u>	298
<u>Method type_adjust</u>	298
<u>Method unmangle</u>	298
<u>Method writeNewPPage</u>	298
<u>Class XMLDocBookpeardecor2Converter</u>	298
Var \$base_dir	299
Var \$category	299
Var \$class	299
Var \$class_data	299
Var \$class_dir	299
Var \$current	300
Var \$currentclass	300
Var \$function_data	300
Var \$juststarted	300
Var \$leftindex	300
Var \$local	300
Var \$method_data	300
Var \$name	301
Var \$outputformat	301
Var \$packagexml	301
Var \$package_pages	301
Var \$page	301
Var \$page_data	301
Var \$page_dir	301
Var \$path	301
Var \$processSpecialRoots	302
Var \$sort_absolutely_everything	302
Var \$sort_page_contents_by_type	302
Var \$sourceloc	302
Var \$template_options	302
Var \$peardecor2_constants	303
Var \$peardecor2_globals	303
Var \$write_constants_xml	303
Var \$write_globals_xml	303
Constructor XMLDocBookpeardecor2Converter	303
Method addSummaryToPackageXml	303
Method Br	304
Method convertClass	304
Method convertDefine	304
Method ConvertErrorLog	304

<a href="#">Method convertFunction</a>	305
<a href="#">Method convertGlobal</a>	305
<a href="#">Method convertInclude</a>	305
<a href="#">Method convertMethod</a>	305
<a href="#">Method convertPackagePage</a>	306
<a href="#">Method convertPage</a>	306
<a href="#">Method convertTutorial</a>	306
<a href="#">Method convertVar</a>	307
<a href="#">Method endClass</a>	307
<a href="#">Method exampleProgramExample</a>	307
<a href="#">Method flushPackageXml</a>	308
<a href="#">Method formatIndex</a>	308
<a href="#">Method formatLeftIndex</a>	308
<a href="#">Method formatPkgIndex</a>	308
<a href="#">Method generateChildClassList</a>	308
<a href="#">Method generateElementIndex</a>	308
<a href="#">Method generateFormattedClassTree</a>	309
<a href="#">Method generateFormattedClassTrees</a>	309
<a href="#">Method generateFormattedInterfaceTrees</a>	309
<a href="#">Method generatePkgElementIndex</a>	310
<a href="#">Method generatePkgElementIndexes</a>	310
<a href="#">Method getCData</a>	310
<a href="#">Method getClassLink</a>	310
<a href="#">Method getDefineLink</a>	311
<a href="#">Method getExampleLink</a>	311
<a href="#">Method getFunctionLink</a>	311
<a href="#">Method getGlobalLink</a>	312
<a href="#">Method getId</a>	312
<a href="#">Method getLink</a>	312
<a href="#">Method getMethodLink</a>	313
<a href="#">Method getPageLink</a>	313
<a href="#">Method getPageName</a>	313
<a href="#">Method getRootTree</a>	313
<a href="#">Method getSourceAnchor</a>	314
<a href="#">Method getTutorialId</a>	314
<a href="#">Method getVarLink</a>	314
<a href="#">Method makeLeft</a>	315
<a href="#">Method Output</a>	315
<a href="#">Method postProcess</a>	315
<a href="#">Method prepareDocBlock</a>	315
<a href="#">Method ProgramExample</a>	315
<a href="#">Method rcNatCmp</a>	316
<a href="#">Method rcNatCmp1</a>	316
<a href="#">Method returnLink</a>	316
<a href="#">Method returnSee</a>	316
<a href="#">Method setTemplateDir</a>	316
<a href="#">Method type_adjust</a>	317
<a href="#">Method unmangle</a>	317
<a href="#">Method wordwrap</a>	317

<a href="#">Method writeExample</a>	317
<a href="#">Method writeFile</a>	317
<a href="#">Method writeNewPPage</a>	318
<a href="#">Method writeSource</a>	318
<a href="#"><b>Package Cpdf Procedural Elements</b></a>	320
<a href="#">class.ezpdf.php</a>	320
<a href="#"><b>Package Cpdf Classes</b></a>	321
<a href="#">Class Cezpdf</a>	321
<a href="#">Var \$ez</a>	321
<a href="#">Var \$ezPageCount</a>	321
<a href="#">Var \$ezPages</a>	321
<a href="#">Var \$y</a>	321
<a href="#">Constructor Cezpdf</a>	321
<a href="#">Method alink</a>	322
<a href="#">Method execTemplate</a>	322
<a href="#">Method ezColumnsStart</a>	322
<a href="#">Method ezColumnsStop</a>	322
<a href="#">Method ezGetCurrentPageNumber</a>	322
<a href="#">Method ezImage</a>	322
<a href="#">Method ezInsertMode</a>	322
<a href="#">Method ezNewPage</a>	323
<a href="#">Method ezOutput</a>	323
<a href="#">Method ezProcessText</a>	323
<a href="#">Method ezPRVTaddPageNumbers</a>	323
<a href="#">Method ezPRVTcleanUp</a>	323
<a href="#">Method ezPrvtGetTextWidth</a>	323
<a href="#">Method ezPRVTpageNumberSearch</a>	323
<a href="#">Method ezPrvtTableColumnHeadings</a>	323
<a href="#">Method ezPrvtTableDrawLines</a>	324
<a href="#">Method ezSetCmMargins</a>	324
<a href="#">Method ezSetDy</a>	324
<a href="#">Method ezSetMargins</a>	324
<a href="#">Method ezSetY</a>	324
<a href="#">Method ezStartPageNumbers</a>	325
<a href="#">Method ezStopPageNumbers</a>	325
<a href="#">Method ezStream</a>	325
<a href="#">Method ezTable</a>	325
<a href="#">Method ezText</a>	325
<a href="#">Method ezWhatPageNumber</a>	326
<a href="#">Method ilink</a>	326
<a href="#">Method loadTemplate</a>	326
<a href="#">Method uline</a>	326
<a href="#">Class Cpdf</a>	326
<a href="#">Var \$addLooseObjects</a>	327
<a href="#">Var \$arc4</a>	327
<a href="#">Var \$arc4_objnum</a>	327
<a href="#">Var \$callback</a>	327
<a href="#">Var \$catalogId</a>	327

<a href="#">Var \$checkpoint</a>	327
<a href="#">Var \$currentBaseFont</a>	328
<a href="#">Var \$currentColour</a>	328
<a href="#">Var \$currentContents</a>	328
<a href="#">Var \$currentFont</a>	328
<a href="#">Var \$currentFontNum</a>	328
<a href="#">Var \$currentLineStyle</a>	328
<a href="#">Var \$currentNode</a>	328
<a href="#">Var \$currentPage</a>	329
<a href="#">Var \$currentStrokeColour</a>	329
<a href="#">Var \$currentTextState</a>	329
<a href="#">Var \$destinations</a>	329
<a href="#">Var \$encrypted</a>	329
<a href="#">Var \$encryptionKey</a>	329
<a href="#">Var \$fileIdentifier</a>	329
<a href="#">Var \$firstPageId</a>	330
<a href="#">Var \$fontFamilies</a>	330
<a href="#">Var \$fonts</a>	330
<a href="#">Var \$infoObject</a>	330
<a href="#">Var \$looseObjects</a>	330
<a href="#">Var \$messages</a>	330
<a href="#">Var \$nCallback</a>	330
<a href="#">Var \$nStack</a>	331
<a href="#">Var \$nStateStack</a>	331
<a href="#">Var \$numFonts</a>	331
<a href="#">Var \$numImages</a>	331
<a href="#">Var \$numObj</a>	331
<a href="#">Var \$numPages</a>	331
<a href="#">Var \$objects</a>	331
<a href="#">Var \$options</a>	332
<a href="#">Var \$procsetObjectId</a>	332
<a href="#">Var \$stack</a>	332
<a href="#">Var \$stateStack</a>	332
<a href="#">Var \$wordSpaceAdjust</a>	332
<a href="#">Constructor Cpdf</a>	332
<a href="#">Method addDestination</a>	333
<a href="#">Method addImage</a>	333
<a href="#">Method addInfo</a>	333
<a href="#">Method addInternalLink</a>	334
<a href="#">Method addJpegFromFile</a>	334
<a href="#">Method addLink</a>	334
<a href="#">Method addMessage</a>	334
<a href="#">Method addObject</a>	335
<a href="#">Method addPngFromFile</a>	335
<a href="#">Method addText</a>	335
<a href="#">Method addTextWrap</a>	335
<a href="#">Method ARC4</a>	336
<a href="#">Method ARC4_init</a>	336
<a href="#">Method checkAllHere</a>	336

<a href="#">Method closeObject</a>	336
<a href="#">Method curve</a>	336
<a href="#">Method ellipse</a>	337
<a href="#">Method encryptInit</a>	337
<a href="#">Method filledEllipse</a>	338
<a href="#">Method filledRectangle</a>	338
<a href="#">Method getFirstPageId</a>	338
<a href="#">Method getFontDecender</a>	338
<a href="#">Method getFontHeight</a>	339
<a href="#">Method getTextWidth</a>	339
<a href="#">Method line</a>	339
<a href="#">Method md5_16</a>	339
<a href="#">Method newPage</a>	339
<a href="#">Method openHere</a>	340
<a href="#">Method openObject</a>	340
<a href="#">Method output</a>	340
<a href="#">Method o_action</a>	340
<a href="#">Method o_annotation</a>	341
<a href="#">Method o_catalog</a>	341
<a href="#">Method o_contents</a>	341
<a href="#">Method o_destination</a>	341
<a href="#">Method o_encryption</a>	342
<a href="#">Method o_font</a>	342
<a href="#">Method o_fontDescriptor</a>	342
<a href="#">Method o_fontEncoding</a>	342
<a href="#">Method o_image</a>	343
<a href="#">Method o_info</a>	343
<a href="#">Method o_outlines</a>	343
<a href="#">Method o_page</a>	343
<a href="#">Method o_pages</a>	344
<a href="#">Method o_procset</a>	344
<a href="#">Method o_viewerPreferences</a>	344
<a href="#">Method partEllipse</a>	344
<a href="#">Method polygon</a>	345
<a href="#">Method rectangle</a>	345
<a href="#">Method reopenObject</a>	345
<a href="#">Method restoreState</a>	345
<a href="#">Method saveState</a>	346
<a href="#">Method selectFont</a>	346
<a href="#">Method setColor</a>	346
<a href="#">Method setEncryption</a>	347
<a href="#">Method setFontFamily</a>	347
<a href="#">Method setLineStyle</a>	347
<a href="#">Method setPreferences</a>	348
<a href="#">Method setStrokeColor</a>	348
<a href="#">Method stopObject</a>	348
<a href="#">Method stream</a>	348
<a href="#">Method transaction</a>	349
<a href="#"><b>Package HTML TreeMenu Procedural Elements</b></a>	351

<a href="#">file_dialog.php</a>	351
<a href="#">Define PHPDOC WEBROOT DIR</a>	352
<a href="#">Package HTML TreeMenu Classes</a>	353
<a href="#">Class DirNode</a>	353
<a href="#">Var \$path</a>	353
<a href="#">Constructor DirNode</a>	353
<a href="#">Class HTML_TreeMenu</a>	354
<a href="#">Var \$items</a>	354
<a href="#">Constructor HTML_TreeMenu</a>	354
<a href="#">Method addItem</a>	354
<a href="#">Class HTML_TreeMenu_DHTML</a>	355
<a href="#">Var \$defaultClass</a>	355
<a href="#">Var \$images</a>	355
<a href="#">Var \$isDynamic</a>	356
<a href="#">Var \$linkTarget</a>	356
<a href="#">Var \$noTopLevelImages</a>	356
<a href="#">Var \$userPersistence</a>	356
<a href="#">Constructor HTML_TreeMenu_DHTML</a>	356
<a href="#">Method toHTML</a>	357
<a href="#">Class HTML_TreeMenu_Listbox</a>	357
<a href="#">Var \$indentChar</a>	357
<a href="#">Var \$indentNum</a>	357
<a href="#">Var \$linkTarget</a>	358
<a href="#">Var \$promoText</a>	358
<a href="#">Constructor HTML_TreeMenu_Listbox</a>	358
<a href="#">Method toHTML</a>	358
<a href="#">Class HTML_TreeMenu_Presentation</a>	358
<a href="#">Var \$menu</a>	359
<a href="#">Constructor HTML_TreeMenu_Presentation</a>	359
<a href="#">Method printMenu</a>	359
<a href="#">Class HTMLTreeNode</a>	360
<a href="#">Var \$cssClass</a>	360
<a href="#">Var \$ensureVisible</a>	360
<a href="#">Var \$events</a>	360
<a href="#">Var \$expanded</a>	360
<a href="#">Var \$icon</a>	360
<a href="#">Var \$isDynamic</a>	361
<a href="#">Var \$items</a>	361
<a href="#">Var \$link</a>	361
<a href="#">Var \$parent</a>	361
<a href="#">Var \$text</a>	361
<a href="#">Constructor HTMLTreeNode</a>	361
<a href="#">Method addItem</a>	362
<a href="#">Method setOption</a>	362
<a href="#">Package org-phpdoc Procedural Elements</a>	365
<a href="#">bug-904820.php</a>	365
<a href="#">Define element</a>	365
<a href="#">Package Smarty Procedural Elements</a>	367

<a href="#">Config_File.class.php</a>	367
<a href="#">Smarty.class.php</a>	368
<a href="#">Define DIR_SEP</a>	368
<a href="#">Define SMARTY_DIR</a>	368
<a href="#">Define SMARTY_PHP_ALLOW</a>	369
<a href="#">Define SMARTY_PHP_PASSTHRU</a>	369
<a href="#">Define SMARTY_PHP_QUOTE</a>	369
<a href="#">Define SMARTY_PHP_REMOVE</a>	369
<a href="#">Smarty_Compiler.class.php</a>	370
<b>Package Smarty Classes</b>	371
<b><a href="#">Class Config_File</a></b>	371
<a href="#">Var \$booleanize</a>	371
<a href="#">Var \$fix_newlines</a>	371
<a href="#">Var \$overwrite</a>	371
<a href="#">Var \$read_hidden</a>	372
<a href="#">Var \$_config_data</a>	372
<a href="#">Constructor Config_File</a>	372
<a href="#">Method clear</a>	372
<a href="#">Method get</a>	372
<a href="#">Method get_file_names</a>	373
<a href="#">Method get_key</a>	373
<a href="#">Method get_section_names</a>	373
<a href="#">Method get_var_names</a>	373
<a href="#">Method load_file</a>	373
<a href="#">Method set_path</a>	374
<b><a href="#">Class Smarty</a></b>	374
<a href="#">Var \$autoload_filters</a>	374
<a href="#">Var \$cache_dir</a>	375
<a href="#">Var \$cache_handler_func</a>	375
<a href="#">Var \$cache_lifetime</a>	375
<a href="#">Var \$cache_modified_check</a>	375
<a href="#">Var \$caching</a>	375
<a href="#">Var \$compiler_class</a>	376
<a href="#">Var \$compiler_file</a>	376
<a href="#">Var \$compile_check</a>	376
<a href="#">Var \$compile_dir</a>	376
<a href="#">Var \$compile_id</a>	376
<a href="#">Var \$config_booleanize</a>	377
<a href="#">Var \$config_class</a>	377
<a href="#">Var \$config_dir</a>	377
<a href="#">Var \$config_fix_newlines</a>	377
<a href="#">Var \$config_overwrite</a>	377
<a href="#">Var \$config_read_hidden</a>	377
<a href="#">Var \$debugging</a>	378
<a href="#">Var \$debugging_ctrl</a>	378
<a href="#">Var \$debug_tpl</a>	378
<a href="#">Var \$default_modifiers</a>	378
<a href="#">Var \$default_resource_type</a>	379
<a href="#">Var \$default_template_handler_func</a>	379

<a href="#">Var \$force_compile</a>	379
<a href="#">Var \$global_assign</a>	379
<a href="#">Var \$left delimiter</a>	380
<a href="#">Var \$php_handling</a>	380
<a href="#">Var \$plugins_dir</a>	380
<a href="#">Var \$request_use_auto_globals</a>	380
<a href="#">Var \$request_vars_order</a>	381
<a href="#">Var \$right delimiter</a>	381
<a href="#">Var \$secure_dir</a>	381
<a href="#">Var \$security</a>	381
<a href="#">Var \$security_settings</a>	381
<a href="#">Var \$template_dir</a>	382
<a href="#">Var \$trusted_dir</a>	382
<a href="#">Var \$undefined</a>	382
<a href="#">Var \$use_sub_dirs</a>	382
<a href="#">Constructor Smarty</a>	382
<a href="#">Method append</a>	383
<a href="#">Method append_by_ref</a>	383
<a href="#">Method assign</a>	383
<a href="#">Method assign_by_ref</a>	383
<a href="#">Method clear_all_assign</a>	384
<a href="#">Method clear_all_cache</a>	384
<a href="#">Method clear_assign</a>	384
<a href="#">Method clear_cache</a>	384
<a href="#">Method clear_compiled_tpl</a>	384
<a href="#">Method clear_config</a>	385
<a href="#">Method config_load</a>	385
<a href="#">Method display</a>	385
<a href="#">Method fetch</a>	385
<a href="#">Method get_config_vars</a>	386
<a href="#">Method get_registered_object</a>	386
<a href="#">Method get_template_vars</a>	386
<a href="#">Method is_cached</a>	386
<a href="#">Method load_filter</a>	386
<a href="#">Method register_block</a>	387
<a href="#">Method register_compiler_function</a>	387
<a href="#">Method register_function</a>	387
<a href="#">Method register_modifier</a>	388
<a href="#">Method register_object</a>	388
<a href="#">Method register_outputfilter</a>	388
<a href="#">Method register_postfilter</a>	388
<a href="#">Method register_prefilter</a>	388
<a href="#">Method register_resource</a>	389
<a href="#">Method template_exists</a>	389
<a href="#">Method trigger_error</a>	389
<a href="#">Method unregister_block</a>	389
<a href="#">Method unregister_compiler_function</a>	390
<a href="#">Method unregister_function</a>	390
<a href="#">Method unregister_modifier</a>	390

<a href="#">Method unregister_object</a>	390
<a href="#">Method unregister_outputfilter</a>	390
<a href="#">Method unregister_postfilter</a>	391
<a href="#">Method unregister_prefilter</a>	391
<a href="#">Method unregister_resource</a>	391
<a href="#">Method compile_resource</a>	391
<a href="#">Method compile_source</a>	391
<a href="#">Method dequote</a>	392
<a href="#">Method eval</a>	392
<a href="#">Method fetch_resource_info</a>	392
<a href="#">Method get_auto_filename</a>	393
<a href="#">Method get_auto_id</a>	393
<a href="#">Method get_compile_path</a>	393
<a href="#">Method get_plugin_filepath</a>	393
<a href="#">Method include</a>	394
<a href="#">Method is_compiled</a>	394
<a href="#">Method parse_resource_name</a>	394
<a href="#">Method process_compiled_include_callback</a>	394
<a href="#">Method read_file</a>	395
<a href="#">Method run_mod_handler</a>	395
<a href="#">Method smarty_cache_attrs</a>	395
<a href="#">Method smarty_include</a>	395
<a href="#">Method trigger_fatal_error</a>	396
<a href="#">Method unlink</a>	396
<a href="#"><b>Class Smarty_Compiler</b></a>	396
<a href="#">Constructor Smarty_Compiler</a>	396
<a href="#">Method add_plugin</a>	397
<a href="#">Method compile_arg_list</a>	397
<a href="#">Method compile_block_tag</a>	397
<a href="#">Method compile_capture_tag</a>	397
<a href="#">Method compile_compiler_tag</a>	398
<a href="#">Method compile_custom_tag</a>	398
<a href="#">Method compile_file</a>	398
<a href="#">Method compile_foreach_start</a>	398
<a href="#">Method compile_if_tag</a>	399
<a href="#">Method compile_include_php_tag</a>	399
<a href="#">Method compile_include_tag</a>	399
<a href="#">Method compile_insert_tag</a>	399
<a href="#">Method compile_plugin_call</a>	399
<a href="#">Method compile_registered_object_tag</a>	400
<a href="#">Method compile_section_start</a>	400
<a href="#">Method compile_smarty_ref</a>	400
<a href="#">Method compile_tag</a>	400
<a href="#">Method expand_quoted_text</a>	401
<a href="#">Method load_filters</a>	401
<a href="#">Method parseAttrs</a>	401
<a href="#">Method parseConfVar</a>	401
<a href="#">Method parseIsExpr</a>	401
<a href="#">Method parseModifiers</a>	402

Method parse_parenth_args . . . . .	402
Method parse_section_prop . . . . .	402
Method parse_var . . . . .	402
Method parse_vars_props . . . . .	403
Method parse_var_props . . . . .	403
Method pop_cacheable_state . . . . .	403
Method push_cacheable_state . . . . .	403
Method quote_replace . . . . .	403
Method syntax_error . . . . .	404
core.assemble plugin_filepath.php . . . . .	405
Function smarty_core_assemble_plugin_filepath . . . . .	405
core.assign smarty_interface.php . . . . .	406
Function smarty_core_assign_smarty_interface . . . . .	406
core.create_dir_structure.php . . . . .	407
Function smarty_core_create_dir_structure . . . . .	407
core.display_debug_console.php . . . . .	408
Function smarty_core_display_debug_console . . . . .	408
core.get_include_path.php . . . . .	409
Function smarty_core_get_include_path . . . . .	409
core.get_microtime.php . . . . .	410
Function smarty_core_get_microtime . . . . .	410
core.get_php_resource.php . . . . .	411
Function smarty_core_get_php_resource . . . . .	411
core.is_secure.php . . . . .	412
Function smarty_core_is_secure . . . . .	412
core.is_trusted.php . . . . .	413
Function smarty_core_is_trusted . . . . .	413
core.load_plugins.php . . . . .	414
Function smarty_core_load_plugins . . . . .	414
core.load_resource_plugin.php . . . . .	415
Function smarty_core_load_resource_plugin . . . . .	415
core.process_cached_inserts.php . . . . .	416
Function smarty_core_process_cached_inserts . . . . .	416
core.process_compiled_include.php . . . . .	417
Function smarty_core_process_compiled_include . . . . .	417
core.read_cache_file.php . . . . .	418
Function smarty_core_read_cache_file . . . . .	418
core.rmdir.php . . . . .	419
Function smarty_core_rmdir . . . . .	419
core.rm_auto.php . . . . .	420
Function smarty_core_rm_auto . . . . .	420
core.run_insert_handler.php . . . . .	421
Function smarty_core_run_insert_handler . . . . .	421
core.smarty_include_php.php . . . . .	422
Function smarty_core_smarty_include_php . . . . .	422
core.write_cache_file.php . . . . .	423
Function smarty_core_write_cache_file . . . . .	423
core.write_compiled_include.php . . . . .	424
Function smarty_core_write_compiled_include . . . . .	424

<a href="#">core.write_compiled_resource.php</a>	425
Function smarty_core_write_compiled_resource	425
<a href="#">core.write_file.php</a>	426
Function smarty_core_write_file	426
<a href="#">block.strip.php</a>	427
Function smarty_block_strip	427
<a href="#">block.textformat.php</a>	428
Function smarty_block_textformat	428
<a href="#">function.assign.php</a>	429
Function smarty_function_assign	429
<a href="#">function.assign_debug_info.php</a>	430
Function smarty_function_assign_debug_info	430
<a href="#">function.config_load.php</a>	431
Function smarty_function_config_load	431
<a href="#">function.counter.php</a>	432
Function smarty_function_counter	432
<a href="#">function.cycle.php</a>	433
Function smarty_function_cycle	433
<a href="#">function.debug.php</a>	435
Function smarty_function_debug	435
<a href="#">function.eval.php</a>	436
Function smarty_function_eval	436
<a href="#">function.fetch.php</a>	437
Function smarty_function_fetch	437
<a href="#">function.html_checkboxes.php</a>	438
Function smarty_function_html_checkboxes	438
Function smarty_function_html_checkboxes_output	439
<a href="#">function.html_image.php</a>	440
Function smarty_function_html_image	440
<a href="#">function.html_options.php</a>	442
Function smarty_function_html_options	442
Function smarty_function_html_options_optgroup	442
Function smarty_function_html_options_optoutput	443
<a href="#">function.html_radios.php</a>	444
Function smarty_function_html_radios	444
Function smarty_function_html_radios_output	445
<a href="#">function.html_select_date.php</a>	446
Function smarty_function_html_select_date	446
<a href="#">function.html_select_time.php</a>	448
Function smarty_function_html_select_time	448
<a href="#">function.html_table.php</a>	449
Function smarty_function_html_table	449
Function smarty_function_html_table_cycle	450
<a href="#">function.mailto.php</a>	451
Function smarty_function_mailto	451
<a href="#">function.math.php</a>	453
Function smarty_function_math	453
<a href="#">function.popup.php</a>	454
Function smarty_function_popup	454

<a href="#">function.popup_init.php</a>	455
<a href="#">Function smarty function popup_init</a>	455
<a href="#">function.var_dump.php</a>	456
<a href="#">Function smarty function var_dump</a>	456
<a href="#">modifier.capitalize.php</a>	457
<a href="#">Function smarty modifier capitalize</a>	457
<a href="#">modifier.cat.php</a>	458
<a href="#">Function smarty modifier cat</a>	458
<a href="#">modifier.count_characters.php</a>	459
<a href="#">Function smarty modifier count_characters</a>	459
<a href="#">modifier.count_paragraphs.php</a>	460
<a href="#">Function smarty modifier count_paragraphs</a>	460
<a href="#">modifier.count_sentences.php</a>	461
<a href="#">Function smarty modifier count_sentences</a>	461
<a href="#">modifier.count_words.php</a>	462
<a href="#">Function smarty modifier count_words</a>	462
<a href="#">modifier.date_format.php</a>	463
<a href="#">Function smarty modifier date_format</a>	463
<a href="#">modifier.debug_print_var.php</a>	464
<a href="#">Function smarty modifier debug_print_var</a>	464
<a href="#">modifier.default.php</a>	465
<a href="#">Function smarty modifier default</a>	465
<a href="#">modifier.escape.php</a>	466
<a href="#">Function smarty modifier escape</a>	466
<a href="#">modifier.htmlentities.php</a>	467
<a href="#">Function smarty modifier htmlentities</a>	467
<a href="#">modifier.indent.php</a>	468
<a href="#">Function smarty modifier indent</a>	468
<a href="#">modifier.lower.php</a>	469
<a href="#">Function smarty modifier lower</a>	469
<a href="#">modifier.nl2br.php</a>	470
<a href="#">Function smarty modifier nl2br</a>	470
<a href="#">modifier.rawurlencode.php</a>	471
<a href="#">Function smarty modifier rawurlencode</a>	471
<a href="#">modifier.regex_replace.php</a>	472
<a href="#">Function smarty modifier regex_replace</a>	472
<a href="#">modifier.replace.php</a>	473
<a href="#">Function smarty modifier replace</a>	473
<a href="#">modifier.spacify.php</a>	474
<a href="#">Function smarty modifier spacify</a>	474
<a href="#">modifier.string_format.php</a>	475
<a href="#">Function smarty modifier string_format</a>	475
<a href="#">modifier.strip.php</a>	476
<a href="#">Function smarty modifier strip</a>	476
<a href="#">modifier.strip_tags.php</a>	477
<a href="#">Function smarty modifier strip_tags</a>	477
<a href="#">modifier.truncate.php</a>	478
<a href="#">Function smarty modifier truncate</a>	478
<a href="#">modifier.upper.php</a>	479

<a href="#">Function smarty_modifier_upper</a>	479
<a href="#">modifier.wordwrap.php</a>	480
<a href="#">Function smarty_modifier_wordwrap</a>	480
<a href="#">outputfilter.trimwhitespace.php</a>	481
<a href="#">Function smarty_outputfilter_trimwhitespace</a>	481
<a href="#">Function smarty_outputfilter_trimwhitespace_replace</a>	481
<a href="#">shared.escape_special_chars.php</a>	483
<a href="#">Function smarty_function_escape_special_chars</a>	483
<a href="#">shared.make_timestamp.php</a>	484
<a href="#">Function smarty_make_timestamp</a>	484
<b><a href="#">Package tests Procedural Elements</a></b>	486
<a href="#">bug-441275.php</a>	486
<a href="#">Function test2_441275</a>	486
<a href="#">Function test_441275</a>	486
<a href="#">bug-441278.php</a>	487
<a href="#">Function test_441278</a>	487
<a href="#">bug-441287.php</a>	488
<a href="#">Function test_441287</a>	488
<a href="#">Function test_4412872</a>	488
<a href="#">bug-441289.php</a>	489
<a href="#">Function test6</a>	489
<a href="#">Function test7</a>	489
<a href="#">Function test8</a>	489
<a href="#">Function test_441289</a>	489
<a href="#">Function test_4412892</a>	489
<a href="#">Function test_4412893</a>	489
<a href="#">Function test_4412894</a>	490
<a href="#">Function test_4412895</a>	490
<a href="#">Function test_bug_567455</a>	490
<a href="#">Function test_bug_567455_2</a>	490
<a href="#">bug-441433.php</a>	491
<a href="#">Function SendEMail</a>	491
<a href="#">Function test_441433</a>	491
<a href="#">bug-443153.php</a>	492
<a href="#">Function test_443153</a>	492
<a href="#">bug-445298.php</a>	493
<a href="#">Function test_445298</a>	493
<a href="#">bug-445305.php</a>	494
<a href="#">Function test_445305</a>	494
<a href="#">bug-445820.php</a>	495
<a href="#">Function test_445820</a>	495
<a href="#">bug-540368.php</a>	496
<a href="#">Function blah</a>	496
<a href="#">bug-542586.php</a>	497
<a href="#">Function func</a>	497
<a href="#">Define testie</a>	497
<a href="#">bug-550489.php</a>	498
<a href="#">Define thisisdumbbutworks</a>	498
<a href="#">bug-551120.php</a>	499

<a href="#">bug-553137.php</a>	500
<a href="#">Function func1</a>	500
<a href="#">Function func2</a>	500
<a href="#">Function func3</a>	500
<a href="#">bug-554712.php</a>	501
<a href="#">Function passbyref</a>	501
<a href="#">Function returnsme</a>	501
<a href="#">bug-557861.php</a>	502
<a href="#">bug-558031.php</a>	503
<a href="#">Function test 558031</a>	503
<a href="#">bug-558051.php</a>	504
<a href="#">Function one</a>	504
<a href="#">bug-559467.php</a>	505
<a href="#">Define CONSTANT1</a>	505
<a href="#">Define CONSTANT2</a>	505
<a href="#">bug-559494.php</a>	506
<a href="#">bug-559668.php</a>	507
<a href="#">Config</a>	508
<a href="#">Global Variable \$ MLIB GLOBAL DSN</a>	508
<a href="#">Define MLIB GLOBAL DEBUG</a>	508
<a href="#">Define MLIB INCLUDE PATH</a>	509
<a href="#">Define MLIB LOG FILE</a>	509
<a href="#">Define MLIB SYSLOG PRIORITY</a>	509
<a href="#">Define MLIB TEMPLATE PATH</a>	509
<a href="#">Define MLIB USE SYSLOG</a>	510
<a href="#">bug-560578.php</a>	513
<a href="#">bug-560595.php</a>	514
<a href="#">Define one</a>	514
<a href="#">bug-562997.php</a>	515
<a href="#">Define PDERROR MULTIPLE PARENT</a>	515
<a href="#">bug-566200.php</a>	516
<a href="#">Function ezStartPageNumbers</a>	516
<a href="#">bug-566600.php</a>	517
<a href="#">Function a</a>	517
<a href="#">bug-567059.php</a>	518
<a href="#">Function aa 567059</a>	518
<a href="#">bug-645588.php</a>	519
<a href="#">Function globalGetVar</a>	519
<a href="#">Function globalSetVar</a>	519
<a href="#">bug-698356.php</a>	521
<a href="#">Function bug698356 Output</a>	521
<a href="#">bug-authoremail.php</a>	522
<a href="#">Function test authoremail</a>	522
<a href="#">bug-defineparse.php</a>	523
<a href="#">Define SMART PATH DELIMITER</a>	523
<a href="#">bug-eofquotes.php</a>	524
<a href="#">Function test eofquotes</a>	524
<a href="#">bug-escaping.php</a>	525
<a href="#">Function test escape</a>	525

<a href="#">Function test_escape2</a>	525
<a href="#">bug-loseprocedural.php</a>	526
<a href="#">Function test</a>	526
<a href="#">Function test2</a>	526
<a href="#">bug-pageleveldocsblocks.php</a>	527
<a href="#">Function dummy</a>	527
<a href="#">bug-quote_new_parser.php</a>	528
<a href="#">Global Variable \$bqnp_tester</a>	528
<a href="#">Define bqnp_tester</a>	528
<a href="#">Function bqnp_testie</a>	528
<a href="#">bug-shortdesc.php</a>	529
<a href="#">Define testContantBlah</a>	529
<a href="#">Define testContantBlah2</a>	529
<a href="#">Define testContantBlah3</a>	529
<a href="#">Define testContantBlah4</a>	529
<a href="#">Define testContantBlah5</a>	529
<a href="#">Package tests Classes</a>	530
<a href="#">Class a</a>	530
<a href="#">Var \$a</a>	530
<a href="#">Var \$c</a>	530
<a href="#">Method b</a>	530
<a href="#">Class b553607_Parser</a>	530
<a href="#">Class baby</a>	531
<a href="#">Var \$oopsieindexing</a>	531
<a href="#">Class brokenlinkstovars</a>	531
<a href="#">Var \$broken</a>	531
<a href="#">Constructor brokenlinkstovars</a>	531
<a href="#">Class bug540341</a>	531
<a href="#">Method get_header2</a>	532
<a href="#">Class bug557861</a>	532
<a href="#">Class bug_489398</a>	532
<a href="#">Var \$test_01</a>	532
<a href="#">Var \$test_02</a>	533
<a href="#">Var \$test_03</a>	533
<a href="#">Var \$test_04</a>	533
<a href="#">Var \$test_05</a>	533
<a href="#">Class childofpriv</a>	533
<a href="#">Class ClubBase</a>	534
<a href="#">Var \$bPrintFlush</a>	534
<a href="#">Constructor ClubBase</a>	534
<a href="#">Method getAllProperties</a>	534
<a href="#">Method getProperty</a>	535
<a href="#">Method getPropType</a>	535
<a href="#">Method loadClass</a>	535
<a href="#">Method printFlush</a>	536
<a href="#">Method printVar</a>	536
<a href="#">Method setDebug</a>	536
<a href="#">Method setProperty</a>	536
<a href="#">Method _ERROR</a>	537

<u>Method PHPDOCUMENTOR_DEBUG</u>	537
<u>Class ctest</u>	538
<u>Var \$t1</u>	538
<u>Var \$t3</u>	538
<u>Constructor ctest</u>	538
<u>Method btest</u>	538
<u>Class few</u>	538
<u>Var \$pfh</u>	539
<u>Class functionincomment</u>	539
<u>Method process</u>	539
<u>Class iConverter</u>	539
<u>Method walk</u>	540
<u>Class iHTMLConverter</u>	540
<u>Method Convert</u>	541
<u>Class iiparserBase</u>	541
<u>Var \$type</u>	542
<u>Var \$value</u>	542
<u>Method getType</u>	542
<u>Method getValue</u>	542
<u>Method setValue</u>	542
<u>Class iNewRender</u>	542
<u>Var \$classpackage</u>	542
<u>Var \$classpackages</u>	543
<u>Var \$classsubpackage</u>	543
<u>Var \$classtree</u>	543
<u>Var \$class_children</u>	543
<u>Var \$data</u>	544
<u>Var \$elements</u>	544
<u>Var \$event_handlers</u>	544
<u>Var \$links</u>	544
<u>Var \$methods</u>	545
<u>Var \$packageoutput</u>	545
<u>Var \$pages</u>	545
<u>Var \$parsePrivate</u>	546
<u>Var \$pkg_elements</u>	546
<u>Var \$private_class</u>	546
<u>Var \$quietMode</u>	546
<u>Var \$targetDir</u>	546
<u>Var \$vars</u>	547
<u>Class iParser</u>	547
<u>Class iparserElement</u>	547
<u>Class kiddie_b587733</u>	548
<u>Constructor kiddie_b587733</u>	548
<u>Class mama</u>	548
<u>Class metoo</u>	548
<u>Var \$mine</u>	548
<u>Class multipl</u>	549
<u>Var \$manyvars</u>	549
<u>Method func</u>	549

<a href="#">Class parent</a>	b587733	549
<a href="#">Class priv1</a>		549
<a href="#">Class RecordWarning</a>		550
<a href="#">Class summary_form</a>		550
<a href="#">Var \$dp</a>		550
<a href="#">Method blah</a>		550
<a href="#">Method get_header2</a>		550
<a href="#">Class test</a>		551
<a href="#">Class test2</a>		551
<a href="#">Class testarraybug</a>		551
<a href="#">Var \$arrayType</a>		552
<a href="#">Var \$arrayType1</a>		552
<a href="#">Var \$myarrayName</a>		552
<a href="#">Var \$myarrayName1</a>		552
<a href="#">Class testClass</a>		552
<a href="#">Constructor testClass</a>		552
<a href="#">Class testme</a>		552
<a href="#">Var \$me</a>		553
<a href="#">bug-541886.php</a>		554
<a href="#">Class test</a>	541886	554
<a href="#">Class notseen</a>		554
<a href="#">HighlightParserGetInlineTagsTests.php</a>		555
<a href="#">IntermediateParserAddPrivatePageTests.php</a>		557
<a href="#">ParserClassGetSourceLocationTests.php</a>		559
<a href="#">ParserPageGetSourceLocationTests.php</a>		561
<a href="#">phpDocumentorSetupCleanConverterNamePieceTests.php</a>		563
<a href="#">phpDocumentorSetupDecideOnOrOffTests.php</a>		565
<a href="#">Define PHPDOCUMENTOR_BASE</a>		565
<a href="#">Define PHPUnit_MAIN_METHOD</a>		565
<a href="#">phpDocumentorTParserGetInlineTagsTests.php</a>		567
<a href="#">Class tests</a>	<a href="#">HighlightParserGetInlineTagsTests</a>	568
<a href="#">Method main</a>		568
<a href="#">Method setUp</a>		568
<a href="#">Method tearDown</a>		569
<a href="#">Method testShowCorrectBehaviorWhenGivenOneEmptyArg</a>		569
<a href="#">Method testShowCorrectBehaviorWhenGivenOneEmptyArgAndFalse</a>		569
<a href="#">Method testShowCorrectBehaviorWhenGivenOneEmptyArgAndTrue</a>		569
<a href="#">Class tests</a>	<a href="#">IntermediateParserAddPrivatePageTests</a>	570
<a href="#">Method main</a>		570
<a href="#">Method setUp</a>		570
<a href="#">Method tearDown</a>		571
<a href="#">Method testShowCorrectBehaviorWhenPrivatePageArrayIsEmpty</a>		571
<a href="#">Method testShowCorrectBehaviorWhenPrivatePageArrayIsNotAlreadyEmpty</a>		571
<a href="#">Class tests</a>	<a href="#">ParserClassGetSourceLocationTests</a>	572
<a href="#">Method main</a>		572
<a href="#">Method setUp</a>		572
<a href="#">Method tearDown</a>		572
<a href="#">Method testWhenLocationNotSetAndPearizeFalse</a>		573
<a href="#">Method testWhenLocationNotSetAndPearizeNull</a>		573

<u>Method testWhenLocationNotSetAndPearizeTrue</u>	573
<u>Method testWhenNonPearLocationSetAndPearizeFalse</u>	574
<u>Method testWhenNonPearLocationSetAndPearizeNull</u>	574
<u>Method testWhenNonPearLocationSetAndPearizeTrue</u>	574
<u>Method testWhenNonPearLocationSetIncludingDotsAndPearizeFalse</u>	575
<u>Method testWhenNonPearLocationSetIncludingDotsAndPearizeNull</u>	575
<u>Method testWhenNonPearLocationSetIncludingDotsAndPearizeTrue</u>	575
<u>Method testWhenNonPearRelativeLocationSetAndPearizeFalse</u>	575
<u>Method testWhenNonPearRelativeLocationSetAndPearizeNull</u>	576
<u>Method testWhenNonPearRelativeLocationSetAndPearizeTrue</u>	576
<u>Method testWhenPearLocationSetAndPearizeFalse</u>	576
<u>Method testWhenPearLocationSetAndPearizeNull</u>	577
<u>Method testWhenPearLocationSetAndPearizeTrue</u>	577
<u>Method testWhenPearLocationSetIncludingDotsAndPearizeFalse</u>	577
<u>Method testWhenPearLocationSetIncludingDotsAndPearizeNull</u>	578
<u>Method testWhenPearLocationSetIncludingDotsAndPearizeTrue</u>	578
<u>Method testWhenPearRelativeLocationSetAndPearizeFalse</u>	578
<u>Method testWhenPearRelativeLocationSetAndPearizeNull</u>	578
<u>Method testWhenPearRelativeLocationSetAndPearizeTrue</u>	579
<u>Class tests_ParserPageGetSourceLocationTests</u>	579
<u>Method main</u>	579
<u>Method setUp</u>	580
<u>Method tearDown</u>	580
<u>Method testWhenLocationNotSetAndPearizeFalse</u>	580
<u>Method testWhenLocationNotSetAndPearizeNull</u>	580
<u>Method testWhenLocationNotSetAndPearizeTrue</u>	581
<u>Method testWhenNonPearLocationSetAndPearizeFalse</u>	581
<u>Method testWhenNonPearLocationSetAndPearizeNull</u>	581
<u>Method testWhenNonPearLocationSetAndPearizeTrue</u>	582
<u>Method testWhenNonPearLocationSetIncludingDotsAndPearizeFalse</u>	582
<u>Method testWhenNonPearLocationSetIncludingDotsAndPearizeNull</u>	582
<u>Method testWhenNonPearLocationSetIncludingDotsAndPearizeTrue</u>	583
<u>Method testWhenNonPearRelativeLocationSetAndPearizeFalse</u>	583
<u>Method testWhenNonPearRelativeLocationSetAndPearizeNull</u>	583
<u>Method testWhenNonPearRelativeLocationSetAndPearizeTrue</u>	584
<u>Method testWhenPearLocationSetAndPearizeFalse</u>	584
<u>Method testWhenPearLocationSetAndPearizeNull</u>	584
<u>Method testWhenPearLocationSetAndPearizeTrue</u>	585
<u>Method testWhenPearLocationSetIncludingDotsAndPearizeFalse</u>	585
<u>Method testWhenPearLocationSetIncludingDotsAndPearizeNull</u>	585
<u>Method testWhenPearLocationSetIncludingDotsAndPearizeTrue</u>	585
<u>Method testWhenPearRelativeLocationSetAndPearizeFalse</u>	586
<u>Method testWhenPearRelativeLocationSetAndPearizeNull</u>	586
<u>Method testWhenPearRelativeLocationSetAndPearizeTrue</u>	586
<u>Class tests_phpDocumentorSetupCleanConverterNamePieceTests</u>	587
<u>Method main</u>	587
<u>Method setUp</u>	587
<u>Method tearDown</u>	588
<u>Method testDoNotAllowTruePathingOnPrimaryWithOneArg</u>	588

<a href="#">Method testDoNotAllowTruePathingOnPrimaryWithTwoArgs</a>	588
<a href="#">Method testDoNotAllowTruePathingOnSecondary</a>	589
<a href="#">Method testDoNotAllowTruePathingOnTertiary</a>	589
<a href="#">Method testExtremeExampleAndInvalidSecondary</a>	589
<a href="#">Method testExtremeExampleAndInvalidTertiaryA</a>	590
<a href="#">Method testExtremeExampleAndInvalidTertiaryB</a>	590
<a href="#">Method testExtremeExampleButValidPrimaryWithOneArg</a>	590
<a href="#">Method testExtremeExampleButValidPrimaryWithTwoArgs</a>	591
<a href="#">Method testExtremeExampleButValidSecondary</a>	591
<a href="#">Method testNormalSecondaryDefault</a>	591
<a href="#">Method testNormalSecondaryDocbookPeardoc2</a>	591
<a href="#">Method testNormalSecondaryFrames</a>	592
<a href="#">Method testNormalSecondarySmarty</a>	592
<a href="#">Method testNormalTertiaryDefault</a>	592
<a href="#">Method testNormalTertiaryDomDefault</a>	593
<a href="#">Method testNormalTertiaryDomEarthli</a>	593
<a href="#">Method testNormalTertiaryDomL0I33t</a>	593
<a href="#">Method testNormalTertiaryDomPhpdocde</a>	594
<a href="#">Method testNormalTertiaryDomPhphmllib</a>	594
<a href="#">Method testNormalTertiaryEarthli</a>	594
<a href="#">Method testNormalTertiaryHands</a>	595
<a href="#">Method testNormalTertiaryL0I33t</a>	595
<a href="#">Method testNormalTertiaryPear</a>	595
<a href="#">Method testNormalTertiaryPhp</a>	595
<a href="#">Method testNormalTertiaryPhpdocde</a>	596
<a href="#">Method testNormalTertiaryPhphmllib</a>	596
<a href="#">Method testNormalWithOneArgPrimaryCHM</a>	596
<a href="#">Method testNormalWithOneArgPrimaryHTML</a>	597
<a href="#">Method testNormalWithOneArgPrimaryPDF</a>	597
<a href="#">Method testNormalWithOneArgPrimaryXML</a>	597
<a href="#">Method testNormalWithTwoArgsPrimaryCHM</a>	598
<a href="#">Method testNormalWithTwoArgsPrimaryHTML</a>	598
<a href="#">Method testNormalWithTwoArgsPrimaryPDF</a>	598
<a href="#">Method testNormalWithTwoArgsPrimaryXML</a>	599
<a href="#">Method testPreventUpToParentPathingOnPrimaryWithOneArg</a>	599
<a href="#">Method testPreventUpToParentPathingOnPrimaryWithTwoArgs</a>	599
<a href="#">Method testPreventUpToParentPathingOnSecondary</a>	600
<a href="#">Method testPreventUpToParentPathingOnTertiary</a>	600
<a href="#">Method testUserDefinedTertiaryValue</a>	600
<a href="#"><b>Class tests_phpDocumentorSetupDecideOnOrOffTests</b></a>	601
<a href="#">Method main</a>	601
<a href="#">Method setUp</a>	601
<a href="#">Method tearDown</a>	601
<a href="#">Method testBasicOff</a>	602
<a href="#">Method testBasicOn</a>	602
<a href="#">Method testFuzzyEmpty</a>	602
<a href="#">Method testFuzzyFalseA</a>	602
<a href="#">Method testFuzzyFalseB</a>	603
<a href="#">Method testFuzzyFalseC</a>	603

<a href="#">Method testFuzzyNoA</a>	603
<a href="#">Method testFuzzyNoB</a>	603
<a href="#">Method testFuzzyNoC</a>	604
<a href="#">Method testFuzzyOffA</a>	604
<a href="#">Method testFuzzyOffB</a>	604
<a href="#">Method testFuzzyOnA</a>	604
<a href="#">Method testFuzzyOnB</a>	605
<a href="#">Method testFuzzyOne</a>	605
<a href="#">Method testFuzzyTrueA</a>	605
<a href="#">Method testFuzzyTrueB</a>	605
<a href="#">Method testFuzzyTrueC</a>	605
<a href="#">Method testFuzzyYesA</a>	606
<a href="#">Method testFuzzyYesB</a>	606
<a href="#">Method testFuzzyYesC</a>	606
<a href="#">Method testFuzzyYesD</a>	606
<a href="#">Method testFuzzyYesE</a>	607
<a href="#">Method testFuzzyZero</a>	607
<a href="#">Method testUnexpectedGreatLiterature</a>	607
<a href="#">Method testUnexpectedLargeNumber</a>	607
<a href="#">Method testUnexpectedNegative</a>	608
<a href="#">Method testUnexpectedNull</a>	608
<a href="#">Method testUnexpectedSpaces</a>	608
<a href="#"><b>Class tests_phpDocumentorTParserGetInlineTagsTests</b></a>	608
<a href="#">Method main</a>	609
<a href="#">Method setUp</a>	609
<a href="#">Method tearDown</a>	609
<a href="#">Method testShowCorrectBehaviorInlineExampleWhenParsePrivateOff</a>	610
<a href="#">Method testShowCorrectBehaviorInlineExampleWhenParsePrivateOn</a>	610
<a href="#">Method testShowCorrectBehaviorInlineInternalWhenParsePrivateOff</a>	610
<a href="#">Method testShowCorrectBehaviorInlineInternalWhenParsePrivateOn</a>	611
<a href="#">Method testShowCorrectBehaviorInlineLinkWhenParsePrivateOff</a>	611
<a href="#">Method testShowCorrectBehaviorInlineLinkWhenParsePrivateOn</a>	611
<a href="#">Method testShowCorrectBehaviorInlineSourceWhenParsePrivateOff</a>	611
<a href="#">Method testShowCorrectBehaviorInlineSourceWhenParsePrivateOn</a>	612
<a href="#">Method testShowCorrectBehaviorInlineTutorialWhenParsePrivateOff</a>	612
<a href="#">Method testShowCorrectBehaviorInlineTutorialWhenParsePrivateOn</a>	612
<a href="#"><b>Class bug_556894_base</b></a>	613
<a href="#">Var \$test</a>	613
<a href="#">Method test</a>	613
<a href="#"><b>Class bug_556894_sub1</b></a>	613
<a href="#"><b>Class bug_556894_sub2</b></a>	614
<a href="#"><b>Package XML_Beautifier_Procedural Elements</b></a>	616
<a href="#">Plain.php</a>	616
<a href="#"><b>Package XML_Beautifier Classes</b></a>	617
<a href="#">Class PHPDoc_XML_Beautifier_Renderer_Plain</a>	617
<a href="#">Method serialize</a>	617
<a href="#"><b>Package phpDocumentor_Procedural Elements</b></a>	619
<a href="#">actions.php</a>	619

<a href="#">builder.php</a>	621
<a href="#">config.php</a>	623
<a href="#">utilities.php</a>	625
<a href="#">Function getDir</a>	625
<a href="#">Function htmlArraySelect</a>	625
<a href="#">Function recurseDir</a>	626
<a href="#">Function showImage</a>	626
<a href="#">Function switchDirTree</a>	626
<a href="#">Function vdump_par</a>	626
<a href="#">top.php</a>	628
<a href="#">new_phpdoc.php</a>	629
<a href="#">phpdoc.php</a>	630
<a href="#">Classes.inc</a>	631
<a href="#">clone.inc.php</a>	632
<a href="#">Function phpDocumentor_clone</a>	632
<a href="#">clone5.inc.php</a>	634
<a href="#">common.inc.php</a>	635
<a href="#">Function debug</a>	635
<a href="#">Function fancy_debug</a>	636
<a href="#">Define PATH_DELIMITER</a>	636
<a href="#">Function phpDocumentor_ConfigFileList</a>	636
<a href="#">Function phpDocumentor_get_class</a>	636
<a href="#">Function phpDocumentor_parse_ini_file</a>	637
<a href="#">Define PHPDOCUMENTOR_VER</a>	637
<a href="#">Define PHPDOCUMENTOR_WEBSITE</a>	637
<a href="#">Define PHPDOCUMENTOR_WINDOWS</a>	637
<a href="#">Define SMART_PATH_DELIMITER</a>	637
<a href="#">Define tokenizer_ext</a>	637
<a href="#">Define IN_PHP5</a>	637
<a href="#">EventStack.inc</a>	638
<a href="#">IntermediateParser.inc</a>	639
<a href="#">Io.inc</a>	640
<a href="#">Function get_include_path</a>	640
<a href="#">Function loinc_mystrucsort</a>	640
<a href="#">Function loinc_sortfiles</a>	641
<a href="#">Function setup_dirs</a>	641
<a href="#">Function set_dir</a>	641
<a href="#">ParserDescCleanup.inc</a>	643
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_B</a>	643
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_BR</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_CODE</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_DOUBLECR</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_ESCAPE</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_ESCAPE_CODE</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_ESCAPE_PRE</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_I</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_KBD</a>	644
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_LIST</a>	645
<a href="#">Define PHPDOCUMENTOR_PDP_EVENT_P</a>	645

<u>Define PHPDOCUMENTOR_PDP_EVENT_PRE</u>	645
<u>Define PHPDOCUMENTOR_PDP_EVENT_SAMP</u>	645
<u>Define PHPDOCUMENTOR_PDP_EVENT_SIMLIST</u>	645
<u>Define PHPDOCUMENTOR_PDP_EVENT_VAR</u>	645
<u>Define PHPDOCUMENTOR_PDP_STATE_B</u>	645
<u>Define PHPDOCUMENTOR_PDP_STATE_BR</u>	645
<u>Define PHPDOCUMENTOR_PDP_STATE_CODE</u>	645
<u>Define PHPDOCUMENTOR_PDP_STATE_DOUBLECR</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_ESCAPE</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_ESCAPE_CODE</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_ESCAPE_PRE</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_I</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_KBD</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_LIST</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_P</u>	646
<u>Define PHPDOCUMENTOR_PDP_STATE_PRE</u>	647
<u>Define PHPDOCUMENTOR_PDP_STATE_SAMP</u>	647
<u>Define PHPDOCUMENTOR_PDP_STATE_SIMLIST</u>	647
<u>Define PHPDOCUMENTOR_PDP_STATE_VAR</u>	647
<u>phpdoc.inc</u>	648
<u>ProceduralPages.inc</u>	649
<u>Publisher.inc</u>	650
<u>Setup.inc.php</u>	651
<u>Global Variable \$interface</u>	651
<u>Global Variable \$phpDocumentor_DefaultCategoryName</u>	651
<u>Global Variable \$phpDocumentor_DefaultPackageName</u>	652
<u>Global Variable \$phpDocumentor_setting</u>	652
<u>Function checkForBugCondition</u>	652
<u>Function decideOnOrOff</u>	652
<u>Function phpDocumentor_out</u>	654
<u>HighlightParserTests.php</u>	655
<u>IntermediateParserTests.php</u>	656
<u>ParserClassTests.php</u>	657
<u>ParserPageTests.php</u>	658
<u>phpDocumentorSetupTests.php</u>	659
<u>phpDocumentorTParserTests.php</u>	660
<u>Define PHPUnit_MAIN_METHOD</u>	660
<u>Package phpDocumentor Classes</u>	661
<u>Class bug_772441</u>	661
<u>Class Classes</u>	661
<u>Method addClass</u>	662
<u>Method addConst</u>	662
<u>Method addMethod</u>	662
<u>Method addPackageToFile</u>	663
<u>Method addVar</u>	663
<u>Method getClass</u>	663
<u>Method getClassByPackage</u>	664
<u>Method getClassesInPath</u>	664
<u>Method getConflicts</u>	664

<a href="#">Method getDefiniteChildren</a>	664
<a href="#">Method getParentClass</a>	665
<a href="#">Method getRoots</a>	665
<a href="#">Method Inherit</a>	665
<a href="#">Method nextFile</a>	666
<a href="#">Method processChild</a>	666
<a href="#">Method setClassParent</a>	667
<a href="#"><b>Class EventStack</b></a>	<b>667</b>
<a href="#">Var \$num</a>	668
<a href="#">Var \$stack</a>	668
<a href="#">Method getEvent</a>	668
<a href="#">Method popEvent</a>	668
<a href="#">Method pushEvent</a>	668
<a href="#"><b>Class Io</b></a>	<b>668</b>
<a href="#">Var \$ignore</a>	669
<a href="#">Var \$phpDocOptions</a>	669
<a href="#">Var \$valid_booleans</a>	669
<a href="#">Constructor Io</a>	669
<a href="#">Method checkIgnore</a>	670
<a href="#">Method dirList</a>	670
<a href="#">Method displayHelpMsg</a>	670
<a href="#">Method getAllFiles</a>	670
<a href="#">Method getBase</a>	671
<a href="#">Method getDirTree</a>	671
<a href="#">Method getReadmeInstallChangelog</a>	672
<a href="#">Method getRegExableSearchString</a>	672
<a href="#">Method getTutorials</a>	672
<a href="#">Method isIncludeable</a>	672
<a href="#">Method parseArgv</a>	673
<a href="#">Method readPhpFile</a>	673
<a href="#">Method removeNonMatches</a>	673
<a href="#">Method _setupIgnore</a>	674
<a href="#"><b>Class phpDocumentor_IntermediateParser</b></a>	<b>674</b>
<a href="#">Var \$all_packages</a>	675
<a href="#">Var \$classes</a>	675
<a href="#">Var \$converters</a>	675
<a href="#">Var \$cur_class</a>	676
<a href="#">Var \$data</a>	676
<a href="#">Var \$db_template</a>	676
<a href="#">Var \$event_handlers</a>	676
<a href="#">Var \$last</a>	677
<a href="#">Var \$lasttype</a>	677
<a href="#">Var \$packagecategories</a>	677
<a href="#">Var \$packageoutput</a>	678
<a href="#">Var \$package_pages</a>	678
<a href="#">Var \$package_parents</a>	678
<a href="#">Var \$pages</a>	679
<a href="#">Var \$parsePrivate</a>	679
<a href="#">Var \$privatepages</a>	679

<a href="#">Var \$private_class</a>	679
<a href="#">Var \$proceduralpages</a>	680
<a href="#">Var \$quietMode</a>	680
<a href="#">Var \$ric</a>	680
<a href="#">Var \$targetDir</a>	681
<a href="#">Var \$templateBase</a>	681
<a href="#">Var \$title</a>	681
<a href="#">Var \$type</a>	681
<a href="#">Var \$undocumentedElementWarnings</a>	682
<a href="#">Var \$uses</a>	682
<a href="#">Constructor <code>phpDocumentor_IntermediateParser</code></a>	682
<a href="#">Method addConverter</a>	682
<a href="#">Method addElementToPage</a>	683
<a href="#">Method addPackageParent</a>	683
<a href="#">Method addPage</a>	683
<a href="#">Method addPageIfNecessary</a>	684
<a href="#">Method addPrivatePage</a>	684
<a href="#">Method addUses</a>	684
<a href="#">Method ClasselementCmp</a>	685
<a href="#">Method Convert</a>	685
<a href="#">Method elementCmp</a>	686
<a href="#">Method handleClass</a>	686
<a href="#">Method handleConst</a>	686
<a href="#">Method handleDefine</a>	687
<a href="#">Method handleDocBlock</a>	687
<a href="#">Method HandleEvent</a>	688
<a href="#">Method handleFunction</a>	688
<a href="#">Method handleGlobal</a>	689
<a href="#">Method handleInclude</a>	689
<a href="#">Method handleMethod</a>	689
<a href="#">Method handlePackagePage</a>	690
<a href="#">Method handlePage</a>	690
<a href="#">Method handleTutorial</a>	690
<a href="#">Method handleVar</a>	691
<a href="#">Method Output</a>	691
<a href="#">Method parsePackagePage</a>	692
<a href="#">Method setParsePrivate</a>	692
<a href="#">Method setQuietMode</a>	692
<a href="#">Method setTargetDir</a>	692
<a href="#">Method setTemplateBase</a>	693
<a href="#">Method setUndocumentedElementWarningsMode</a>	693
<a href="#">Method _guessPackage</a>	693
<a href="#">Class <code>phpDocumentor_setup</code></a>	694
<a href="#">Var \$dirs</a>	694
<a href="#">Var \$files</a>	694
<a href="#">Var \$hidden</a>	695
<a href="#">Var \$ignoresymlinks</a>	695
<a href="#">Var \$ignore_files</a>	695
<a href="#">Var \$packages</a>	695

<a href="#">Var \$parse</a>	695
<a href="#">Var \$render</a>	695
<a href="#">Var \$setup</a>	696
<a href="#">Constructor <code>phpDocumentor::setup</code></a>	696
<a href="#">Method <code>checkIgnoreTag</code></a>	696
<a href="#">Method <code>createDocs</code></a>	696
<a href="#">Method <code>parseHiddenFiles</code></a>	696
<a href="#">Method <code>parseIni</code></a>	696
<a href="#">Method <code>readCommandLineSettings</code></a>	697
<a href="#">Method <code>readConfigFile</code></a>	697
<a href="#">Method <code>setDirectoriesToParse</code></a>	697
<a href="#">Method <code>setFilesToParse</code></a>	697
<a href="#">Method <code>setIgnore</code></a>	697
<a href="#">Method <code>setJavadocDesc</code></a>	697
<a href="#">Method <code>setMemoryLimit</code></a>	697
<a href="#">Method <code>setPackageOutput</code></a>	698
<a href="#">Method <code>setParsePrivate</code></a>	698
<a href="#">Method <code>setQuietMode</code></a>	698
<a href="#">Method <code>setTargetDir</code></a>	698
<a href="#">Method <code>setTemplateBase</code></a>	698
<a href="#">Method <code>setTitle</code></a>	698
<a href="#">Method <code>setUndocumentedElementWarnings</code></a>	698
<a href="#"><b>Class ProceduralPages</b></a>	699
<a href="#">Var \$curfile</a>	699
<a href="#">Var \$defineconflicts</a>	699
<a href="#">Var \$definesbyfile</a>	700
<a href="#">Var \$definesbynamefile</a>	700
<a href="#">Var \$functionconflicts</a>	700
<a href="#">Var \$functionsbyfile</a>	701
<a href="#">Var \$functionsbynamefile</a>	701
<a href="#">Var \$globalconflicts</a>	701
<a href="#">Var \$globalsbyfile</a>	702
<a href="#">Var \$globalsbynamefile</a>	702
<a href="#">Var \$ignorepages</a>	702
<a href="#">Var \$includesbyfile</a>	702
<a href="#">Var \$pageclasspackages</a>	703
<a href="#">Var \$pageconflicts</a>	703
<a href="#">Var \$pagepackages</a>	703
<a href="#">Var \$pages</a>	704
<a href="#">Var \$pathpages</a>	704
<a href="#">Method <code>addClassPackageToFile</code></a>	704
<a href="#">Method <code>addDefine</code></a>	704
<a href="#">Method <code>addFunction</code></a>	705
<a href="#">Method <code>addGlobal</code></a>	705
<a href="#">Method <code>addInclude</code></a>	705
<a href="#">Method <code>addPage</code></a>	705
<a href="#">Method <code>addPagePackage</code></a>	705
<a href="#">Method <code>getPathInfo</code></a>	706
<a href="#">Method <code>getRealPath</code></a>	706

<a href="#">Method ignorePage</a>	706
<a href="#">Method pathMatchesParsedFile</a>	707
<a href="#">Method replaceElement</a>	707
<a href="#">Method setName</a>	707
<a href="#">Method setParseBase</a>	707
<a href="#">Method setupPagePackages</a>	707
<a href="#">Method setupPages</a>	708
<a href="#"><u>Class Publisher</u></a>	708
<a href="#">Var \$popEvent</a>	709
<a href="#">Var \$pushEvent</a>	709
<a href="#">Var \$subscriber</a>	709
<a href="#">Var \$tokens</a>	709
<a href="#">Method publishEvent</a>	709
<a href="#">Method subscribe</a>	709
<a href="#"><u>Class tests_HighlightParserTests</u></a>	710
<a href="#">Method main</a>	710
<a href="#">Method suite</a>	710
<a href="#"><u>Class tests_IntermediateParserTests</u></a>	711
<a href="#">Method main</a>	711
<a href="#">Method suite</a>	711
<a href="#"><u>Class tests_ParserClassTests</u></a>	711
<a href="#">Method main</a>	711
<a href="#">Method suite</a>	712
<a href="#"><u>Class tests_ParserPageTests</u></a>	712
<a href="#">Method main</a>	712
<a href="#">Method suite</a>	712
<a href="#"><u>Class tests_phpDocumentorSetupTests</u></a>	712
<a href="#">Method main</a>	713
<a href="#">Method suite</a>	713
<a href="#"><u>Class tests_phpDocumentorTParserTests</u></a>	713
<a href="#">Method main</a>	713
<a href="#">Method suite</a>	713
<a href="#"><u>DescHTML.inc</u></a>	715
<a href="#"><u>Class parserB</u></a>	716
<a href="#">Method Convert</a>	716
<a href="#"><u>Class parserBr</u></a>	717
<a href="#">Method Convert</a>	717
<a href="#"><u>Class parserCode</u></a>	718
<a href="#">Method Convert</a>	718
<a href="#"><u>Class parserDescVar</u></a>	718
<a href="#">Method Convert</a>	719
<a href="#"><u>Class parserI</u></a>	719
<a href="#">Method Convert</a>	720
<a href="#"><u>Class parserKbd</u></a>	720
<a href="#">Method Convert</a>	721
<a href="#"><u>Class parserList</u></a>	721
<a href="#">Var \$items</a>	721
<a href="#">Var \$numbered</a>	722
<a href="#">Constructor parserList</a>	722

<a href="#">Method addItem</a>	722
<a href="#">Method addList</a>	722
<a href="#">Method Convert</a>	722
<a href="#">Class parserPre</a>	723
<a href="#">Method Convert</a>	723
<a href="#">Class parserSamp</a>	724
<a href="#">Method Convert</a>	724
<a href="#">DocBlockTags.inc</a>	725
<a href="#">Class parserAccessTag</a>	726
<a href="#">Var \$isValid</a>	726
<a href="#">Var \$Keyword</a>	726
<a href="#">Constructor parserAccessTag</a>	726
<a href="#">Method Convert</a>	727
<a href="#">Method getString</a>	727
<a href="#">Class parserExampleTag</a>	727
<a href="#">Var \$Keyword</a>	728
<a href="#">Constructor parserExampleTag</a>	728
<a href="#">Method ConvertSource</a>	728
<a href="#">Method getSourceLink</a>	729
<a href="#">Class parserFileSourceTag</a>	729
<a href="#">Var \$Keyword</a>	729
<a href="#">Var \$path</a>	730
<a href="#">Var \$source</a>	730
<a href="#">Constructor parserFileSourceTag</a>	730
<a href="#">Method Convert</a>	730
<a href="#">Method ConvertSource</a>	730
<a href="#">Method getSourceLink</a>	731
<a href="#">Method writeSource</a>	731
<a href="#">Class parserLicenseTag</a>	732
<a href="#">Var \$Keyword</a>	732
<a href="#">Constructor parserLicenseTag</a>	732
<a href="#">Class parserLinkTag</a>	732
<a href="#">Var \$Keyword</a>	733
<a href="#">Constructor parserLinkTag</a>	733
<a href="#">Class parserMethodTag</a>	733
<a href="#">Var \$Keyword</a>	734
<a href="#">Var \$ReturnType</a>	734
<a href="#">Class parserNameTag</a>	734
<a href="#">Var \$Keyword</a>	735
<a href="#">Constructor parserNameTag</a>	735
<a href="#">Method Convert</a>	735
<a href="#">Class parserParamTag</a>	735
<a href="#">Var \$Keyword</a>	736
<a href="#">Class parserPropertyReadTag</a>	736
<a href="#">Var \$Keyword</a>	736
<a href="#">Class parserPropertyTag</a>	737
<a href="#">Var \$Keyword</a>	737
<a href="#">Var \$ReturnType</a>	737
<a href="#">Constructor parserPropertyTag</a>	737

<a href="#">Class parserPropertyWriteTag</a>	738
<a href="#">Var \$Keyword</a>	738
<a href="#">Class parserReturnTag</a>	738
<a href="#">Var \$converted returnType</a>	739
<a href="#">Var \$Keyword</a>	739
<a href="#">Var \$returnType</a>	739
<a href="#">Constructor parserReturnTag</a>	740
<a href="#">Method Convert</a>	740
<a href="#">Class parserSeeTag</a>	740
<a href="#">Var \$Keyword</a>	741
<a href="#">Constructor parserSeeTag</a>	741
<a href="#">Method Convert</a>	741
<a href="#">Class parserStaticvarTag</a>	741
<a href="#">Var \$Keyword</a>	742
<a href="#">Class parserTag</a>	742
<a href="#">Var \$Keyword</a>	743
<a href="#">Var \$type</a>	743
<a href="#">Constructor parserTag</a>	743
<a href="#">Method Convert</a>	743
<a href="#">Method getString</a>	744
<a href="#">Method HandleEvent</a>	744
<a href="#">Class parserTutorialTag</a>	744
<a href="#">Var \$Keyword</a>	745
<a href="#">Method Convert</a>	745
<a href="#">Class parserUsedByTag</a>	746
<a href="#">Var \$Keyword</a>	746
<a href="#">Constructor parserUsedByTag</a>	746
<a href="#">Method Convert</a>	746
<a href="#">Class parserUsesTag</a>	747
<a href="#">Var \$Keyword</a>	747
<a href="#">Constructor parserUsesTag</a>	747
<a href="#">Method Convert</a>	748
<a href="#">Method getDescription</a>	748
<a href="#">Class parserVarTag</a>	748
<a href="#">Var \$Keyword</a>	749
<a href="#">Var \$returnType</a>	749
<a href="#">Errors.inc</a>	750
<a href="#">Global Variable \$phpDocumentor_errors</a>	750
<a href="#">Global Variable \$phpDocumentor_error_descrip</a>	754
<a href="#">Global Variable \$phpDocumentor_warning_descrip</a>	756
<a href="#">Function addError</a>	756
<a href="#">Function addErrorDie</a>	756
<a href="#">Function addWarning</a>	757
<a href="#">Define PDERROR ACCESS WRONG PARAM</a>	757
<a href="#">Define PDERROR BEAUTIFYING FAILED</a>	757
<a href="#">Define PDERROR CANNOT EXTEND SELF</a>	757
<a href="#">Define PDERROR CANT HAVE INLINE IN TAGNAME</a>	758
<a href="#">Define PDERROR CANT NEST IN B</a>	758
<a href="#">Define PDERROR CHILD TUTORIAL NOT FOUND</a>	758

Define PDERROR CLASS CONFLICT	758
Define PDERROR CLASS EXISTS	758
Define PDERROR CLASS NOT IN PACKAGE	758
Define PDERROR CLASS PARENT NOT FOUND	758
Define PDERROR CONVERTER NOT FOUND	758
Define PDERROR CONVERTER OVR GFCT	758
Define PDERROR DANGEROUS PHP BUG EXISTS	759
Define PDERROR DB TEMPLATE UNTERMINATED	759
Define PDERROR DOCBLOCK CONFLICT	759
Define PDERROR DOCBLOCK GOES CLASS	759
Define PDERROR DUMB USES	759
Define PDERROR ELEMENT IGNORED	759
Define PDERROR EMPTY EXAMPLE TITLE	759
Define PDERROR EXAMPLE NOT FOUND	759
Define PDERROR FUNCTION HAS NONAME	759
Define PDERROR GLOBAL NOT FOUND	760
Define PDERROR ID MUST BE INLINE	760
Define PDERROR IGNORE TAG IGNORED	760
Define PDERROR ILLEGAL PACKAGENAME	760
Define PDERROR INHERITANCE CONFLICT	760
Define PDERROR INHERITDOC DONT WORK HERE	760
Define PDERROR INLINETAG IN SEE	760
Define PDERROR INTERNAL NOT CLOSED	760
Define PDERROR INVALID VALUES	761
Define PDERROR LOOP RECURSION LIMIT REACHED	761
Define PDERROR MALFORMED GLOBAL TAG	761
Define PDERROR MALFORMED TAG	761
Define PDERROR MISSING PROPERTY TAG NAME	761
Define PDERROR MULTIPLE ACCESS TAGS	761
Define PDERROR MULTIPLE CATEGORY TAGS	761
Define PDERROR MULTIPLE GLOBAL TAGS	761
Define PDERROR MULTIPLE NAME TAGS	761
Define PDERROR MULTIPLE PACKAGE TAGS	762
Define PDERROR MULTIPLE PARENT	762
Define PDERROR MULTIPLE RETURN TAGS	762
Define PDERROR MULTIPLE SUBPACKAGE TAGS	762
Define PDERROR MULTIPLE VAR TAGS	762
Define PDERROR NAME ALIAS SAME AS TARGET	762
Define PDERROR NEED WHITESPACE	762
Define PDERROR NESTED INTERNAL	762
Define PDERROR NOTODO INCLUDE	762
Define PDERROR NO CONVERTERS	762
Define PDERROR NO CONVERTER HANDLER	763
Define PDERROR NO DOCBLOCK ID	763
Define PDERROR NO PACKAGE TAG	763
Define PDERROR NO PAGE LEVELDOCBLOCK	763
Define PDERROR OVERRIDDEN PACKAGE TAGS	763
Define PDERROR OVERRIDDEN SUBPACKAGE TAGS	763
Define PDERROR PACKAGECAT SET	763

<u>Define PDERROR PACKAGEOUTPUT DELETES PARENT FILE</u>	763
<u>Define PDERROR PARENT NOT FOUND</u>	763
<u>Define PDERROR PARSEPRIVATE</u>	764
<u>Define PDERROR PDFFUNCTION NO FUNC</u>	764
<u>Define PDERROR PDF METHOD DOESNT EXIST</u>	764
<u>Define PDERROR PDF TEMPVAR DOESNT EXIST</u>	764
<u>Define PDERROR PRIVATE ASSUMED</u>	764
<u>Define PDERROR SOURCECODE IGNORED</u>	764
<u>Define PDERROR SOURCE TAG FUNCTION NOT FOUND</u>	764
<u>Define PDERROR SOURCE TAG IGNORED</u>	764
<u>Define PDERROR TAG NOT HANDLED</u>	764
<u>Define PDERROR TEMPLATEDIR DOESNT EXIST</u>	764
<u>Define PDERROR TEXT OUTSIDE LI</u>	765
<u>Define PDERROR TUTORIAL IS OWN CHILD</u>	765
<u>Define PDERROR TUTORIAL IS OWN GRANDPA</u>	765
<u>Define PDERROR TUTORIAL NOT FOUND</u>	765
<u>Define PDERROR TUTORIAL SUBSECTION NOT FOUND</u>	765
<u>Define PDERROR UL IN UL</u>	765
<u>Define PDERROR UNCLOSED TAG</u>	765
<u>Define PDERROR UNDOCUMENTED ELEMENT</u>	765
<u>Define PDERROR UNKNOWN COMMANDLINE</u>	765
<u>Define PDERROR UNKNOWN TAG</u>	766
<u>Define PDERROR UNMATCHED LIST TAG</u>	766
<u>Define PDERROR UNMATCHED TUTORIAL TAG</u>	766
<u>Define PDERROR UNTERMINATED ATTRIB</u>	766
<u>Define PDERROR UNTERMINATED ENTITY</u>	766
<u>Define PDERROR UNTERMINATED INLINE TAG</u>	766
<u>Class ErrorTracker</u>	766
<u>Var \$curfile</u>	767
<u>Var \$errors</u>	767
<u>Var \$lasterror</u>	767
<u>Var \$lastwarning</u>	767
<u>Var \$linenum</u>	767
<u>Var \$warnings</u>	767
<u>Method addError</u>	767
<u>Method addErrorReturn</u>	768
<u>Method addWarning</u>	768
<u>Method handleEvent</u>	768
<u>Method returnErrors</u>	769
<u>Method returnLastError</u>	769
<u>Method returnLastWarning</u>	769
<u>Method returnWarnings</u>	769
<u>Class RecordError</u>	769
<u>Var \$type</u>	770
<u>Method output</u>	770
<u>Class RecordWarning</u>	770
<u>Var \$data</u>	770
<u>Var \$file</u>	771
<u>Var \$linenum</u>	771

<a href="#">Var \$num</a>	771
<a href="#">Var \$type</a>	771
<a href="#">Constructor RecordWarning</a>	771
<a href="#">Method output</a>	772
<a href="#">InlineTags.inc</a>	773
<a href="#">Class parserExampleInlineTag</a>	773
<a href="#">Constructor parserExampleInlineTag</a>	774
<a href="#">Method arrayConvert</a>	774
<a href="#">Method getProgramListing</a>	775
<a href="#">Method setSource</a>	775
<a href="#">Class parserIdInlineTag</a>	775
<a href="#">Var \$category</a>	776
<a href="#">Var \$id</a>	776
<a href="#">Var \$inlinetype</a>	776
<a href="#">Var \$package</a>	776
<a href="#">Var \$subpackage</a>	776
<a href="#">Var \$tutorial</a>	776
<a href="#">Constructor parserIdInlineTag</a>	776
<a href="#">Method Convert</a>	777
<a href="#">Class parserInheritdocInlineTag</a>	777
<a href="#">Var \$inlinetype</a>	778
<a href="#">Constructor parserInheritdocInlineTag</a>	778
<a href="#">Method Convert</a>	778
<a href="#">Class parserInlineTag</a>	778
<a href="#">Var \$inlinetype</a>	779
<a href="#">Var \$type</a>	779
<a href="#">Constructor parserInlineTag</a>	779
<a href="#">Method getString</a>	779
<a href="#">Method Strlen</a>	779
<a href="#">Class parserLinkInlineTag</a>	780
<a href="#">Var \$linktext</a>	780
<a href="#">Constructor parserLinkInlineTag</a>	780
<a href="#">Method Convert</a>	781
<a href="#">Method ConvertPart</a>	781
<a href="#">Class parserSourceInlineTag</a>	781
<a href="#">Var \$end</a>	782
<a href="#">Var \$inlinetype</a>	782
<a href="#">Var \$source</a>	782
<a href="#">Var \$start</a>	782
<a href="#">Constructor parserSourceInlineTag</a>	783
<a href="#">Method arrayConvert</a>	783
<a href="#">Method Convert</a>	783
<a href="#">Method getString</a>	784
<a href="#">Method setSource</a>	784
<a href="#">Method stringConvert</a>	784
<a href="#">Method Strlen</a>	784
<a href="#">Class parserTocInlineTag</a>	785
<a href="#">Var \$inlinetype</a>	785
<a href="#">Constructor parserTocInlineTag</a>	785

<a href="#">Method Convert</a>	785
<a href="#">Method setPath</a>	786
<a href="#">Method setTOC</a>	786
<a href="#">Class parserTutorialInlineTag</a>	786
<a href="#">Constructor parserTutorialInlineTag</a>	787
<a href="#">Method Convert</a>	787
<a href="#">LinkClasses.inc</a>	788
<a href="#">Class abstractLink</a>	789
<a href="#">Var \$category</a>	789
<a href="#">Var \$fileAlias</a>	789
<a href="#">Var \$name</a>	789
<a href="#">Var \$package</a>	789
<a href="#">Var \$path</a>	789
<a href="#">Var \$subpackage</a>	789
<a href="#">Var \$type</a>	789
<a href="#">Method addLink</a>	790
<a href="#">Class classLink</a>	790
<a href="#">Var \$type</a>	790
<a href="#">Class constLink</a>	791
<a href="#">Var \$type</a>	791
<a href="#">Class defineLink</a>	791
<a href="#">Var \$type</a>	791
<a href="#">Class functionLink</a>	792
<a href="#">Var \$type</a>	792
<a href="#">Class globalLink</a>	792
<a href="#">Var \$type</a>	793
<a href="#">Class methodLink</a>	793
<a href="#">Var \$class</a>	793
<a href="#">Var \$type</a>	793
<a href="#">Method addLink</a>	793
<a href="#">Class pageLink</a>	794
<a href="#">Var \$type</a>	794
<a href="#">Class tutorialLink</a>	794
<a href="#">Var \$section</a>	795
<a href="#">Var \$title</a>	795
<a href="#">Var \$type</a>	795
<a href="#">Method addLink</a>	795
<a href="#">Class varLink</a>	795
<a href="#">Var \$type</a>	796
<a href="#">ParserData.inc</a>	797
<a href="#">Class parserBase</a>	797
<a href="#">Var \$type</a>	798
<a href="#">Var \$value</a>	798
<a href="#">Method getType</a>	798
<a href="#">Method getValue</a>	798
<a href="#">Method setValue</a>	798
<a href="#">Class parserData</a>	799
<a href="#">Var \$classelements</a>	799
<a href="#">Var \$clean</a>	799

<a href="#">Var \$docblock</a>	800
<a href="#">Var \$elements</a>	800
<a href="#">Var \$links</a>	800
<a href="#">Var \$parent</a>	800
<a href="#">Var \$privateclasselements</a>	800
<a href="#">Var \$privateelements</a>	800
<a href="#">Var \$tutorial</a>	801
<a href="#">Var \$type</a>	801
<a href="#">Method addElement</a>	801
<a href="#">Method addLink</a>	801
<a href="#">Method addTutorial</a>	801
<a href="#">Method explicitDocBlock</a>	802
<a href="#">Method getClasses</a>	802
<a href="#">Method getLink</a>	802
<a href="#">Method getName</a>	802
<a href="#">Method getTutorial</a>	802
<a href="#">Method hasClasses</a>	802
<a href="#">Method hasExplicitDocBlock</a>	802
<a href="#">Method hasInterfaces</a>	802
<a href="#">Method isClean</a>	802
<a href="#">Method setDocBlock</a>	803
<a href="#">Method setParent</a>	803
<a href="#"><b>Class parserPage</b></a>	803
<a href="#">Var \$category</a>	804
<a href="#">Var \$file</a>	804
<a href="#">Var \$id</a>	804
<a href="#">Var \$modDate</a>	804
<a href="#">Var \$name</a>	804
<a href="#">Var \$origName</a>	804
<a href="#">Var \$package</a>	804
<a href="#">Var \$packageOutput</a>	804
<a href="#">Var \$parserVersion</a>	805
<a href="#">Var \$path</a>	805
<a href="#">Var \$source</a>	805
<a href="#">Var \$sourceLocation</a>	805
<a href="#">Var \$subpackage</a>	805
<a href="#">Var \$type</a>	805
<a href="#"><b>Constructor parserPage</b></a>	806
<a href="#">Method getFile</a>	806
<a href="#">Method getName</a>	806
<a href="#">Method getPackageOutput</a>	806
<a href="#">Method getParseData</a>	806
<a href="#">Method getPath</a>	806
<a href="#">Method getSourceLocation</a>	806
<a href="#">Method getType</a>	807
<a href="#">Method setFile</a>	807
<a href="#">Method setName</a>	807
<a href="#">Method setPackageOutput</a>	807
<a href="#">Method setPath</a>	808

<a href="#">Method setSource</a>	808
<a href="#">Method setSourceLocation</a>	808
<a href="#">Class parserStringWithInlineTags</a>	809
<a href="#">Var \$type</a>	809
<a href="#">Var \$value</a>	809
<a href="#">Method add</a>	809
<a href="#">Method Convert</a>	809
<a href="#">Method getString</a>	810
<a href="#">Method hasInlineTag</a>	810
<a href="#">Method setSource</a>	810
<a href="#">Method trimmedStrlen</a>	811
<a href="#">ParserDocBlock.inc</a>	812
<a href="#">Class parserDesc</a>	812
<a href="#">Var \$type</a>	813
<a href="#">Method add</a>	813
<a href="#">Method hasInheritDoc</a>	813
<a href="#">Method hasSource</a>	813
<a href="#">Method replaceInheritDoc</a>	813
<a href="#">Class parserDocBlock</a>	814
<a href="#">Var \$category</a>	814
<a href="#">Var \$desc</a>	814
<a href="#">Var \$endlinenumber</a>	814
<a href="#">Var \$explicitcategory</a>	814
<a href="#">Var \$explicitpackage</a>	815
<a href="#">Var \$funcglobals</a>	815
<a href="#">Var \$hasaccess</a>	815
<a href="#">Var \$hasname</a>	815
<a href="#">Var \$linenumber</a>	815
<a href="#">Var \$package</a>	815
<a href="#">Var \$packagedescrip</a>	816
<a href="#">Var \$params</a>	816
<a href="#">Var \$processed_desc</a>	816
<a href="#">Var \$processed_sdesc</a>	816
<a href="#">Var \$properties</a>	816
<a href="#">Var \$return</a>	816
<a href="#">Var \$sdesc</a>	817
<a href="#">Var \$statics</a>	817
<a href="#">Var \$subpackage</a>	817
<a href="#">Var \$subpackagedescrip</a>	817
<a href="#">Var \$tags</a>	817
<a href="#">Var \$unknown_tags</a>	817
<a href="#">Var \$var</a>	817
<a href="#">Constructor parserDocBlock</a>	817
<a href="#">Method addAccess</a>	818
<a href="#">Method addExample</a>	818
<a href="#">Method addFileSource</a>	818
<a href="#">Method addFuncGlobal</a>	818
<a href="#">Method addKeyword</a>	819
<a href="#">Method addLink</a>	819

<a href="#">Method addName</a>	819
<a href="#">Method addPackage</a>	819
<a href="#">Method addParam</a>	820
<a href="#">Method addProperty</a>	820
<a href="#">Method addReturn</a>	820
<a href="#">Method addSee</a>	820
<a href="#">Method addStaticVar</a>	821
<a href="#">Method addTag</a>	821
<a href="#">Method addUnknownTag</a>	821
<a href="#">Method addUses</a>	821
<a href="#">Method addVar</a>	822
<a href="#">Method canSource</a>	822
<a href="#">Method cantSource</a>	822
<a href="#">Method changeGlobal</a>	822
<a href="#">Method changeParam</a>	822
<a href="#">Method changeStatic</a>	823
<a href="#">Method getDesc</a>	823
<a href="#">Method getEndLineNumber</a>	823
<a href="#">Method getExplicitCategory</a>	823
<a href="#">Method getExplicitPackage</a>	823
<a href="#">Method getKeyword</a>	823
<a href="#">Method getLineNumber</a>	823
<a href="#">Method getSDesc</a>	823
<a href="#">Method getType</a>	823
<a href="#">Method hasInheritDoc</a>	823
<a href="#">Method listParams</a>	824
<a href="#">Method listProperties</a>	824
<a href="#">Method listTags</a>	824
<a href="#">Method overridePackage</a>	824
<a href="#">Method postProcess</a>	824
<a href="#">Method replaceInheritDoc</a>	824
<a href="#">Method resetParams</a>	825
<a href="#">Method setDesc</a>	825
<a href="#">Method setEndLineNumber</a>	825
<a href="#">Method setExplicitCategory</a>	825
<a href="#">Method setExplicitPackage</a>	825
<a href="#">Method setLineNumber</a>	825
<a href="#">Method setShortDesc</a>	826
<a href="#">Method setSource</a>	826
<a href="#">Method updateGlobals</a>	826
<a href="#">Method updateModifiers</a>	827
<a href="#">Method updateParams</a>	827
<a href="#">Method updateStatics</a>	827
<a href="#">ParserElements.inc</a>	828
<a href="#">Class parserClass</a>	828
<a href="#">Var \$curfile</a>	829
<a href="#">Var \$extends</a>	829
<a href="#">Var \$ignore</a>	829
<a href="#">Var \$parent</a>	829

<a href="#">Var \$sourceLocation</a>	830
<a href="#">Var \$tutorial</a>	830
<a href="#">Var \$type</a>	830
<a href="#">Var \$implements</a>	830
<a href="#">Method addImplements</a>	830
<a href="#">Method addTutorial</a>	831
<a href="#">Method getChildClassList</a>	831
<a href="#">Method getConflicts</a>	831
<a href="#">Method getConstNames</a>	831
<a href="#">Method getConsts</a>	832
<a href="#">Method getExtends</a>	832
<a href="#">Method getImplements</a>	832
<a href="#">Method getInheritedConsts</a>	832
<a href="#">Method getInheritedMethods</a>	832
<a href="#">Method getInheritedVars</a>	832
<a href="#">Method getLink</a>	833
<a href="#">Method getMethod</a>	833
<a href="#">Method getMethodNames</a>	833
<a href="#">Method getMethods</a>	833
<a href="#">Method getModifiers</a>	833
<a href="#">Method getParent</a>	833
<a href="#">Method getParentClassTree</a>	834
<a href="#">Method getSourceLocation</a>	834
<a href="#">Method getTutorial</a>	834
<a href="#">MethodgetVar</a>	835
<a href="#">MethodgetVarNames</a>	835
<a href="#">MethodgetVars</a>	835
<a href="#">MethodhasConst</a>	835
<a href="#">MethodhasMethod</a>	835
<a href="#">MethodhasVar</a>	835
<a href="#">MethodisInterface</a>	835
<a href="#">MethodsetAccessModifiers</a>	836
<a href="#">MethodsetExtends</a>	836
<a href="#">MethodsetInterface</a>	836
<a href="#">MethodsetModifiers</a>	836
<a href="#">MethodsetParent</a>	836
<a href="#">MethodsetParentNoClass</a>	837
<a href="#">MethodsetSourceLocation</a>	837
<a href="#">Class parserConst</a>	837
<a href="#">Var \$class</a>	837
<a href="#">Var \$type</a>	838
<a href="#">Constructor parserConst</a>	838
<a href="#">Method getClass</a>	838
<a href="#">Method getLink</a>	838
<a href="#">Class parserDefine</a>	839
<a href="#">Var \$type</a>	839
<a href="#">Method getConflicts</a>	839
<a href="#">Method getLink</a>	839
<a href="#">Class parserElement</a>	840

<u>Var \$conflicts</u>	840
<u>Var \$docblock</u>	840
<u>Var \$endlinenumber</u>	841
<u>Var \$file</u>	841
<u>Var \$linenumber</u>	841
<u>Var \$name</u>	841
<u>Var \$path</u>	841
<u>Method getEndLineNumber</u>	841
<u>Method getFile</u>	841
<u>Method getLineNumber</u>	841
<u>Method getName</u>	841
<u>Method getPackage</u>	841
<u>Method getPath</u>	841
<u>Method setDocBlock</u>	842
<u>Method setEndLineNumber</u>	842
<u>Method setFile</u>	842
<u>Method setLineNumber</u>	842
<u>Method setName</u>	842
<u>Method setPath</u>	842
<b>Class parserFunction</b>	843
<u>Var \$globals</u>	843
<u>Var \$params</u>	843
<u>Var \$returnsreference</u>	844
<u>Var \$source</u>	844
<u>Var \$statics</u>	844
<u>Var \$type</u>	844
<u>Method addGlobals</u>	844
<u>Method addParam</u>	845
<u>Method addSource</u>	845
<u>Method addStatics</u>	845
<u>Method getConflicts</u>	845
<u>Method getFunctionCall</u>	846
<u>Method getIntricateFunctionCall</u>	846
<u>Method getLink</u>	846
<u>MethodgetParam</u>	846
<u>MethodgetReturnsReference</u>	846
<u>MethodgetSource</u>	846
<u>MethodhasSource</u>	846
<u>MethodlistGlobals</u>	847
<u>MethodlistParams</u>	847
<u>MethodlistStatics</u>	847
<u>MethodsetReturnsReference</u>	847
<b>Class parserGlobal</b>	847
<u>Var \$datatype</u>	847
<u>Var \$type</u>	847
<u>Method getConflicts</u>	847
<u>Method getDataType</u>	848
<u>Method getLink</u>	848
<u>Method setDataType</u>	848

<a href="#">Class parserInclude</a>	849
<a href="#">Var \$type</a>	849
<a href="#">Class parserMethod</a>	849
<a href="#">Var \$class</a>	850
<a href="#">Var \$isConstructor</a>	850
<a href="#">Var \$isDestructor</a>	850
<a href="#">Var \$type</a>	850
<a href="#">Var \$modifiers</a>	851
<a href="#">Constructor parserMethod</a>	851
<a href="#">Method addParam</a>	851
<a href="#">Method getClass</a>	851
<a href="#">Method getFunctionCall</a>	851
<a href="#">Method getImplements</a>	851
<a href="#">Method getIntricateFunctionCall</a>	851
<a href="#">Method getLink</a>	852
<a href="#">Method getModifiers</a>	852
<a href="#">Method getOverrides</a>	852
<a href="#">Method getOverridingMethods</a>	852
<a href="#">Method getOverridingMethodsForClass</a>	852
<a href="#">Method setConstructor</a>	853
<a href="#">Method setDestructor</a>	853
<a href="#">Method setModifiers</a>	853
<a href="#">Class parserPackagePage</a>	853
<a href="#">Var \$package</a>	853
<a href="#">Var \$type</a>	853
<a href="#">Constructor parserPackagePage</a>	854
<a href="#">Method Convert</a>	854
<a href="#">Class parserTutorial</a>	854
<a href="#">Var \$children</a>	854
<a href="#">Var \$ini</a>	854
<a href="#">Var \$linked_element</a>	855
<a href="#">Var \$name</a>	855
<a href="#">Var \$next</a>	855
<a href="#">Var \$package</a>	855
<a href="#">Var \$parent</a>	855
<a href="#">Var \$path</a>	856
<a href="#">Var \$prev</a>	856
<a href="#">Var \$tutorial_type</a>	856
<a href="#">Var \$type</a>	856
<a href="#">Var \$xml</a>	856
<a href="#">Constructor parserTutorial</a>	857
<a href="#">Method Convert</a>	857
<a href="#">Method getLink</a>	857
<a href="#">Method getNext</a>	857
<a href="#">Method getParent</a>	857
<a href="#">Method getPrev</a>	858
<a href="#">Method getTitle</a>	858
<a href="#">Method isChildOf</a>	858
<a href="#">Method setNext</a>	858

<a href="#">Method setParent</a>	859
<a href="#">Method setPrev</a>	859
<a href="#">Class parserVar</a>	860
<a href="#">Var \$class</a>	860
<a href="#">Var \$type</a>	860
<a href="#">Var \$modifiers</a>	860
<a href="#">Constructor parserVar</a>	860
<a href="#">Method getClass</a>	860
<a href="#">Method getLink</a>	861
<a href="#">Method getModifiers</a>	861
<a href="#">Method getOverrides</a>	861
<a href="#">Method getOverridingVars</a>	861
<a href="#">Method getOverridingVarsForClass</a>	861
<a href="#">Method setModifiers</a>	861
<a href="#">Beautifier.php</a>	863
<a href="#">Tokenizer.php</a>	864
<a href="#">Define PHPDOC BEAUTIFIER CDATA</a>	864
<a href="#">Define PHPDOC XMLTOKEN EVENT ATTRIBUTE</a>	864
<a href="#">Define PHPDOC XMLTOKEN EVENT CDATA</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT CLOSETAG</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT COMMENT</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT DEF</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT DOUBLEQUOTE</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT ENTITY</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT IN CDATA</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT NOEVENTS</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT OPENTAG</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT PI</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT SINGLEQUOTE</a>	865
<a href="#">Define PHPDOC XMLTOKEN EVENT XML</a>	865
<a href="#">Define STATE XMLTOKEN ATTRIBUTE</a>	866
<a href="#">Define STATE XMLTOKEN CDATA</a>	866
<a href="#">Define STATE XMLTOKEN CLOSETAG</a>	866
<a href="#">Define STATE XMLTOKEN COMMENT</a>	866
<a href="#">Define STATE XMLTOKEN DEF</a>	866
<a href="#">Define STATE XMLTOKEN DOUBLEQUOTE</a>	866
<a href="#">Define STATE XMLTOKEN ENTITY</a>	866
<a href="#">Define STATE XMLTOKEN IN CDATA</a>	866
<a href="#">Define STATE XMLTOKEN NOEVENTS</a>	866
<a href="#">Define STATE XMLTOKEN OPENTAG</a>	866
<a href="#">Define STATE XMLTOKEN PI</a>	866
<a href="#">Define STATE XMLTOKEN SINGLEQUOTE</a>	866
<a href="#">Define STATE XMLTOKEN XML</a>	867
<a href="#">HighlightParser.inc</a>	868
<a href="#">Parser.inc</a>	870
<a href="#">Define PARSER EVENT ACCESS MODIFIER</a>	870
<a href="#">Define PARSER EVENT ARRAY</a>	870
<a href="#">Define PARSER EVENT CLASS</a>	870
<a href="#">Define PARSER EVENT CLASS CONSTANT</a>	871

<a href="#">Define PARSER EVENT CLASS MEMBER</a>	871
<a href="#">Define PARSER EVENT COMMENT</a>	871
<a href="#">Define PARSER EVENT COMMENTBLOCK</a>	871
<a href="#">Define PARSER EVENT DEFINE</a>	871
<a href="#">Define PARSER EVENT DEFINE GLOBAL</a>	871
<a href="#">Define PARSER EVENT DEFINE PARAMS</a>	871
<a href="#">Define PARSER EVENT DEFINE PARAMS PARENTHESIS</a>	871
<a href="#">Define PARSER EVENT DESC</a>	871
<a href="#">Define PARSER EVENT DOCBLOCK</a>	871
<a href="#">Define PARSER EVENT DOCBLOCK TEMPLATE</a>	871
<a href="#">Define PARSER EVENT DOCKEYWORD</a>	871
<a href="#">Define PARSER EVENT DOCKEYWORD EMAIL</a>	872
<a href="#">Define PARSER EVENT END DOCBLOCK TEMPLATE</a>	872
<a href="#">Define PARSER EVENT END STATEMENT</a>	872
<a href="#">Define PARSER EVENT EOFQUOTE</a>	872
<a href="#">Define PARSER EVENT ESCAPE</a>	872
<a href="#">Define PARSER EVENT FUNCTION</a>	872
<a href="#">Define PARSER EVENT FUNCTION PARAMS</a>	872
<a href="#">Define PARSER EVENT FUNCTION PARAM VAR</a>	872
<a href="#">Define PARSER EVENT FUNC GLOBAL</a>	872
<a href="#">Define PARSER EVENT GLOBAL VALUE</a>	872
<a href="#">Define PARSER EVENT IMPLEMENTS</a>	872
<a href="#">Define PARSER EVENT INCLUDE</a>	872
<a href="#">Define PARSER EVENT INCLUDE PARAMS</a>	872
<a href="#">Define PARSER EVENT INCLUDE PARAMS PARENTHESIS</a>	873
<a href="#">Define PARSER EVENT INLINE DOCKEYWORD</a>	873
<a href="#">Define PARSER EVENT LOGICBLOCK</a>	873
<a href="#">Define PARSER EVENT METHOD</a>	873
<a href="#">Define PARSER EVENT METHOD LOGICBLOCK</a>	873
<a href="#">Define PARSER EVENT NOEVENTS</a>	873
<a href="#">Define PARSER EVENT OUTPHP</a>	873
<a href="#">Define PARSER EVENT PHPCODE</a>	873
<a href="#">Define PARSER EVENT QUOTE</a>	873
<a href="#">Define PARSER EVENT QUOTE VAR</a>	873
<a href="#">Define PARSER EVENT SINGLEQUOTE</a>	873
<a href="#">Define PARSER EVENT STATIC VAR</a>	873
<a href="#">Define PARSER EVENT STATIC VAR VALUE</a>	874
<a href="#">Define PARSER EVENT TAGS</a>	874
<a href="#">Define PARSER EVENT VAR</a>	874
<a href="#">Define PARSER EVENT VAR ARRAY</a>	874
<a href="#">Define PARSER EVENT VAR ARRAY COMMENT</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT CLASS</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT CONST</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT DEFINE</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT DOCBLOCK</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT DOCBLOCK TEMPLATE</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT END DOCBLOCK TEMPLATE</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT END PAGE</a>	874
<a href="#">Define PHPDOCUMENTOR EVENT FUNCTION</a>	875

<a href="#">Define PHPDOCUMENTOR EVENT GLOBAL</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT INCLUDE</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT MESSAGE</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT NEWFILE</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT NEWLINENUM</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT NEWSTATE</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT PACKAGEPAGE</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT PAGE</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT README INSTALL CHANGELOG</a>	875
<a href="#">Define PHPDOCUMENTOR EVENT TUTORIAL</a>	876
<a href="#">Define PHPDOCUMENTOR EVENT VAR</a>	876
<a href="#">Define STATE ACCESS MODIFIER</a>	876
<a href="#">Define STATE ARRAY</a>	876
<a href="#">Define STATE CLASS</a>	876
<a href="#">Define STATE CLASS CONSTANT</a>	876
<a href="#">Define STATE CLASS MEMBER</a>	876
<a href="#">Define STATE COMMENT</a>	876
<a href="#">Define STATE COMMENTBLOCK</a>	876
<a href="#">Define STATE DEFINE</a>	876
<a href="#">Define STATE DEFINE PARAMS</a>	876
<a href="#">Define STATE DEFINE PARAMS PARENTHESIS</a>	876
<a href="#">Define STATE DESC</a>	877
<a href="#">Define STATE DOCBLOCK</a>	877
<a href="#">Define STATE DOCBLOCK TEMPLATE</a>	877
<a href="#">Define STATE DOCKEYWORD</a>	877
<a href="#">Define STATE DOCKEYWORD EMAIL</a>	877
<a href="#">Define STATE END CLASS</a>	877
<a href="#">Define STATE END DOCBLOCK TEMPLATE</a>	877
<a href="#">Define STATE EOFQUOTE</a>	877
<a href="#">Define STATE ESCAPE</a>	877
<a href="#">Define STATE FUNCTION</a>	877
<a href="#">Define STATE FUNCTION PARAMS</a>	877
<a href="#">Define STATE FUNCTION PARAM VAR</a>	877
<a href="#">Define STATE FUNC GLOBAL</a>	878
<a href="#">Define STATE GLOBAL</a>	878
<a href="#">Define STATE GLOBAL VALUE</a>	878
<a href="#">Define STATE IMPLEMENTS</a>	878
<a href="#">Define STATE INCLUDE</a>	878
<a href="#">Define STATE INCLUDE PARAMS</a>	878
<a href="#">Define STATE INCLUDE PARAMS PARENTHESIS</a>	878
<a href="#">Define STATE INLINE DOCKEYWORD</a>	878
<a href="#">Define STATE LOGICBLOCK</a>	878
<a href="#">Define STATE METHOD</a>	878
<a href="#">Define STATE METHOD LOGICBLOCK</a>	878
<a href="#">Define STATE NOEVENTS</a>	878
<a href="#">Define STATE OUTPHP</a>	878
<a href="#">Define STATE PHPCODE</a>	879
<a href="#">Define STATE QUOTE</a>	879
<a href="#">Define STATE QUOTE VAR</a>	879

<u>Define STATE SINGLEQUOTE</u>	879
<u>Define STATE STATIC VAR</u>	879
<u>Define STATE STATIC VAR VALUE</u>	879
<u>Define STATE TAGS</u>	879
<u>Define STATE VAR</u>	879
<u>Define STATE VAR ARRAY</u>	879
<u>Define STATE VAR ARRAY COMMENT</u>	879
<u>Define T ABSTRACT</u>	879
<u>Define T CONST</u>	879
<u>Define T DOC COMMENT</u>	879
<u>Define T FINAL</u>	879
<u>Define T IMPLEMENTS</u>	879
<u>Define T INTERFACE</u>	879
<u>Define T ML COMMENT</u>	879
<u>Define T PRIVATE</u>	879
<u>Define T PROTECTED</u>	879
<u>Define T PUBLIC</u>	880
<u>phpDocumentorTParser.inc</u>	881
<u>TutorialHighlightParser.inc</u>	882
<u>Define STATE TUTORIAL ATTRIBUTE</u>	883
<u>Define STATE TUTORIAL CLOSETAG</u>	883
<u>Define STATE TUTORIAL COMMENT</u>	883
<u>Define STATE TUTORIAL DOUBLEQUOTE</u>	883
<u>Define STATE TUTORIAL ENTITY</u>	883
<u>Define STATE TUTORIAL ITAG</u>	883
<u>Define STATE TUTORIAL NOEVENTS</u>	883
<u>Define STATE TUTORIAL OPENTAG</u>	883
<u>Define STATE TUTORIAL SINGLEQUOTE</u>	883
<u>Define TUTORIAL EVENT ATTRIBUTE</u>	883
<u>Define TUTORIAL EVENT CLOSETAG</u>	883
<u>Define TUTORIAL EVENT COMMENT</u>	883
<u>Define TUTORIAL EVENT DOUBLEQUOTE</u>	884
<u>Define TUTORIAL EVENT ENTITY</u>	884
<u>Define TUTORIAL EVENT ITAG</u>	884
<u>Define TUTORIAL EVENT NOEVENTS</u>	884
<u>Define TUTORIAL EVENT OPENTAG</u>	884
<u>Define TUTORIAL EVENT SINGLEQUOTE</u>	884
<u>XMLpackagePageParser.inc</u>	885
<u>Define PHPDOCUMENTOR PDP EVENT ATTRIBUTES</u>	885
<u>Define PHPDOCUMENTOR PDP EVENT CDATA</u>	885
<u>Define PHPDOCUMENTOR PDP EVENT ENTITY</u>	886
<u>Define PHPDOCUMENTOR PDP EVENT PROGRAMLISTING</u>	886
<u>Define PHPDOCUMENTOR PDP EVENT TAG</u>	886
<u>Define PHPDOCUMENTOR PDP STATE ATTRIBUTES</u>	886
<u>Define PHPDOCUMENTOR PDP STATE CDATA</u>	886
<u>Define PHPDOCUMENTOR PDP STATE ENTITY</u>	886
<u>Define PHPDOCUMENTOR PDP STATE PROGRAMLISTING</u>	886
<u>Define PHPDOCUMENTOR PDP STATE TAG</u>	886
<u>Class Parser</u>	886

<u>Constructor Parser</u>	887
<u>Method categoryTagHandler</u>	887
<u>Method checkEventPop</u>	887
<u>Method checkEventPush</u>	887
<u>Method configWordParser</u>	888
<u>Method defaultTagHandler</u>	888
<u>Method endTag</u>	888
<u>Method exampleTagHandler</u>	888
<u>Method getParserEventName</u>	889
<u>Method globalTagHandler</u>	889
<u>Method invalidTagHandler</u>	889
<u>Method packageTagHandler</u>	890
<u>Method paramTagHandler</u>	890
<u>Method parse</u>	890
<u>Method propertyTagHandler</u>	891
<u>Method returnTagHandler</u>	891
<u>Method setupStates</u>	891
<u>Method staticvarTagHandler</u>	892
<u>Method usesTagHandler</u>	892
<u>Method varTagHandler</u>	892
<u>Class parserDescParser</u>	893
<u>Constructor parserDescParser</u>	893
<u>Method doSimpleList</u>	893
<u>Method getParserEventName</u>	894
<u>Method parse</u>	894
<u>Method setupStates</u>	894
<u>Class phpDocumentorTParser</u>	895
<u>Var \$eventHandlers</u>	896
<u>Var \$inlineTagHandlers</u>	896
<u>Var \$source_location</u>	896
<u>Constructor phpDocumentorTParser</u>	896
<u>Method parse</u>	896
<u>Class phpDocumentor_HighlightParser</u>	897
<u>Constructor phpDocumentor_HighlightParser</u>	897
<u>Method configWordParser</u>	897
<u>Method newLineNum</u>	898
<u>Method parse</u>	898
<u>Method setLineNum</u>	899
<u>Method setupStates</u>	899
<u>Class phpDocumentor_HighlightWordParser</u>	899
<u>Method backupPos</u>	900
<u>Method getWord</u>	900
<u>Method nextToken</u>	900
<u>Method setup</u>	900
<u>Class phpDocumentor_peardoc2_XML_Beautifier</u>	901
<u>Method formatFile</u>	901
<u>Method formatString</u>	901
<u>Class phpDocumentor_TutorialHighlightParser</u>	902
<u>Constructor phpDocumentor_TutorialHighlightParser</u>	902

<u>Method checkEventPop</u>	902
<u>Method checkEventPush</u>	903
<u>Method configWordParser</u>	903
<u>Method getParserEventName</u>	904
<u>Method newLineNum</u>	904
<u>Method parse</u>	904
<u>Method setLineNum</u>	905
<u>Method setupStates</u>	905
<u>Class phpDocumentor XML Beautifier Tokenizer</u>	905
<u>Var \$eventHandlers</u>	906
Constructor <u>phpDocumentor XML Beautifier Tokenizer</u>	906
<u>Method checkEventPop</u>	906
<u>Method checkEventPush</u>	906
<u>Method configWordParser</u>	907
<u>Method getParserEventName</u>	907
<u>Method inCDATAHandler</u>	907
<u>Method parseString</u>	907
<u>Method setupStates</u>	908
<u>Class ppageParser</u>	909
<u>Var \$package</u>	909
<u>Var \$subpackage</u>	909
Constructor <u>ppageParser</u>	909
<u>Method defaultHandler</u>	909
<u>Method handleInlineDockeyword</u>	910
<u>Method parse</u>	910
<u>Method setupStates</u>	910
<u>Class XMLPackagePageParser</u>	910
<u>Var \$context</u>	911
<u>Var \$eventHandlers</u>	911
<u>Var \$pars</u>	911
<u>Var \$refsect1id</u>	911
<u>Var \$refsect2id</u>	911
<u>Var \$refsect3id</u>	911
Constructor <u>XMLPackagePageParser</u>	911
<u>Method getParserEventName</u>	912
<u>Method parse</u>	913
<u>Method setupStates</u>	913
<u>find_phpdoc.php</u>	914
<u>PackagePageElements.inc</u>	915
<u>Class parserCData</u>	916
<u>Method Convert</u>	916
<u>Class parserEntity</u>	917
Constructor <u>parserEntity</u>	917
<u>Method Convert</u>	917
<u>Class parserXMLDocBookTag</u>	918
<u>Var \$attributes</u>	918
<u>Var \$name</u>	919
Constructor <u>parserXMLDocBookTag</u>	919
<u>Method add</u>	919

<a href="#">Method addAttribute</a>	919
<a href="#">Method addCData</a>	920
<a href="#">Method Convert</a>	920
<a href="#">Method endCData</a>	920
<a href="#">Method getId</a>	920
<a href="#">Method getSubSection</a>	921
<a href="#">Method getTitle</a>	921
<a href="#">Method getTOC</a>	921
<a href="#">Method hasTitle</a>	921
<a href="#">Method setId</a>	921
<a href="#">Method setTitle</a>	922
<a href="#">Method setTOC</a>	922
<a href="#">Method startCData</a>	922
<a href="#"><u>phpDocumentorTWordParser.inc</u></a>	923
<a href="#"><u>WordParser.inc</u></a>	924
<a href="#">Class ObjectWordParser</a>	924
<a href="#">    Constructor ObjectWordParser</a>	925
<a href="#">    Method getWord</a>	925
<a href="#">    Method nextIsObjectOrNonNL</a>	925
<a href="#">    Method setup</a>	925
<a href="#">Class phpDocumentorTWordParser</a>	925
<a href="#">    Method addFileSource</a>	926
<a href="#">    Method addSource</a>	926
<a href="#">    Method backupPos</a>	926
<a href="#">    Method concatTokens</a>	927
<a href="#">    Method findGlobal</a>	927
<a href="#">    Method getFileSource</a>	927
<a href="#">    Method getSource</a>	928
<a href="#">    Method getWord</a>	928
<a href="#">    Method setup</a>	928
<a href="#">    Method tokenEquals</a>	928
<a href="#">Class WordParser</a>	928
<a href="#">    Method backupPos</a>	929
<a href="#">    Method getBlock</a>	929
<a href="#">    Method getPos</a>	929
<a href="#">    Method getSource</a>	929
<a href="#">    Method getWord</a>	930
<a href="#">    Method setPos</a>	930
<a href="#">    Method setSeparator</a>	930
<a href="#">    Method setup</a>	931
<a href="#">    Method setWhitespace</a>	931
<a href="#"><b>Appendices</b></a>	932
<a href="#">    Appendix A - Class Trees</a>	933
<a href="#">        phpDocumentor</a>	933
<a href="#">        HTML_TreeMenu</a>	939
<a href="#">        tests</a>	939
<a href="#">        org-phpdoc</a>	945
<a href="#">        Converters</a>	945
<a href="#">        XML_Beautifier</a>	946

<a href="#">Cpdf</a>	946
<a href="#">Smarty</a>	947
<a href="#">Appendix B - README/CHANGELOG/INSTALL</a>	948
<a href="#">README</a>	949
<a href="#">ChangeLog</a>	952
<a href="#">INSTALL</a>	952
<a href="#">FAQ</a>	954
<a href="#">Appendix C - Source Code</a>	958
<a href="#">Package phpDocumentor</a>	959
<a href="#">source code: Converter.inc</a>	960
<a href="#">source code: CHMdefaultConverter.inc</a>	1034
<a href="#">source code: HTMLframesConverter.inc</a>	1060
<a href="#">source code: HTMLSmartyConverter.inc</a>	1086
<a href="#">source code: class.phpdocpdf.php</a>	1113
<a href="#">source code: ParserPDF.inc</a>	1118
<a href="#">source code: PDFdefaultConverter.inc</a>	1126
<a href="#">source code: XMLDocBookpeardoc2Converter.inc</a>	1140
<a href="#">Package phpDocumentor</a>	1163
<a href="#">source code: file_dialog.php</a>	1164
<a href="#">Package phpDocumentor</a>	1168
<a href="#">source code: builder.php</a>	1169
<a href="#">source code: config.php</a>	1171
<a href="#">source code: top.php</a>	1181
<a href="#">source code: new_phpdoc.php</a>	1183
<a href="#">source code: phpdock.php</a>	1184
<a href="#">source code: Classes.inc</a>	1185
<a href="#">source code: clone.inc.php</a>	1203
<a href="#">source code: clone5.inc.php</a>	1204
<a href="#">source code: common.inc.php</a>	1205
<a href="#">source code: EventStack.inc</a>	1210
<a href="#">source code: IntermediateParser.inc</a>	1212
<a href="#">source code: Io.inc</a>	1238
<a href="#">source code: phpdock.inc</a>	1252
<a href="#">source code: DeschHTML.inc</a>	1253
<a href="#">source code: DocBlockTags.inc</a>	1259
<a href="#">source code: Errors.inc</a>	1277
<a href="#">source code: InlineTags.inc</a>	1292
<a href="#">source code: LinkClasses.inc</a>	1306
<a href="#">source code: Beautifier.php</a>	1311
<a href="#">source code: Tokenizer.php</a>	1313
<a href="#">source code: HighlightParser.inc</a>	1323
<a href="#">source code: find_phpdock.php</a>	1356
<a href="#">Appendix D - Todo List</a>	1357

# phpDocumentor Guide to Creating Fantastic Documentation

*What makes good documentation? This is unanswerable, but there are a few things to keep in mind*

## Why write good documentation for open source code?

Writing good documentation is essential to the success of any software project. The quality of documentation can be even more important than the quality of the code itself, as a good first impression will prompt developers to look further into your code. phpDocumentor is designed to make it easier to create documentation. phpDocumentor even makes it possible to generate separate sets of documentation from the same source!

## Writing Great Documentation

### Consider the audience

The first question any writer must ask is "Who is my audience?" This will answer many questions. For most software, such as any MS product, the software is intended only to be used. The programmers are the only people who have access to the source code. The challenge in this case is to write good documentation for end-users.

When writing documentation for an open-source project intending to be both used and extended, this challenge is magnified by the fact that many people will also intend to extend the source, or even find obscure bugs and fix them. These two audiences tend to be opposed to each other in their needs.

An end-user generally wants:

- Instruction-style writing, that explains and describes general concepts more than how a particular variable is used
- Interface information only, no low-level details
- Examples of how to use, and tutorials

Whereas a programmer may want in addition:

- Details on how program elements interact, which elements use others
- Where in the source code an action or series of actions occurs
- How to extend the code to add new functionality

These two kinds of users can not be serviced by just API documentation or just tutorials, but a subtle blend of the two.

## Using phpDocumentor to make documentation for two separate audiences

In essence, there may need to be two separate sets of documentation! phpDocumentor can be used to create this

option using a few things. First, using the command-line file-selection options, one can write two sets of documentation, one for end-users, and the other for programmers, and put them in different subdirectories. For instance, one could put enduser docs in "enduser/tutorials" and programmer docs in "programmer/tutorials." In addition, using the [@access](#) tag, one can mark programmer-level elements with @access private, and they will be ignored by default. The [@internal](#) tag and [inline {@internal}}](#) inline tag construct can be used to mark documentation that is low-level or internal to a smaller audience of developers. When creating programmer documentation, turn on the parse private option (see [-pp, --parseprivate](#)), and the low-level detail will be generated.

More important, the in-code documents in your [DocBlocks](#) must be written for both end-users and programmers wherever possible.

phpDocumentor uses the chaining feature of tutorials to link together related documentation like chapters in a book, and this is another way to organize documentation. Experiment, and find what's best for your project's needs.

## Tips on Revising

There are many ways of creating good documentation, but perhaps the best is to read what you have written from different perspectives. Open up your documentation, and try to use it to figure out your software project. If this is difficult, go back and edit or rewrite. Remove anything that is confusing or unnecessary, make sure everything is there that is needed, and then when it seems good, ask a php user that doesn't know your project, or even a non-programmer to read it and use their reactions to tailor the documentation.

## Conclusion

The use of phpDocumentor will definitely improve the look and feel of your documentation, but don't stop there, let it help you to create truly dynamic documents that are easily maintained and kept up-to-date. After all, great docs for great programs will help to secure PHP's rightful throne as the best out there!

## References for learning how to use phpDocumentor

- [phpDocumentor Quickstart](#)

# phpDocumentor Quickstart

*phpDocumentor for newbies*

## What is phpDocumentor? What can it do?

phpDocumentor is a tool written in PHP designed to create complete documentation directly from both PHP code and external documentation. The truth is, PHP source code is so lucid, it can practically serve as its own documentation. phpDocumentor taps into this fact by parsing out all kinds of logical structures already found in PHP, such as files, classes, functions, define constants, global variables, and class variables/methods and organizes them into a traditional manual format. In addition, new with version 1.3.0, source code elements introduced in PHP 5 (class constants, interfaces, and others) can also be parsed, if phpDocumentor is run through PHP 5. Output can be created for remote web browsing, print, and integration into IDE help systems through converters for HTML, PDF, and CHM (windows help files).

phpDocumentor generates manual-format documentation by reading it from special PHP comments called [DocBlocks](#).

DocBlocks are where you as the author of a software project should document helpful information that can be used by others (or your future self) to determine how to use and extend your PHP package.

Although the ability to add succinct documentation in the source code is essential, it cannot replace the importance of verbose documentation that is not in the source code, such as a user manual, or tutorials such as the one you are reading now. If the text you see here were placed in the source code for phpDocumentor, it would serve little useful purpose, clutter up the code, and be difficult to locate. However, the ability to hyperlink between documentation in the source code and external documentation is essential. External documentation for function foo must be able to reference the generated in-code documentation, and with phpDocumentor this is finally possible. To learn more, read about [phpDocumentor Tutorials](#).

## Installation

There are two official installation methods for phpDocumentor. The first is through downloading and extracting one of the available archives downloadable through pear.php.net and sourceforge.net, and the other is through the PEAR installer. There is planning for a [Phing](#) task and for distribution through other new and promising installation frameworks like the [ZZ/OSS installer](#). However, only the two official installation methods are supported by phpDocumentor's developers.

## Download from Pear.Php.net or Sourceforge.net

To install phpDocumentor from a .zip or tarball downloaded directly from pear.php.net or

sourceforge.net, first determine whether you will be using phpDocumentor's web or command-line interface (see the [Basic usage of phpDocumentor tool](#) section for help in making this decision).

If you wish to use the command-line interface, unzip the archive into any directory, say /home/myuser/phpdoc or C:\Program Files\phpdoc, and add that directory to your path statement. To use, run the "phpdoc" command. In windows, you will need to edit the phpdoc.bat file, and change the first line to the path of the CLI version of PHP (usually C:\php4\cli\php.exe by default).

To use the web interface, you must have a web server such as Apache installed, and must have a working PHP sapi for that webserver. To test, save the code below as phpinfo.php in your document root and browse to <http://localhost/phpinfo.php>

phpinfo.php:

```
1 <?php
2 phpinfo\(\);
3 ?>
```

If you see a beautiful purple display of PHP information, then your PHP setup is working. To use phpDocumentor's web interface, simply unzip the archive into a subdirectory of your document root (such as phpdoc) and browse to that location (<http://localhost/phpdoc>).

## Caution

A Javascript-enabled browser such as Netscape, Mozilla, Internet Explorer, Opera, or Konqueror is required to view the newer web interface. If you wish to use a non-javascript browser such as links/lynx, use the old web interface, phpdoc.php at <http://localhost/phpdoc/phpdoc.php>.

## Installation through [PEAR](#)

To install phpDocumentor through PEAR, you must first have a working installation of PEAR. Instructions for properly installing PEAR are located at the official PEAR website, <http://pear.php.net>. phpDocumentor developers do not support installation issues with PEAR, instead seek help from PEAR developers.

Installing phpDocumentor for use on the command-line is simple. Simply run:

```
$ pear install PhpDocumentor
```

and you then have full access  
to the phpdoc command, both in windows and unix, without further configuration.

To install phpDocumentor to use the web interface, you must first change one of PEAR's configuration variables, data\_dir, to be a sub-directory of your web server's document root. The simplest way to do this is through PEAR's command-line interface with the command:

```
$ pear config-set data_dir /path/to/document_root/pear
```

Configuring this value through the web interface is also simple. Click on the configuration icon in the left-hand frame, and type in the path in the data\_dir text box.

Once this configuration is complete, simply install phpDocumentor as described in the second paragraph above, and you can then browse to <http://localhost/pear/PhpDocumentor> to have access to the web interface. Once this configuration step has been taken, there is never any need to change, and you can easily upgrade to future phpDocumentor versions by using pear's upgrade command.

## How to document your code for use with phpDocumentor

Documenting your code projects is straightforward and intuitive for the most part.

Before you begin, you may wish to download the sample documentation project and try running phpDocumentor on them as you read along, found at

[Source code for sample1.php](#), [Source code for sample2.php](#), [Source code for sample3.php](#) or found bundled in the tutorials/ directory of your phpDocumentor distribution. Before parsing, copy the examples to another directory.

First of all, run phpDocumentor on the blank file sample1.php. If you are using the command-line interface, run this command from the tutorials/sample directory:

```
$ phpdoc -o HTML:frames:earthli -f sample1.php -t docs
```

If you are using the web interface, click the "Files" tab, and type in the full path to tutorials/sample/sample1.php. Then click the "Output" tab and type in the full path to tutorials/sample/docs, and finally click the "Create" button in the lower right-hand corner.

With a web browser, open the newly created tutorials/sample/docs/index.html file and you will see the rich array of information that phpDocumentor can parse from the source code even without any documentation comments. Inheritance information, polymorphism and constants are all recognized automatically. Note that the only element that is not automatically documented is the global variable - to do this, you must use a phpDocumentor DocBlock as described in the next section of this quickstart manual. Also note that although the include\_once specifies a file within the include\_path, phpDocumentor will not automatically parse sample2.php, you must manually specify the files or directories to parse.

If you're feeling adventurous, experiment with the parse options available and parse the sample files a few times to see how they affect the documentation output.

To find out options in the command-line interface, type

```
$ phpdoc -h
```

## Documenting your PHP source code with comments

Now open [sample2.php](#). This is the same code content as sample1.php, but it contains a full array of phpDocumentor DocBlock comments.

Note that every DocBlock comment is a C-style comment with two leading asterisks (\*), like so:

```
1  /**
2  *
3  */
```

All other comments are ignored by the documentation parser. Note that although most of the documentation is plain English, there are a few "@" characters floating around the source code. This symbol is used to pass a special command to the parser, and is called a tag. If the symbol is at the beginning of a line, it is a standard tag, and if it is enclosed in {curly brackets}, it is an inline tag. You can read more about tags at [phpDocumentor tags](#) and [phpDocumentor Inline tags](#).

```
1  /**
2  * {@inlinetag}
3  * this is @not a standardtag - must begin a line.
4  * this is a valid {@inlinetag} also
5  * @standardtag
6  */
```

### *Documenting global variables*

Notice this code from sample2.php:

```
1  /**
2  * Special global variable declaration DocBlock
3  * @global integer $GLOBALS['_myvar']
4  * @name $_myvar
5  */
6  $GLOBALS['_myvar'] = 6;
```

In this segment, we can see the two important tags used for documenting global variables. The [@global](#) tag is used to tell the parser how to find a global variable declaration. Without this tag, no documentation would be generated for `$_myvar`. The [@global](#) tag can take many forms - be sure to specify the type and global name, with no description, or the parser will generate a warning and fail to document the variable.

Now, parse [sample3.php](#) and observe the generated documentation.

The [@name](#) tag is used to tell the parser how the global variable would be referenced by a global statement in a function.

```
1  /**
2  * @global integer this is a function-based @global tag,
3  *      because the variable name is missing
4  */
5  function someFunction()
6  {
```

```
7     global$_myvar;
8 }
```

The @global tag here will associate the information with the declared global statements in the function body in the same order of their declaration.

### *Page-level DocBlock warnings*

The single most commonly asked question about using phpDocumentor involves warnings about Page-level DocBlocks. This section will answer any questions about this warning once and for all.

phpDocumentor organizes procedural elements and classes into special groupings called [packages](#).

In earlier versions of phpDocumentor, if package was not specified explicitly using the [@package](#) tag, the program would make an educated guess as to which package a source element belongs to.

Over time, it became apparent that in many cases, source elements were incorrectly grouped into a package due to the guesswork phpDocumentor uses. Finally, the decision was made to require an explicit @package tag, and to raise a warning any time this tag was missing from a top-level source element.

The greatest confusion comes from the documenting of files. phpDocumentor documents all source elements by reading the DocBlock that immediately precedes the element, like so:

```
1  <?php
2 /**
3  * Documents the class following
4  * @package SomePackage
5  */
6 class SomeClass {
7 }
8 ?>
```

phpDocumentor also can document the contents of a file. But how can a DocBlock immediately precede the file that contains it? The easy answer is to assume the first DocBlock in a file documents the file that contains it, and this works well, but can be deceptive:

```
1  <?php
2 /**
3  * Documents the class following or the file?
4  * @package SomePackage
5  */
6 class SomeClass {
7 }
8 ?>
```

The same example shows the ambiguity - does this DocBlock document the class, or the file? To resolve this ambiguity, phpDocumentor uses a simple algorithm to make its decision.

1. If the first DocBlock in a file contains a @package tag, it documents the file unless it precedes a class declaration

If the first DocBlock in a file precedes another DocBlock, it documents the file

## Writing external documentation and linking to source code documentation

Although documentation parsed directly from source code is tremendously useful, it cannot stand on its own. In addition, truly useful in-code documentation must be succinct enough so that the code is not completely obscured by the documentation. External documentation is a must for a complete documentation solution. However, external documentation must be able to link to API source documentation to be useful. With a constantly changing API documentation, it is very easy for external documentation to become out of date. In addition, external documentation must be in a format that can be converted into other formats such as HTML, PDF and XML.

phpDocumentor provides a simple and elegant solution to all of these problems. External documentation in DocBook format can be easily converted to other formats. Using inline tags, phpDocumentor can generate a consistent manual in many different formats by combining the output from parsing the source and parsing external documentation. The words you read at this moment are in a DocBook-based file located in tutorials/phpDocumentor/phpDocumentor.quickstart.pkg

```
1 <refentry id="@id"      >
2 <refnamediv>
3   <refname> Simple Tutorial</refname>
4   <refpurpose> The simplest Tutorial Possible</refpurpose>
5 </refnamediv>
6 <refsynopsisdiv>
7   <author>
8     Gregory Beaver
9     <authorblurb>
10    {@link mailto:celog@php.net celog@php.net}
11   </authorblurb>
12   </author>
13 </refsynopsisdiv>
14 <refsect1 id="@id intro"      >
15   <para> Hello World</para>
16 </refsect1>
17 </refentry>
```

## Basic usage of phpDocumentor tool

Now that phpDocumentor is installed, how can you use it to document a project? First, you must understand how phpDocumentor divides your code into packages and subpackages. If you haven't already, now would be a good time to read the [How to document your code for use with phpDocumentor](#) section of this quickstart.

## Command-line tool, phpdoc

## docbuilder, the phpDocumentor web interface



# Source code for sample1.php

*phpDocumentor for newbies tutorial*

## sample1.php

```
1 <?php
2 // sample file #1
3
4 include_once 'sample2.php';
5
6 $GLOBALS['_myvar'] = 6;
7
8 define('testing', 6);
9 define('anotherconstant', strlen('hello'));
10
11 function firstFunc($param1, $param2= 'optional')
12 {
13     static $staticvar= 7;
14     global $_myvar;
15     return $staticvar
16 }
17
18 class myclass {
19     var $firstvar= 6;
20     var $secondvar=
21         array(
22             'stuff' =>
23                 array(
24                     6,
25                     17,
26                     'armadillo'
27                 ),
28             testing => anotherconstant
29 );
30
31     function myclass()
32     {
33         $this-> firstvar= 7;
34     }
35
36     function parentfunc($param1)
37     {
38         if ($param1) {
39             return 6;
40         } else {
41             return new babyclass;
42         }
43     }
44 }
45
46 class babyclass extends myclass {
47     var $secondvar= 42;
```

```
48 var $thirdvar
49
50 function babyclass()
51 {
52     parent::myclass();
53     $this-> firstvar++;
54 }
55
56 function parentfunc($param)
57 {
58     returnnew myclass;
59 }
60 }
61 ?>
```

# Source code for sample2.php

*phpDocumentor for newbies tutorial*

## sample2.php

```
1 <?php
2 /**
3 * Sample File 2, phpDocumentor Quickstart
4 *
5 * This file demonstrates the rich information that can be included in
6 * in-code documentation through DocBlocks and tags.
7 * @author Greg Beaver <cellog@php.net>
8 * @version 1.0
9 * @package sample
10 */
11 // sample file #1
12 /**
13 * Dummy include value, to demonstrate the parsing power of phpDocumentor
14 */
15 include_once 'sample3.php';
16
17 /**
18 * Special global variable declaration DocBlock
19 * @global integer $GLOBALS['_myvar']
20 * @name $_myvar
21 */
22 $GLOBALS['_myvar'] = 6;
23
24 /**#@+
25 * Constants
26 */
27 /**
28 * first constant
29 */
30 define('testing', 6);
31 /**
32 * second constant
33 */
34 define('anotherconstant', strlen('hello'));
35
36 /**
37 * A sample function docblock
38 * @global string document the fact that this function uses $_myvar
39 * @staticvar integer $staticvar this is actually what is returned
40 * @param string $param1 name to declare
41 * @param string $param2 value of the name
42 * @return integer
43 */
44 function firstFunc($param1, $param2= 'optional')
45 {
46     static $staticvar= 7;
47     global $_myvar;
```

```

48     return $staticvar;
49 }
50
51 /**
52 * The first example class, this is in the same package as the
53 * procedural stuff in the start of the file
54 * @package sample
55 * @subpackage classes
56 */
57 class myclass {
58 /**
59 * A sample private variable, this can be hidden with the --parseprivate
60 * option
61 * @access private
62 * @var integer|string
63 */
64 var $firstvar= 6;
65 /**
66 * @link http://www.example.com Example link
67 * @see myclass()
68 * @uses testing, anotherconstant
69 * @var array
70 */
71 var $secondvar=
72     array(
73         'stuff' =>
74             array(
75                 6,
76                 17,
77                 'armadillo'
78             ),
79             testing => anotherconstant
80     );
81
82 /**
83 * Constructor sets up {@link $firstvar}
84 */
85 function myclass()
86 {
87     $this-> firstvar= 7;
88 }
89
90 /**
91 * Return a thingie based on $paramie
92 * @param boolean $paramie
93 * @return integer|babyclass
94 */
95 function parentfunc($paramie)
96 {
97     if ($paramie) {
98         return 6;
99     } else {
100         return new babyclass;
101     }
102 }
103 }
104
105 /**
106 * @package sample1

```

```

107  /*
108  class babyclass extends myclass {
109  /**
110  * The answer to Life, the Universe and Everything
111  * @var integer
112  */
113 var $secondvar= 42;
114 /**
115 * Configuration values
116 * @var array
117 */
118 var $thirdvar
119
120 /**
121 * Calls parent constructor, then increments {@link $firstvar}
122 */
123 function babyclass()
124 {
125     parent::myclass();
126     $this-> firstvar++;
127 }
128
129 /**
130 * This always returns a myclass
131 * @param ignored $paramie
132 * @return myclass
133 */
134 function parentfunc($paramie)
135 {
136     returnnew myclass;
137 }
138 }
139 ?>
```

# Source code for sample3.php

*phpDocumentor for newbies tutorial*

## sample3.php

```
1 <?php
2 /**
3 * Sample File 3, phpDocumentor Quickstart
4 *
5 * This file demonstrates the use of the @name tag
6 * @author Greg Beaver <cellog@php.net>
7 * @version 1.0
8 * @package sample
9 */
10 /**
11 * Special global variable declaration DocBlock
12 * @global integer $GLOBALS['_myvar']
13 */
14 $GLOBALS['_myvar'] = 6;
15 /**
16 * Notice that the @name tag does not validate what you give it!
17 * @global string $GLOBALS['turkey']
18 * @name $turkify
19 */
20 $GLOBALS['turkey'] = 'tester';
21 /**
22 * Sample function @global usage
23 *
24 * Note that the $turkey variable is not linked to its documentation at
25 * {@link $turkify} because of the improper @name tag
26 * @global integer
27 * @global string this one has the optional description
28 */
29 function testit()
30 {
31     global $_myvar, $turkey
32 }
33 ?>
```

# phpDocumentor Tutorial

*An in-depth look at using phpDocumentor to document PHP Source Code, and  
phpDocumentor internals*

## Introduction

phpDocumentor is the most advanced automatic documentation system written for PHP, in PHP. This package has many features:

- **NEW** The first auto-documentor with PHP 5 support
- Extended documentation in docbook format with linking to any element and to other documentation, including sub-sections, from the source code (see [phpDocumentor Tutorials](#))
- docblock templates to cut down on repetition
- XML:DocBook:peardoc2 templates for PEAR developers
- Greater ease of extending a Converter, see [Converter Manual](#)
- ability to parse any PHP file, regardless of documentation format
- conforms loosely to the [JavaDOC protocol](#), and will be familiar to Java programmers
- documents all includes, constants, functions, static functions, classes, methods, static variables, class variables, and can document global variables and external tutorials
- auto-linking to pre-defined PHP functions
- Output in HTML, CHM, PDF, XML DocBook formats
- templateable with many bundled templates
- automatic linking to elements in any documented package
- documents name conflicts between packages to help avoid PHP errors
- document CVS repository directly.
- support for JavaDoc doclet-like output through Converters
- error/warning tracking system
- extreme class intelligence: inherits documentation, package
- complete phpdoc.de DocBlock tag support. Additions include @var, @magic, @deprec, @todo, and phpdoc.de parsing of @param.
- alphabetical indexing of all elements by package and overall
- class trees
- MUCH more than just this short list

## phpDocumentor Basics

### Starting Out From Scratch

The documentation process begins with the most basic element of phpDocumentor: a **Documentation block** or **DocBlock**. A basic DocBlock looks like this:

```
1  /**
2  *
3  */
```

A DocBlock is an extended C++-style PHP comment that begins with "/\*" and has an "\*" at the beginning of every line. DocBlocks precede the element they are documenting.

## Caution

Any line within a DocBlock that doesn't begin with a \* will be ignored.

To document function "foo()", place the DocBlock immediately before the function declaration:

```
1  /**
2  * Defies imagination, extends boundaries and saves the world ...all before breakfast!
3  */
4  function foo()
5  {
6  }
```

This example will apply the DocBlock to "define('me',2);" instead of to "function foo()":

```
1  /**
2  * DocBlock for function foo?
3  *
4  * No, this will be for the constant me!
5  */
6  define('me',2);
7  function foo($param= me)
8  {
9  }
```

define() statements, functions, classes, class methods, and class vars, include() statements, and global variables can all be documented, see  
[Elements of the source code that can be documented](#)

## DocBlocks

A DocBlock contains three basic segments in this order:

- Short Description
- Long Description
- Tags

The Short Description starts on the first line, and can be terminated with a blank line or a period. A period inside a word (like example.com or 0.1 %) is ignored. If the Short Description would become more than three lines long, only the first line is taken. The Long Description continues for as many lines as desired and may contain html markup for display formatting. Here is a sample DocBlock with a Short and a Long Description:

```
1  /**
2  * return the date of Easter
```

```
3  *
4  * Using the formula from "Formulas that are way too complicated for anyone to
5  * ever understand except for me" by Irwin Nerdy, this function calculates the
6  * date of Easter given a date in the Ancient Mayan Calendar, if you can also
7  * guess the birthday of the author.
8  */
```

Optionally, you may enclose all paragraphs in a `<p></p>` tag. Be careful, if the first paragraph does not begin with `<p>`, phpDocumentor will assume that the DocBlock is using the simple double linebreak to define paragraph breaks as in:

```
1  /**
2  * Short desc
3  *
4  * Long description first sentence starts here
5  * and continues on this line for a while
6  * finally concluding here at the end of
7  * this paragraph
8  *
9  * The blank line above denotes a paragraph break
10 */
```

Here is an example of using `<p>`

```
1  /**
2  * Short desc
3  *
4  * <p>Long description first sentence starts here
5  * and continues on this line for a while
6  * finally concluding here at the end of
7  * this paragraph</p>
8  * This text is completely ignored! it is not enclosed in p tags
9  * <p>This is a new paragraph</p>
10 */
```

phpDocumentor also supports JavaDoc's DocBlock format through the command-line option [-j, --javadocdesc](#). Due to the non-xhtml compliant unmatched `p` tag, we highly recommend you avoid this syntax whenever possible

```
1  /**
2  * <p>
3  * Short desc is only to the first period.
4  * This means a sentence like:
5  * "Parses Mr./Mrs. out of $_GET." will
6  * parse a short description of "Parses Mr."
7  * which is rather silly. Long description is
8  * the entire DocBlock description including the
9  * Short desc, and paragraphs begin where p is like:
10 * <p>
11 * The p above denotes a paragraph break
12 */
```

phpDocumentor will convert all whitespace into a single space in the long description, use paragraph breaks to define newlines, or <pre>, as discussed in the next section.

## DocBlock Description details

As of phpDocumentor 1.2.0, the long and short description of a DocBlock is parsed for a few select html tags that determine additional formatting. Due to the nature of phpDocumentor's output as multiple-format, straight html is not allowed in a DocBlock, and will be converted into plain text by all of the converters unless it is one of these tags:

- <b> -- emphasize/bold text
- <code> -- Use this to surround php code, some converters will highlight it
- <br> -- hard line break, may be ignored by some converters
- <i> -- italicize/mark as important
- <kbd> -- denote keyboard input/screen display
- <li> -- list item
- <ol> -- ordered list
- <p> -- If used to enclose all paragraphs, otherwise it will be considered text
- <pre> -- Preserve line breaks and spacing, and assume all tags are text (like XML's CDATA)
- <samp> -- denote sample or examples (non-php)
- <ul> -- unordered list
- <var> -- denote a variable name

Do not think of these tags as text, they are parsed into objects and converted into the appropriate output format by the converter. So, a b tag may become <emphasis> in DocBook, and a <p> tag might become " " (4 spaces) in the PDF converter! The text output is determined by the template options file options.ini found in the base directory of every template. For instance, the options.ini file for the HTML:frames:default template is in phpDocumentor/Converters/HTML/frames/templates/default/options.ini

For the rare case when the text "<b>" is needed in a DocBlock, use a double delimiter as in <<b>>. phpDocumentor will automatically translate that to the physical text "<b>".

Similarly, if you need an actual "@" in your DocBlock's description parts, you should be careful to either ensure it is not the first character on a line, or else escape it ("\@") to avoid it being interpreted as a PhpDocumentor tag marker.

```
1  /**
2  * Demonstrate an @include file
3  * line in a code block:
4  *
5  * <code>
6  * \@include somefile.php
7  * </code>
8  */
```

This will parse as if it were:

```

1  /**
2  * Demonstrate an @include file
3  * line in a code block:
4  *
5  * <code>
6  * @include somefile.php
7  * </code>
8 */

```

The `<code>`, `<kbd>`, and `<pre>` tags ignore any html listed above except for their own closing tags (`</code>` `</kbd>` `</pre>`).

This is obviously necessary for each of those tag's behavior of controlling the appearance of the text inside their tag blocks, without allowing other nested tags (like `<b>`) interfering inside them.

In general, other tags will allow other tags to be nested in them  
(i.e. `<samp><b>bold_my_sample</b></samp>`).

**New 1.2.0rc1:** If you need to use the closing comment `"*/"` in a DocBlock, use the special escape sequence `"{@*}"`. Here's an example:

```

1  /**
2  * Simple DocBlock with a code example containing a docblock
3  *
4  * <code>
5  * /**
6  *   * My sample DocBlock in code
7  *   {@*}
8  * </code>
9 */

```

This will parse as if it were:

```

1  /**
2  * Simple DocBlock with a code example containing a docblock
3  *
4  * <code>
5  * /**
6  *   * My sample DocBlock in code
7  *   */
8  * </ code>
9 */

```

**New 1.2.0rc1:** The phpEdit tool supports very clever list extraction from DocBlocks, and now phpDocumentor supports the same cleverness. This example:

```

1  /**
2  * Simple DocBlock with simple lists
3  *
4  * Here's a simple list:
5  * - item 1
6  * - item 2, this one

```

```

7  *  is multi-line
8  * - item 3
9  * end of list. Next list is ordered
10 * 1 ordered item 1
11 * 2 ordered item 2
12 * end of list. This is also ordered:
13 * 1. ordered item 1
14 * 2. ordered item 2
15 */

```

phpDocumentor recognizes any simple list that begins with "-", "+", "#", and "o", and any ordered list with sequential numbers or numbers followed by "." as above. The list delimiter must be followed by a space (not a tab!) and whitespace must line up exactly, or phpDocumentor will not recognize the list items as being in the same list.

```

1 /**
2 * Simple DocBlock with screwy lists
3 *
4 * +not a list at all, no space
5 * Here's 3 lists!
6 * - item 1
7 * - item 2, this one
8 *  is multi-line
9 *- item 3
10 */

```

Again, you must line up the list iterators in the same vertical column in order for those list items to be considered "on the same list". Also, you cannot create nested lists using the simple list iterators... doing so means it's no longer a "simple" list! Varying the spacing of the list iterators does not create nesting of the lists... it only starts up new simple lists. If you specifically need a nested list, you must use the list tags (<ol>, <ul>):

```

1 /**
2 * DocBlock with nested lists
3 *
4 * Here's the "complex" list:
5 * <ul>
6 * <li>outer item 1</li>
7 * <li>outer item 2, this one
8 *  is multi-line</li>
9 * <li>item 3 is a nested inner list
10 * <ul>
11 * <li>inner item 1</li>
12 * <li>inner item 2</li>
13 * </ul>
14 * <li>outer item 4</li>
15 * </ul>
16 */

```

In some cases, the text description in a doc tag can contain a list, be it simple

or complex. Notice that a title line is needed, with the list items themselves beginning on the next line:

```
1  /**
2  * DocBlock with nested lists
3  * in the tag descriptions
4  * @todo My Simple TODO List
5  *   - item 1
6  *   - item 2
7  *   - item 3
8  * @todo My Complex TODO List
9  *   <ol>
10 *     <li>item 1.0</li>
11 *     <li>item 2.0</li>
12 *     <li>item 3.0</li>
13 *     <ol>
14 *       <li>item 3.1</li>
15 *       <li>item 3.2</li>
16 *     </ol>
17 *     <li>item 4.0</li>
18 *   </ol>
19 */
```

Tagged lists are always a more robust and predictable option for you to use. Simple lists are convenient, but if you find yourself trying to bend a simple list into displaying a certain way in your generated docs, you may be better served by switching to a tagged list instead.

For in-depth information on how phpDocumentor parses the description field, see [ParserDescCleanup.inc](#)

## DocBlock Templates

New for version 1.2.0, phpDocumentor supports the use of DocBlock templates. The purpose of a DocBlock template is to reduce redundant typing. For instance, if a large number of class variables are private, one would use a DocBlock template to mark them as private. DocBlock templates simply augment any normal DocBlocks found in the template block.

A DocBlock template is distinguished from a normal DocBlock by its header. Here is the most basic DocBlock template:

```
1  /**#@+
2  *
3  */
```

The text that marks this as a DocBlock template is "/\*\*#@+"  
- all 6 characters must be present. DocBlock templates are applied to all documentable elements until the ending template marker:

```
1  /**@-*/
```

Note that all 8 characters must appear as "/\*#@-\*/" in order for phpDocumentor to recognize them as a template. Here is an example of a DocBlock template in action:

```
1 class Bob
2 {
3     // beginning of docblock template area
4     /**#@+
5      * @access private
6      * @var string
7      */
8     var $_var1= 'hello';
9     var $_var2= 'my';
10    var $_var3= 'name';
11    var $_var4= 'is';
12    var $_var5= 'Bob';
13    var $_var6= 'and';
14    var $_var7= 'I';
15    /**
16     * Two words
17     */
18    var $_var8= 'like strings';
19    /**#@-*/
20    var $publicvar= 'Lookee me!';
21 }
```

This example will parse as if it were:

```
1 class Bob
2 {
3     // beginning of docblock template area
4     /**
5      * @access private
6      * @var string
7      */
8     var $_var1= 'hello';
9     /**
10      * @access private
11      * @var string
12      */
13     var $_var2= 'my';
14     /**
15      * @access private
16      * @var string
17      */
18     var $_var3= 'name';
19     /**
20      * @access private
21      * @var string
22      */
23     var $_var4= 'is';
24     /**
25      * @access private
26      * @var string
```

```

27     */
28 var $_var5= 'Bob';
29 /**
30 * @access private
31 * @var string
32 */
33 var $_var6= 'and';
34 /**
35 * @access private
36 * @var string
37 */
38 var $_var7= 'I';
39 /**
40 * Two words
41 * @access private
42 * @var string
43 */
44 var $_var8= 'like strings';
45 var $publicvar= 'Lookeee me!';
46 }

```

Note that for \$\_var8 the DocBlock template merged with the DocBlock. The rules for merging are simple:

- The long description of the docblock template is added to the front of the long description. The short description is ignored.
- All tags are merged from the docblock template

## Tags

Tags are single words prefixed by a "@" symbol. Tags inform phpDocumentor how to present information and modify display of documentation. All tags are optional, but if you use a tag, they do have specific requirements to parse properly.

A list of all possible tags in phpDocumentor follows:

```

1 /**
2 * The short description
3 *
4 * As many lines of extendend description as you want {@link element}
5 * links to an element
6 * {@link http://www.example.com Example hyperlink inline link} links to
7 * a website. The inline
8 * source tag displays function source code in the description:
9 * {@source }
10 *
11 * In addition, in version 1.2+ one can link to extended documentation like this
12 * documentation using {@tutorial phpDocumentor/phpDocumentor.howto.pkg}
13 * In a method/class var, {@inheritdoc} may be used to copy documentation from
14 * the parent method
15 * {@internal}
16 * This paragraph explains very detailed information that will only
17 * be of use to advanced developers, and can contain

```

```

18 * {@link http://www.example.com Other inline links!} as well as text}}
19 *
20 * Here are the tags:
21 *
22 * @abstract
23 * @access public or private
24 * @author author name <author@email>
25 * @copyright name date
26 * @deprecated description
27 * @deprec alias for deprecated
28 * @example /path/to/example
29 * @exception Javadoc-compatible, use as needed
30 * @global type $globalvarname
31 or
32 * @global type description of global variable usage in a function
33 * @ignore
34 * @internal private information for advanced developers only
35 * @param type [$varname] description
36 * @return type description
37 * @link URL
38 * @name procpagealias
39 or
40 * @name $globalvaralias
41 * @magic phpdoc.de compatibility
42 * @package package name
43 * @see name of another element that can be documented,
44 * produces a link to it in the documentation
45 * @since a version or a date
46 * @static
47 * @staticvar type description of static variable usage in a function
48 * @subpackage sub package name, groupings inside of a project
49 * @throws Javadoc-compatible, use as needed
50 * @todo phpdoc.de compatibility
51 * @var type a data type for a class variable
52 * @version version
53 */
54 function if_there_is_an_inline_source_tag_this_must_be_a_function()
55 {
56 //...
57 }

```

In addition, tutorials allow two addition inline tags: {@id}, used to allow direct linking to sections in a tutorial, and {@toc}, used to generate a table of contents from {@id}s in the file. Think of {@id} like an [HTML tag](#), it serves the same function.

See [inline {@id}](#) and [inline {@toc}](#) for detailed information

In the example below, {@id} is used to name the refsect1 "mysection" and the refsect2 "mysection.mysubsection" - note that the sub-sections inherit the parent section's id.

### **Example:**

```
role = "xml"<refentry id="@id">
<refsect1 id="@id mysection">
<refsect2 id="@id mysubsection">
</refsect2>
</refsect1>
</refentry>
```

For an in-depth look at phpDocumentor tags, read [phpDocumentor tags](#), and for an in-depth look at inline tags, read [phpDocumentor Inline tags](#).

## Documenting your PHP project

### Where to begin

Before you begin documenting your PHP project, you might want to try a test run on your undocumented source code to see what kind of information phpDocumentor will automatically extract from the source code. phpDocumentor is designed to make the task of documenting minimally redundant. This means less work, and better, up-to-date documentation with less effort than it used to take.

#### Caution

phpDocumentor is a great tool, but it will not write good documentation for you. Please read the [phpDocumentor Guide to Creating Fantastic Documentation](#)

### Elements of the source code that can be documented

#### *Dividing projects into packages*

To understand the role of packages and how to use [@package](#), it is important to know the logic behind packaging in PHP. The quest for structured programming led to the invention of functions, then classes, and finally packages. Traditionally, a re-usable software module was a collection of variables, constants and functions that could be used by another software package. PHP is an example of this model, as there are many extensions that consist of constants and functions like the tokenizer extension. One can think of the tokenizer extension as a package: it is a complete set of data, variables and functions that can be used in other programs. A more structured format of this model is of course objects, or classes. A class contains variables and functions (but no constants in PHP). A single class packages together related functions and variables to be re-used.

phpDocumentor defines package in two ways:

1. Functions, Constants and Global Variables are grouped into files (by the filesystem), which are in turn grouped into packages using the @package tag in a page-level DocBlock
2. Methods and Class Variables are grouped into classes (by PH),

which are in turn grouped into packages in a Class DocBlock

These two definitions of package are exclusive. In other words, it is possible to have classes of a different package of the file that contains it! Here's an example:

## Caution

It may be possible, but don't put classes into a different package from the file they reside in, that will be very confusing and unnecessary. This behavior is deprecated, in version 2.0, phpDocumentor will halt parsing upon this condition.

```
1  <?php
2  /**
3  * Pretend this is a file
4  *
5  * Page-level DocBlock is here because it is the first DocBlock
6  * in the file, and is immediately followed by the second
7  * DocBlock before any documentable element is declared
8  * (function, define, class, global variable, include)
9  * @package pagepackage
10 */
11 /**
12 * Here is the class DocBlock
13 *
14 * The class package is different from the page package!
15 * @package classpackage
16 */
17 class myclass
18 {
19 }
20 ?>
```

For more information on page-level versus class-level packaging, see [Procedural Elements](#)

Perhaps the best way to organize packages is to put all procedural code into separate files from classes. [PEAR](#) recommends putting every class into a separate file. For small, utility classes, this may not be the best solution for all cases, but it is still best to separate packages into different files for consistency.

## Advanced phpDocumentor: tutorials and extended Documentation

phpDocumentor developers have received a number of requests to allow linking to external documentation, and to sections of that documentation. If phpDocumentor only could create HTML documents, this would be a simple task, but the presence of PDF and now XML converters as well as future possibilities complicates this question tremendously. What is the solution?

Give phpDocumentor the ability to parse external documentation in a common format and then convert it to the appropriate format for each converter. The implementation of this solution in version 1.2.0 is very versatile. Making use of the standard DocBook XML format, external documentation can be designed and then reformatted for any output. No longer is external documentation tied down to one "look." Here's a short

list of the benefits of this approach:

Benefits of using DocBook as the format:

- DocBook is very similar to HTML at the basic level and very easy to learn.
- Each template has its own options.ini file which determines how the DocBook tags will be translated into the output language - no need to learn xslt.
- Adding in xslt support will be very easy to allow for future customization

Benefits of integrating tutorials/external documentation into phpDocumentor:

- Linking to external documentation from within API docs is possible
- Linking to API docs from external documentation is also possible
- Customizable table of contents, both of all tutorials and within a tutorial via [inline {@toc}](#)
- It is possible to create User-level documentation that has direct access to Programmer-level documentation

User-level documentation generally consists of tutorials and information on how to use a package, and avoids extreme detail. On the other hand, programmer-level documentation has all the details a programmer needs to extend and modify a package. phpDocumentor has been assisting the creation of programmer-level documentation since its inception. With the addition of tutorials, it can now ease the creation of user-level documentation.

For an in-depth look at how to use tutorials, read [phpDocumentor Tutorials](#)

## Running phpDocumentor

There are two bundled ways of invoking phpDocumentor, the command-line phpdoc, or the web interface phpdoc.php/new\_phpdoc.php.

## Using the new Web Interface docbuilder

The new web interface requires a working installation of PHP with a web server, and must be accessible from the document root. Docbuilder is the creation of Andrew Eddies, and it combines some slick formatting with the functionality of the old web interface. The docbuilder interface can be accessed via index.html in the install directory of phpDocumentor, or the docbuilder subdirectory.

## Using the old Web Interface [phpdoc.php](#) or [new\\_phpdoc.php](#)

In order to use the web interface, there are a number of factors that must be set up before anything else can occur. First, you need a working web server with php (it had to be said). This manual will not assist with that setup. Next, the phpdoc.php or new\_phpdoc.php file needs to be accessible by the webserver. In unix, place a symbolic link using ln -s to the desired interface into the public

html directory.

## Caution

Security is always an issue with the internet. Do not place phpDocumentor into the web server publicly available path on a server connected to the internet. Make sure that phpDocumentor will not have the power to overwrite **ANY** system or user files.

Note that since the webserver runs as user nobody in unix, the generated files will be owned by nobody. The only way to change this is to either run phpDocumentor from the command-line or to add a chuser wrapper around httpd. **We do not recommend using a chuser wrapper or running phpDocumentor as root.** It is much easier and safer to use a config file (see [phpDocumentor's dynamic User-defined config files](#)) from the command line.

## Using the Command-line tool

### *Running the command-line in MS Windows*

Running phpDocumentor from the command-line is fairly straightforward, even in windows. It is recommended to use the web interface in windows, from a non-publicly accessible server root, as none of the permissions issues exist that exist in unix. However, to use phpDocumentor from the command line is possible. First, put the location of php into the system path, then type:

```
C:\>php-cli C:\path\to\phpdoc\phpdoc [commandline]
```

An alternative is to edit the phpdoc.bat file, and place phpdoc in the path, then you can run phpdoc as a normal command.

### *Running the command-line in unix*

Running the command-line tool phpdoc is very easy in unix:

```
.\phpdoc [commandline]
```

*-c, --config*

Use this option to load a config file (see [phpDocumentor's dynamic User-defined config files](#))

"phpdoc -c default" will load the default.ini file

*-cp, --converterparams*

This option is only used to pass dynamic parameters to extended converters. The options passed should be separated by commas, and are placed in the global variable `$_phpDocumentor_setting['converterparams']`, an array. It is the responsibility of the Converter to access this variable, the code included with phpDocumentor does nothing with it.

*-ct, --customtags*

Use this option to specify tags that should be included in the list of valid tags for the current run of phpdoc

"phpdoc -ct mytag,anotherTag" will tell phpDocumentor to parse this DocBlock:

```
1  /**
2  * @mytag this is my tag
3  * @anotherTag this is another tag
4  */
```

without raising any errors, and include the tags in the known tags list for the template to handle. Note that in version 1.2.0+, the unknown\_tags array is passed to templates.

*-dh, --hidden*

Use this option to tell phpDocumentor to parse files and directories that begin with a period (.)

*-dc, --defaultcategoryname*

This will tell phpDocumentor to group any uncategorized elements into the primary categoryname. If not specified, the primary category is "default." The category should be specified using the [@category](#) tag.

```
1  /**
2  * This package has no category and will be grouped into the default
3  * category unless -dc categoryname is used
4  * @package mypackage
5  */
6  class nocategory
7  {
8  }
```

*-dn, --defaultpackagename*

Use this option to tell phpDocumentor what the primary package's name is. This will also tell phpDocumentor to group any unpackaged elements into the primary packagename. If not specified, the primary package is "default"

```
1  /**
2  * This class has no package and will be grouped into the default
3  * package unless -dn packagename is used
4  */
5  class nopackage
6  {
7  }
```

*-d, --directory*

This or the -f option must be present, either on the command-line or in the config file.

Unlike -f, -d does not accept wildcards. Use -d to specify full paths or relative paths to the current directory that phpDocumentor should recursively search for documentable files. phpDocumentor will search through all subdirectories of any directory in the command-line list for tutorials/ and any files that have valid .php extensions as defined in phpDocumentor.ini. For example:

"phpdoc -d relative/path/to/dir1,/fullpath/to/here,.."

*-ed, --examplesdir*

The -ed option is used to specify the full path to a directory that example files are located in. This command-line setting affects the output of the [@example](#) tag.

"phpdoc -ed /fullpath/to/examples"

*-f, --filename*

This or the -d option must be present, either on the command-line or in the config file.

This option is used to specify individual files or expressions with wildcards \* and ?. Use \* to match any string, and ? to match any single character.

- phpdoc -f m\*.p\* will match myfile.php, mary\_who.isthat.phtml, etc.
- phpdoc -f /path/m\* will match /path/my/files/here.php, /path/mommy.php /path/mucho/grande/what.doc, etc.
- phpdoc -f file?.php will match file1.php, file2.php and will not match file10.php

*-i, --ignore*

Use the -i option to exclude files and directories from parsing. The -i option recognizes the \* and ? wildcards, like -f does. In addition, it is possible to ignore a subdirectory of any directory using "dirname/".

- phpdoc -i tests/ will ignore /path/to/here/tests/\* and /path/tests/\*
- phpdoc -i \*.inc will ignore all .inc files
- phpdoc -i \*path/to/\* will ignore /path/path/to/my/\* as well as /path/to/\*
- phpdoc -i \*test\* will ignore /path/tests/\* and /path/here/my\_test.php

Since v1.3.2, the value or pattern you provide will be case-sensitive (OS permitting, anyway), and will be applied relative to the top-level directory in the -d argument.

- phpdoc -i CVS/ will ignore /path/to/CVS/\* but will not ignore /path/to/cvs
- phpdoc -d /home/myhome/cvs/myproject -i cvs/ will ignore /home/myhome/cvs/myproject/files/cvs/\* but will not ignore /home/myhome/cvs/\*

*-is, --ignoresymlinks*

Use the -is option to explicitly ignore any symlinks, whether they be links to directories or links to files.

This only works on a Unix/Linux environment, since Windows doesn't have symlinks.

*-it, --ignore-tags*

Use the `-it` option to exclude specific tags from output. This is used for creating multiple sets of documentation for different audiences. For instance, to generate documentation for end-users, it may not be desired to output all `@todo` tags, so `--ignore-tags @todo` would be used. To ignore inline tags, surround them with brackets {} like `--ignore-tags {@internal}`.

`--ignore-tags` will not ignore the core tags `@global`, `@access`, `@package`, `@ignore`, `@name`, `@param`, `@return`, `@staticvar` or `@var`.

## Caution

The `--ignore-tags` option requires a full tag name like `--ignore-tags @todo`. `--ignore-tags todo` will not work.

*-j, --javadocdesc*

Use this command-line option to enforce strict compliance with JavaDoc. JavaDoc DocBlocks are much less flexible than `phpDocumentor`'s DocBlocks (see [DocBlocks](#) for information).

*-o, --output*

Use this required option to tell `phpDocumentor` which Converters and templates to use. In this release, there are several choices:

- `HTML:frames:*` - output is HTML with frames.
  - `HTML:frames:default` - JavaDoc-like template, very plain, minimal formatting
  - `HTML:frames:earthli` - BEAUTIFUL template written by Marco von Ballmoos
  - `HTML:frames:l0l33t` - Stylish template
  - `HTML:frames:phpdoc.de` - Similar to `phpdoc.de`'s PHPDoc output
  - `HTML:frames:phptmlib` - Very nice user-contributed template
  - all of the templates listed above are also available with javascript expandable indexes, as `HTML:frames:DOM/name` where name is default, `l0l33t`, `phpdoc.de`, etcetera
  - `HTML:frames:phpedit` - Based on output from [PHPEdit](#) Help Generator
- `HTML:Smarty:*` - output is HTML with no frames.
  - `HTML:Smarty:default` - Bold template design using css to control layout
  - `HTML:Smarty:HandS` - Layout is based on PHP, but more refined, with logo image
  - `HTML:Smarty:PHP` - Layout is identical to the PHP website
- `CHM:default:*` - output is CHM, compiled help file format (Windows help).
  - `CHM:default:default` - Windows help file, based on `HTML:frames:l0l33t`
- `PDF:default:*` - output is PDF, Adobe Acrobat format
  - `PDF:default:default` - standard, plain PDF formatting
- `XML:DocBook:*` - output is XML, in DocBook format
  - `XML:DocBook/peardoc2:default` - documentation ready for compiling into `peardoc` for online `pear.php.net` documentation, 2nd revision

In 1.2.0+, it is possible to generate output for several converters and/or templates at once. Simply specify a comma-delimited list of output:converter:template like:

```
phpdoc -o HTML:frames:default,HTML:Smarty:PHP,HTML:frames:phpedit,PDF:default:default -t ./docs -d .  
-pp, --parseprivate
```

By default, phpDocumentor does not create documentation for any elements marked [@access](#) private. This allows creation of end-user API docs that filter out low-level information that will not be useful to most developers using your code. To create complete documentation of all elements, use this command-line option to turn on parsing of elements marked private with [@access](#) private.

```
-po, --packageoutput
```

Use this option to remove all elements grouped by [@package](#) tags in a comma-delimited list from output. This option is useful for projects that are not organized into separate files for each package. Parsing speed is not affected by this option, use [-i, --ignore](#) whenever possible instead of --packageoutput.

```
phpdoc -o HTML:frames:default -t ./docs -d . -po mypackage,thatotherpackage will only display  
documentation for elements marked with "@package mypackage" or "@package thatotherpackage"
```

```
-p, --pear
```

Use this option to parse a [PEAR](#)-style repository. phpDocumentor will do several "smart" things to make life easier:

- PEAR-style destructors are automatically parsed
- If no [@package](#) tag is present in a file or class DocBlock, it will guess the package based on subdirectory
- If a variable or method name begins with `_`, it will be assumed to be [@access](#) private, unless an [@access](#) tag is present. Constructors are also assumed to be [@access](#) private in the current version.

Warnings will be raised for every case above, as [@package](#) should always be explicit, as should [@access](#) private. We do not like to assume that phpDocumentor knows better than you what to do with your code, and so will always require explicit usage or raise warnings if phpDocumentor does make any assumptions.

```
-q, --quiet
```

This option tells phpDocumentor to suppress output of parsing/conversion information to the console. Use this for cron jobs or other situations where human intervention will not be needed.

```
-ric, --readmeinstallchangelog
```

This comma-separated list of files specifies files that should be parsed and included in the documentation. phpDocumentor will automatically grab files listed here if and only if they are located in the top-level directory that is parsed. In other words, if you parse these files:

- /path/to/files/file1.php  
/path/to/files/file2.php  
/path/to/files/README  
/path/to/files/subdir/file2.php  
/path/to/files/subdir/README

```
-s, --sourcecode
```

This option tells phpDocumentor to generate highlighted cross-referenced source code for every file parsed. The highlighting code is somewhat unstable, so use this option at your own risk.

*-t, --target*

Use this to tell phpDocumentor where to put generated documentation. Legal values are paths that phpDocumentor will have write access and directory creation privileges.

*-tb, --templatebase*

-tb accepts a single pathname as its parameter. The default value of -tb if unspecified is <phpDocumentor install directory>/phpDocumentor. This raises a very important point to understand. The -tb command-line is designed to tell phpDocumentor to look for **all** templates for all converters in subdirectories of the path specified. By default, templates are located in a special subdirectory structure.

For the hypothetical outputformat:convertername:templatename [-o, --output](#) argument, the directory structure is Converters/outputformat/convertname/templates/templatename/. The class file for convertname should be in the convertname/templates directly, right beside your templatename directory/ies. In addition, Smarty expects to find two subdirectories, templates/ (containing the Smarty template files) and templates\_c/ (which will contain the compiled template files).

"phpdoc -tb ~/phpdoctemplates -o HTML:frames:default" will only work if the directory ~/phpdoctemplates/Converters/HTML/frames/default/templates contains all the template files, and ~/phpdoctemplates/Converters/HTML/frames/default/templates\_c exists.

*-ti, --title*

Set the global title of the generated documentation

*-ue, --undocumentedelements*

This option tells phpDocumentor whether or not to suppress warnings about certain objects (classes, methods) that are not documented via a DocBlock comment. Use this to help identify objects that you still need to write documentation for.

## **phpDocumentor's dynamic User-defined config files**

The new [-c, --config](#) command-line options heralds a new era of ease in using phpDocumentor. What used to require:

```
phpdoc -t /path/to/output -d path/to/directory1,/another/path,/third/path\  
-f /path/to/anotherfile.php -i *test.php,tests/ -pp on -ti My Title -o HTML:frames:phpedit
```

Can now be done in a single stroke!

```
phpdoc -c myconfig
```

What makes this all possible is the use of config files that contain all of the command-line information. There are two configuration files included in the distribution of phpDocumentor, and both are in the user/ subdirectory. To use a configuration file "myconfig.ini," simply place it in the user directory, and the command-line "phpdoc -c myconfig" will tell phpDocumentor to read all the command-line settings from

that config file. Configuration files may also be placed in another directory, just specify the full path to the configuration file:

```
phpdoc -c /full/path/to/myconfig.ini
```

The best way to ensure your config file matches the format expected by phpDocumentor is to copy the default.ini config file, and modify it to match your needs. Lines that begin with a semi-colon (;) are ignored, and can be used for comments.

Be sure to put in your config file all the runtime options you need, because **all other command-line arguments are ignored** if you use a config file.

## phpDocumentor.ini - global configuration options

The phpDocumentor.ini file contains several useful options, most of which will never need to be changed. However, some are useful. The section [\_phpDocumentor\_options] contains several configuration settings that may be changed to enhance output documentation. For Program Location, the relative path is specified as Program\_Root/relativepath/to/file. The prefix "Program\_Root" may be changed to any text, including none at all, or "Include ", etc. The [\_phpDocumentor\_setting] section can be used to specify a config file that should be used by default, so phpDocumentor may be run without any command-line arguments whatsoever! The [\_phpDocumentor\_phpfile\_exts] section tells phpDocumentor which file extensions are php files, add any non-standard extensions, such as "class" to this section.

# Documentable PHP Elements

*Elements of PHP source that phpDocumentor can automatically document*

**Authors:** Joshua Eichorn

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

Gregory Beaver

[celflog@php.net](mailto:celflog@php.net)

## Introduction to Documentable Elements

phpDocumentor is capable of automatically documenting include statements, define statements, functions, procedural pages, classes, class variables, and class methods.

## Procedural Elements

From phpDocumentor's perspective, the basic container for procedural elements (as in real life) is the file that contains them. To reflect this, it is possible to document information about the entire contents of a file. This is accomplished through the use of a page-level DocBlock (see [DocBlocks](#) for basic information on what a DocBlock is). A page-level DocBlock is the only DocBlock that cannot precede the element that it is documenting, as there is no way to precede a file. To solve this issue, the way phpDocumentor finds a page-level DocBlock is to parse the first DocBlock in a file as the page-level DocBlock, with certain conditions.

```
1  <?php
2  /**
3  * Page-level DocBlock
4  */
5  define('oops','Page-level DocBlock it is not!');
6  ?>
```

This last example has one DocBlock, and it is the first DocBlock in a file, but it is not a Page-level DocBlock. How can phpDocumentor tell the difference between a Page-level DocBlock and any other DocBlock? Simple:

```
1  <?php
2  /**
3  * Pretend this is a file
4  *
5  * Page-level DocBlock is here because it is the first DocBlock
6  * in the file, and contains a @package tag
7  * @package pagepackage
8  */
9  define("almost" , "Now the Page-level DocBlock is for the page, and the Define has
no docblock" );?>
```

In phpDocumentor version 1.2.2, a Page-level DocBlock is the first DocBlock in a file if it contains a [@package](#) tag. However, this example will raise a warning like WARNING in test.php on line 8: Page-level DocBlock precedes "define almost", use another DocBlock to document the source element. You can eliminate the warning by adding documentation to the define as follows:

```
1  <?php
2  /**
3   * Page-level DocBlock
4   * @package pagepackage
5   */
6  /**
7   * Define DocBlock
8   */
9  define("ahhhh", "Now the Page-level DocBlock is for the page, and the Define
10 DocBlock for the define");
11 ?>
```

Now, the page has its documentation, and the define has its own documentation.

So, a DocBlock is a page-level DocBlock IF AND ONLY IF it is both:

1. The first DocBlock in a file
2. One of:
3. Contains a [@package](#) tag
4. Immediately followed by another DocBlock for any documentable PHP element **this is deprecated, always use a @package tag**

A Page-level DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tags:

- [@package](#)
- [@subpackage](#)

## Caution

phpDocumentor will not document a file like the first example, there must be at least one documentable PHP element in the file.

## include/include\_once/include\_once/include statements

phpDocumentor extracts the filename and attempts to link to documentation for that filename if possible. Include statements may only have any of the [Standard phpDocumentor Tags](#)

phpDocumentor will attempt to locate the included file in the list of files parsed, and if found will make a link to that file's documentation.

## define statements

A define statement's DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tag:

- [@name](#)

## function declarations

A function's DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tags:

- [@global](#)
- [@param](#)
- [@return](#)
- [@staticvar](#)
- [inline {@source}](#)

## global variables

A global variable's DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tag:

- [@name](#)

In addition, the global variable's DocBlock **must** contain the [@global](#) tag.

## Class Elements

A class's DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tags:

- [@package](#)
- [@subpackage](#)
- [@static](#)

## DocBlock inheritance

New in version 1.2.0, DocBlock's are inherited by child classes, variables, and methods. There are a few simple rules for inheritance:

- tags [@author](#), [@version](#), and [@copyright](#) are automatically inherited unless explicitly specified in the DocBlock
- As in previous versions, [@package](#) and [@subpackage](#) are inherited unless explicitly specified in the DocBlock. We recommend that you explicitly use @package and @subpackage for every class to avoid problems with name conflicts that may arise
- If there is no short description, the short description will be inherited.
- If there is no long description, the long description will be inherited.
- If there is a long description, and you still want to inherit the parent's description, use [inline {@inheritDoc}](#)

```

1  /**
2  * short desc
3  *
4  * long desc
5  * @package test
6  * @author me
7  * @version 1.0
8  * @abstract
9  * @copyright never
10 */
11 class parclass
12 {
13 }
14
15 // inherits entire DocBlock minus @abstract
16 class child1 extends parclass
17 {
18 }
19
20 // inherits DocBlock minus @abstract, short desc
21 /**
22 * overriding short desc
23 */
24 class child2 extends parclass
25 {
26 }
27
28 // inherits @author, @version, @copyright, @package
29 /**
30 * overriding short desc
31 *
32 * overriding long desc
33 */
34 class child3 extends parclass
35 {
36 }
37
38 // inherits @version, @copyright, @package
39 /**
40 * overriding short desc
41 *
42 * overriding long desc
43 * @author you
44 */
45 class child4 extends parclass
46 {
47 }

```

## class variables

A class variable's DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tag:

- [@var](#)

## class methods

A method's DocBlock may have any of the standard phpDocumentor Tags (see [Standard phpDocumentor Tags](#)) plus the following tags:

- [@global](#)
- [@param](#)
- [@return](#)
- [@static](#)
- [@staticvar](#)
- [inline {@source}](#)

# phpDocumentor Tutorials

*Writing User-level documentation/tutorials with phpDocumentor*

## What are phpDocumentor-style tutorials/extended documentation?

New in version 1.2.0+, tutorials are the missing link in auto-documentation. We've had many requests from developers to be able to include links to external documentation, such as the reference of template tags for a template engine. The source code would be harder to read if this kind of information was spread along the 150-300 lines it would take to describe the usage. phpDocumentor Tutorials solve the problem of including external documentation by leveraging existing tried-and-true XML DocBook format and the power of phpDocumentor's inline tags (see [phpDocumentor Inline tags](#)). Now, it is possible to cross-reference between external documentation and generated API documentation through the versatile [inline {@link}](#), [inline {@tutorial}](#), and [{@tutorial}](#) tag.

## How to write phpDocumentor-style tutorials/extended documentation

Tutorials may be written as any legal DocBook top-level tag (`<book>`, `<chapter>`, `<article>`, `<refentry>`), but it is **highly** recommended for any project that may become a part of PEAR to write with the `<refentry>` tag as the top-level, as this will automatically translate into peardoc2-ready tutorials. Other projects have no such constraint and may do anything they wish.

### Inline Tags that may be used in tutorials

There are several inline tags that may be used in tutorials. The [inline {@link}](#) and [inline {@tutorial}](#) tags should be used in the same manner as in API source. Inline tags that are specific to tutorials include [inline {@id}](#) and [inline {@toc}](#). [Inline {@id}](#) is used to mark areas in a similar manner to the way that `<a name="section">` in HTML. [{@id}](#) will automatically generate an id tag appropriate for the output format. The [{@toc}](#) tag is used at the top of a tutorial to generate an internal table of contents for that tutorial. The [{@toc}](#) tag extracts all sections that have an [{@id}](#) tag and uses the title of that section for the generated table of contents. [{@id}](#) is also used to mark subsections to allow linking to a sub-section of a tutorial, much the way that in HTML file.html#subsection is used.

### Where to put tutorials/extended documentation so phpDocumentor will find it

To simplify linking of tutorials, all tutorials must be in a subdirectory named "tutorials" and must be passed in a [-d, --directory](#) or [-f, --filename](#) command-line switch.

In addition, tutorials are grouped by the same packages that group documented elements (see [Dividing projects into packages](#) for more information). To associate a tutorial with a package, place it in a subdirectory of tutorials/ named after the package. A tutorial for package "phpDocumentor" goes in "tutorials/phpDocumentor," a tutorial for subpackage "CHMDefault" of package "Converters" goes in "tutorials/Converters/CHMdefault/" - for an example, see the tutorials/ subdirectory of phpDocumentor.

At the bare minimum, assuming your one package is named "MyApp", you should create the "tutorials/MyApp/MyApp.pkg" file, with this minimal content:

```
1 <refentry id="@id"      >
2   <refnamediv>
3     <refname> User Guide for My Application</refname>
4     <refpurpose> To demonstrate the various tasks available in My app</refpurpose>
5   </refnamediv>
6   <refsynopsisdiv>
7     <author>
8       My Name
9     <authorblurb>
10    {@link mailto:myself@myhost.net My Name}
11  </authorblurb>
12 </author>
13 </refsynopsisdiv>
14 {@toc}
15 <refsect1 id="@id intro"      >
16   <title> User Guide</title>
17   <para>
18     This will be the User Guide for My app.
19   </para>
20 </refsect1>
21 </refentry>
```

One common "gotcha" here is ensuring that you name the PKG file and its parent directory with your package's exact name, as well as ensuring you have indeed used the "@package MyApp" tag in your codebase. Running PhpDocumentor against this minimal tutorial file, when no PHP files use this package name, will NOT result in your tutorial being generated.

A second common "gotcha" is that if you attempt to run PhpDocumentor against ONLY the tutorial directory and its files, you'll get "ERROR: nothing parsed"... because you did not give PhpDocumentor any PHP files to process. PhpDocumentor does need at least one "code" file to process, along with the tutorials, in order for any tutorials to be generated. This is a common mistake when first trying to write tutorial docs... writing the files and then trying to run PhpDocumentor against ONLY the tutorials directory to test your tutorial files. You will need at least one PHP file included in your test run, to avoid getting "ERROR: nothing parsed".

A third "gotcha" can occur when you make your tutorial for your package and expect that the top-level index.html file will show its content, only to see an empty page showing little more than "Welcome to default!". What has happened is that the default package name for a documentation generation is "default", so unless you specify "--defaultpackagename MyApp", then your resulting top-level index.html file wants to tie itself to a "default.pkg" tutorial file for the "default" package, which does not exist, and it therefore decides there is no tutorial and instead displays that generic "Welcome" message.

## Naming tutorials: .pkg, .cls, .proc

There are three ways of associating tutorials/extended documentation with the elements they are written for:

- Package-level tutorials always have file extension "pkg"
- Class-level tutorials always have file extension "cls"
- Procedural-level tutorials always have file extension "proc"

### *Package-level documentation*

The primary documentation for a package or subpackage should go in a file with the same name as the package or subpackage. For package "phpDocumentor" the primary package-level documentation is in "tutorials/phpDocumentor/phpDocumentor.pkg". For subpackage "CHMdefault" of package "Converters," the primary subpackage-level documentation is in "tutorials/Converters/CHMDefault/CHMdefault.pkg"

Other package-level documentation can have any filename, but must have the package-tutorial extension .pkg.

### *Class-level documentation*

Similar to package-level documentation, the primary documentation for a class must have the same name as the class. The primary documentation for the class "Converter" in package "Converters" will be found in "tutorials/Converters/Converter.cls."

### *Procedural-level documentation*

If a package contains a number of functions, it may be good to include procedural-level documentation. If a particular file contains a large amount of procedural information that must be documented, it is possible to link tutorials to that file's API docs in the same manner it is possible to link to a class's API docs.

The primary documentation for the procedural component of a PHP file must have the same name as the file. The primary procedural documentation for the file test.php in package "examples" will be found in "tutorials/examples/test.php.proc."

## **Creating a tutorial hierarchy**

If you've ever wanted to separate a large user-level document into separate documents the way the PHP manual does, you're in luck. phpDocumentor provides a dynamic and flexible solution to this problem. When parsing a tutorial, phpDocumentor looks for an .ini file associated with that tutorial in the same directory. In other words, for "tutorials/phpDocumentor/phpDocumentor.pkg," phpDocumentor will search for a file named "tutorials/phpDocumentor/phpDocumentor.pkg.ini" and will parse out a section named [Linked Tutorials]. This section defines every child tutorial of the tutorial phpDocumentor.pkg

In order for a tutorial to be linked to another tutorial as a child, it must be in the same package, subpackage and be the same tutorial type. tutorials/phpDocumentor/phpDocumentor.pkg cannot have tutorials/phpDocumentor/oops.cls as a child tutorial, nor can it have tutorials/phpDocumentor/oops/almostworks.pkg as a child tutorial. It can have tutorials/phpDocumentor/works.pkg as a child, because the tutorial is in the same package, and is package-level documentation.

To help enforce this rule, the [Linked Tutorials] section is a list of filenames minus path and extension.

Here's an example tutorials/test/test.pkg.ini:

**Example:**

```
role = "ini"[Linked Tutorials]
```

```
child1  
child2  
child3
```

phpDocumentor will search for tutorials/test/child1.pkg, tutorials/test/child2.pkg, and tutorials/test/child3.pkg and link them in that order. Note that any tutorial can have an .ini file, allowing unlimited depth of tutorials. We don't recommend going beyond 3 sub-levels, that will probably confuse readers more than it helps them.

For a working example, check out the .ini files in phpDocumentor's own tutorials/ directory.

## Converting tutorials/extended documentation to HTML/CHM/PDF/XML/...

Support is built into the existing Converters for automatic conversion of tutorials to the desired output without any extra work on your part. However, if the output is not quite right, and something needs to be changed, this is very easy. For every template of every converter, there is a file in the template directory called "options.ini." In other words, for the HTML:frames:default template, the options.ini file is phpDocumentor/Converters/HTML/frames/templates/default/options.ini. Open this file to see all of the template options for conversion of data.

The first section of the .ini file, [desctranslate], is for conversion of DocBlocks and is covered elsewhere (see [DocBlock Description details](#)).

After that comes the section that deals with tutorials, [ppage].

### [ppage] section of options.ini

The [ppage] section contains simple rules for transforming DocBook tags and DocBook entities into any output. Re-ordering is supported for attributes and titles only. The best way to learn how to use this simple and powerful transformation is to study the options.ini files for HTML:frames:phpedit and PDF:default:default

#### *DocBook Entity translation*

The translation of entities like &rdquo; is handled through entries in the [ppage] section like the following for html:

```
role = "ini"[ppage]  
" = "  
& = &  
&rdquo; = &rdquo;
```

or the following, for PDF:

```
role = "ini"[ppage]
" = """
& = &
&rdquo; = """
```

Note that to specify a ", it must be enclosed in double quotes, hence the usage of """

### *DocBook Tag translation*

Each tag is transformed using a few possibilities. If a tag is to be simply transformed into another tag (as in `<p></p>` to `<para></para>`, use

- `tagname = newtagname`

`phpDocumentor` will automatically enclose the tag contents with the new tagname and all of its attributes

If the start and endtag should be different, specify the exact text using a slash before the tagname as in:

- `tagname = <starttext attr="myval" />`
- `/tagname = "\n"`

If a tag needs only a new tag and attribute name (for example link and linkend become a and href):

- `tagname = newtagname`
- `tagname->attr = newattrname`

If a tag needs only a new tag and attribute name, and a new attribute value (for example table frame="all" becomes table frame="border"):

- `tagname = newtagname`
- `tagname->attr = newattrname`
- `tagname->attr+val = newval`

If a tag needs only a new tag and attribute name, and two attributes combine into one (for example table colsep="1" rowsep="1" becomes table rules="all"):

- `tagname = newtagname`
- `tagname->attr = newattrname`
- `tagname->attr2 = newattrname`
- `tagname->attr+val|attr2+val = newval`

If an attribute should be ignored:

- `tagname = newtagname`
- `tagname!attr =`

If all attributes should be ignored:

- tagname = newtagname
- tagname! =

If a tag should have all attributes and be a single tag (like <br />):

- tagname = newtagname
- tagname/ =

If an attribute should be changed to a new name for all cases (like role changed to class, which would be \$attr\$role = class):

- \$attr\$attrname = newattrname

## Tips for using the .ini files

The parsing of these ini files is performed by [phpDocumentor\\_parse\\_ini\\_file\(\)](#), a much more powerful version of PHP's [http://www.php.net/parse\\_ini\\_file](http://www.php.net/parse_ini_file) built-in function. If a value is encased in quotation marks, it will be stripped of backslashes (\) and used as is. In other words:

### Example:

```
role = "ini"tagname = " Some "enclosed"\n\ttext"
```

will parse out as tagname = Some "enclosed" text.

Otherwise, this example:

### Example:

```
role = "ini"tagname = Some "enclosed"\n\ttext
```

will parse out as tagname = Some "enclosed"\n\ttext.

Note that escaping of quotation marks is not needed, `phpDocumentor_parse_ini()` only strips the outer tags and then performs <http://www.php.net/stripcslashes> on the remaining string.

## What next?

Write your tutorials and extended documentation!

# phpDocumentor Manual

## *Introduction to phpDocumentor*

**Authors:** *Gregory Beaver*

[cellog@php.net](mailto:cellog@php.net)

*Joshua Eichorn*

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

*Kellin*

[passionplay@hotmail.com](mailto:passionplay@hotmail.com)

*Juan Pablo Morales*

[ju-moral@uniandes.edu.co](mailto:ju-moral@uniandes.edu.co)

*Florian Clever*

[fclever@verinform.com](mailto:fclever@verinform.com)

*Dennis Kehrig*

[TheGeek@web.de](mailto:TheGeek@web.de)

*Robert Hoffmann*

[half-dead@users.sourceforge.net](mailto:half-dead@users.sourceforge.net)

*Roberto Berto*

[darkelder@users.sourceforge.net](mailto:darkelder@users.sourceforge.net)

*Dave Viner*

[dviner@yahoo-inc.com](mailto:dviner@yahoo-inc.com)

*Walter A Boring IV*

[hemna@users.sourceforge.net](mailto:hemna@users.sourceforge.net)

*Mark van Reunswolde*

[support@esdsoftware.net](mailto:support@esdsoftware.net)

*Julien Damon*

[julien.damon@free.fr](mailto:julien.damon@free.fr)

*Harald Fielker*

[fielker@informatik.fh-augsburg.de](mailto:fielker@informatik.fh-augsburg.de)

*Marco von Ballmoos*

[mvonballmo@users.sourceforge.net](mailto:mvonballmo@users.sourceforge.net)

*Andrew Eddie*

[eddieajau@users.sourceforge.net](mailto:eddieajau@users.sourceforge.net)

*William K. Hardeman*

[william\\_hardeman@hands-solutions.com](mailto:william_hardeman@hands-solutions.com)

*Chuck Burgess*

[ashnazg@php.net](mailto:ashnazg@php.net)

## Documentation

The release of phpDocumentor 1.2.0+ marks a radical shift in the design philosophy of phpDocumentor. No longer is this tool intended only to generate documentation from the source code. Now it facilitates generating user-level documentation for people who only wish to use your software, and not to modify it. This allows complete integration of all the documentation for the software project, and that means even greater ease of documenting, and ultimately more time to work on making the software work!

The old functionality of phpDocumentor has not been changed, only extended. The old HTMLdefaultConverter has been ported to a new HTMLframesConverter. The output looks the same, or better, but the templates are now smarty.php.net-based templates, and allow much greater customization. To start out right away, read [phpDocumentor Tutorial](#).

## Bugs and Features (guaranteed to be separate entities...)

If you think you've found a bug in phpDocumentor, please check our bug trackers online at [PEAR](#) and [http://sourceforge.net/tracker/?func=browse&group\\_id=11194&atid=111194](http://sourceforge.net/tracker/?func=browse&group_id=11194&atid=111194) [Sourceforge](#), the bug may have already been fixed (we develop at a rapid rate). If you don't find it, please submit a bug (preferably at PEAR), and we will attempt to fix it as quickly as possible.

If there is a feature you're dying to have, check our feature tracker at [PEAR](#) and [http://sourceforge.net/tracker/?func=browse&group\\_id=11194&atid=361194](http://sourceforge.net/tracker/?func=browse&group_id=11194&atid=361194) [Sourceforge](#). Please don't hesitate to look right at the source code and help apply your feature. We love that, and can easily give you access to the cvs.

## How to Help Out

If you would like to help out with phpDocumentor, don't be shy. Email [Joshua Eichorn](#) right away and say you'd like to be added to the developer team. We need people who are willing to develop Converters, translate documentation, and handle various bugs and feature requests as they come up. phpDocumentor is fast becoming the de facto standard of automatic documentors in PHP, so it is in your best interest to make sure it works the way you need it to work.

# phpDocumentor tags

*How to use tags in DocBlocks*

## Tags Manual

Welcome to the phpDocumentor Tags Manual

phpDocumentor tags are very similar to tags for the JavaDoc tool for Sun's Java Programming Language. Tags are only parsed if they are the first thing on a new line of a DocBlock. You may use the @ character freely throughout documents as long as it does not begin a new line. An example:

```
1  /**
2  * tags demonstration
3  * @author this tag is parsed, but this @version tag is ignored
4  * @version 1.0 this version tag is parsed
5  */
```

Any tags that phpDocumentor does not recognize will not be parsed, and will be displayed in text flow as if they are part of the DocBlock long description. The example below displays text "tags demonstration @foobar this is silly," and also displays the author as "this tag is parsed, but this @version tag is ignored"

```
1  /**
2  * tags demonstration
3  * @foobar this is silly
4  * @author this tag is parsed, but this @version tag is ignored
5  */
```

Inline tags display in the text flow where they appear, and are not separated from the description. As of version 1.2, there are several inline tags. The allowed inline tag list is different for tutorials and regular in-code documentation. See [phpDocumentor Inline tags](#) for more information.

The example below will display the text "this function works heavily with [foo\(\)](#) to rule the world" where [foo\(\)](#) represents a hyperlink that links to the function foo()'s documentation.

```
1  /**
2  * inline tags demonstration
3  *
4  * this function works heavily with {@link foo()} to rule the world
5  */
6  function bar()
7  {
8  }
9
10 function foo()
```

```
11 {  
12 }
```

## Standard phpDocumentor Tags

Most tags can be in the DocBlock of any documentable PHP element (for a definition of a documentable PHP element, see [Documentable PHP Elements](#)). Here is a list of the standard tags:

- [@access](#)
- [@author](#)
- [@copyright](#)
- [@deprecated](#)
- [@example](#)
- [@ignore](#)
- [@internal](#)
- [@link](#)
- [@see](#)
- [@since](#)
- [@tutorial](#)
- [@version](#)
- [inline {@internal}](#)
- [inline {@inheritdoc}](#)
- [inline {@link}](#)

# @abstract

*Document an abstract class, class variable or method*

## Description

Use the @abstract tag to declare a method, class variable, or class that must be re-defined in a child class to be valid.

### Warning:

The @abstract tag is only valid in PHP 4, PHP 5 has a keyword abstract.

## Example

Here's an example:

```
1  /**
2  * example of basic @abstract usage in a class
3  * Use this if every single element of the class is abstract
4  * @abstract
5  */
6  class myabstractclass
7  {
8      function function1($baz)
9      {
10         ...
11     }
12
13     function function2()
14     {
15         ...
16     }
17 }
```

# @access

*Access control for an element. @access private prevents documentation of the following element (if enabled).*

## Description

@access controls phpDocumentor's documentation of an element.

If @access is private, the element will not be documented unless specified by command-line switch --parseprivate.

## Example

```
1  /**
2   * function func1, public access is assumed
3   */
4  function func1()
5  {
6  }
7
8  /**
9   * function func2, access is private, will not be documented
10  * @access private
11  */
12 function func2()
13 {
14 }
15
16 /**
17 * This is possible, but redundant. An element has @access public by default
18 * @access public
19 */
20 class class1
21 {
22 /**
23 * all text in this DocBlock will be ignored, unless command-line switch or
24 * setting in a user INI file enables documenting of private elements
25 * @access private
26 */
27 var $private_var
28 /**
29 * Protected is allowed, but does absolutely nothing. Use it to inform users
30 * that an element should only be referenced by this and child classes,
31 * and not directly
32 * @access protected
33 */
34
35 /**
36 * this function is documented
37 */
```

```
38     function publicmethod()
39     {
40     }
41 }
```

# @author

*Author of current element*

## Description

The @author tag is used to document the author of any element that can be documented (global variable, include, constant, function, define, class, variable, method, page). phpDocumentor will take any text between angle brackets (< and >) and try to parse it as an email address. If successful, it will be displayed with a mailto link in the page

**NEW v1.2** - @author is now inherited by child classes from a parent class, see [inline {@inheritDoc}](#).

## Example

```
1  /**
2   * Page-Level DocBlock example.
3   * displays as Gregory Beaver<u>cellog@php.net</u>
4   * , where underlined text is a "mailto:cellog@php.net" link
5   * @author Gregory Beaver <cellog@php.net>
6   */
7 /**
8  * function datafunction
9  * another contributor authored this function
10 * @author Joe Shmoe
11 */
12 function datafunction()
13 {
14 ...
15 }
```

# @category

*Specify a category to organize the documented element's package into*

## Description

The @category tag is used to organize groups of packages together. This is directly applicable to the XML:DocBook/peardoc2 Converter, and can be used by other converters. Other Converters packaged with phpDocumentor ignore the category, but this may change in future versions. It is also possible to dynamically specify category using the [-dc, --defaultcategoryname](#) command-line switch.

## Example

Here's an example:

```
1  /**
2  * Page-Level DocBlock
3  * @package MyPackage
4  * @category mycategory
5  */
6
7 /**
8 * @global array used for stuff
9 */
10 function mine()
11 {
12     global$baz;
13     ...
14 }
```

# @copyright

*Document Copyright information*

## Description

The @copyright tag is used to document the copyright information of any element that can be documented (global variable, include, constant, function, define, class, variable, method, page). phpDocumentor will display the copyright string unaltered.

**NEW v1.2** - @copyright is now inherited by child classes from a parent class, see [inline {@inheritdoc}](#).

## Example

```
1  /**
2   * Page-Level DocBlock example.
3   * @author Gregory Beaver <cellog@php.net>
4   * @copyright Copyright (c) 2002, Gregory Beaver
5   */
6 /**
7  * function datafunction
8 */
9 function datafunction()
10 {
11 ...
12 }
```

# @deprecated

*Document elements that have been deprecated and should not be used as they may be removed at any time from a future version*

## Description

The @deprecated tag is used to document the deprecation version or other information of any element that can be documented except for page (global variable, include, constant, function, define, class, variable, method). If present, phpDocumentor will display the optional version/info string unaltered.

Use @deprecated to notify users of deprecated elements that should not be used any longer

## Example

Here's an example:

```
1  /**
2  * @deprecated deprecated since version 2.0
3  */
4  function uselessfunction()
5  {
6  ...
7  }
8
9 /**
10 * also legal
11 * @deprecated
12 */
13 class stupidclass
14 {
15 ...
16 }
```

# @example

*Include an external example file with syntax highlighting*

## Description

The example tag can be used to parse an example file for syntax highlighting and linking to documentation. This versatile tag attempts to read the file from the full path specified, and will accept any path that <http://www.php.net/fopen> will accept. phpDocumentor checks the path to ensure that the file retrieved has a valid .php extension as defined in phpDocumentor.ini, and then opens the file. It will parse the file, and output syntax-highlighted source with line numbers, links to documentation and will then add a link to the documentation to that file.

If given an absolute path, phpDocumentor will not search for the example file any further. If given a relative path (no leading c:\ or /) phpDocumentor searches for examples files first in the directory specified by the [-ed, --examplesdir](#) command-line, if present. As of phpDocumentor 1.2.1, it will next search for the file in an examples/ subdirectory of the current file's directory. Otherwise, it will search for a subdirectory named "examples" in the top-level parsing directory, and if not found, in the top-level directory.

The top-level parsing directory is simply the directory that every file parsed has in common.

The [@filesource](#) tag serves a similar purpose, but instead of parsing a separate file, it parses the current file's source.

To do an inline code example, use the html tag <code> or the new [inline {@example}](#) tag

### Warning:

@example only works with PHP 4.3.0+ due to the use of the tokenizer extension, which was not stable prior to PHP 4.3.0. Go to <http://www.php.net> and download PHP 4.3.0 to use @example

## Example

Here's an example:

```
1  /**
2  * My function
3  *
4  * Here is an inline example:
5  * <code>
6  * <?php
7  * echo strlen\('6'\);;
8  * ?>
9  * </code>
10 * @example /path/to/example.php How to use this function
11 * @example anotherexample.inc This example is in the "examples" subdirectory
```

```
12  */
13 function mine()
14 {
15 }
```

# @final

*Document a class method that should never be overridden in a child class*

## Description

Use the @final tag to declare a method that cannot be overridden in a child class.

### Warning:

The @final tag is only valid in PHP 4, PHP 5 has a keyword final.

## Example

Here's an example:

```
1  /**
2  * example of basic @final usage in a class
3  */
4  class myclass
5  {
6      /**
7      * function1 should never be overridden
8      * @final
9      */
10     function function1($baz)
11     {
12         ...
13     }
14
15    function function2()
16    {
17        ...
18    }
19 }
```

# @filesource

*create a syntax-highlighted cross-referenced file containing source code of the current file and link to it*

## Description

The @filesource tag can only be used in a page-level DocBlock, it will be ignored anywhere else. phpDocumentor parses the file source of the current file, and outputs syntax-highlighted source code with line numbers, links to documentation and then adds a link to the generated file in the documentation.

The [@example](#) tag serves a similar purpose, but is designed to parse and create a link to an external example file.

### Warning:

@filesource only works with PHP 4.3.0+ due to the use of the tokenizer extension, which was not stable prior to PHP 4.3.0. Go to <http://www.php.net> and download PHP 4.3.0 to use @filesource

## Example

Here's an example:

```
1  /**
2  * Contains several example classes that I want to parse but I
3  * also want to show the full source
4  * @package mypackage
5  * @subpackage example
6  * @filesource
7  */
8 /**
9  * This class does things
10 * @package mypackage
11 * @subpackage example
12 */
13 class oneofmany extends mainclass
14 {
15 ...
```

# @global

*Document a global variable, or its use in a function/method*

## Description

Since there is no standard way to declare global variables, phpDocumentor requires that a @global tag be used in a docblock preceding a global variable's definition. To support previous usage of @global, there is an alternate syntax that applies to DocBlocks preceding a function, used to document usage of global variables. in other words, There are two usages of @global: definition and function usage.

phpDocumentor will not attempt to automatically parse out any global variables. Only one @global tag is allowed per global variable DocBlock. A global variable DocBlock must be followed by the global variable's definition before any other element or DocBlock occurs in the source, or an error will be raised.

datatype should be a valid PHP type or "mixed."

\$varname should be the EXACT name of the global variable as it is declared in the source (use [@name](#) to change the name displayed by documentation)

The function/method @global syntax is used to document usage of global variables in a function, and MUST NOT have a \$ starting the third word. The datatype will be ignored if a match is made between the declared global variable and a variable documented in the project.

phpDocumentor will check to see if the type specified is the name of a class that has been parsed. If so, it will make a link to that class as the type instead of just the type.

phpDocumentor will display the optional description unmodified

## Example

Here's an example of documenting the definition of a global variable:

```
1  /**
2  * example of incorrect @global declaration #1
3  * @global bool $GLOBALS['baz']
4  * @author blahblah
5  * @version -6
6  */
7  include( "file.ext" );
8 // error - element encountered before global variable declaration, docblock will apply to this include!
9 $GLOBALS['baz'] = array('foo','bar');
10
11 /** example of incorrect @global declaration #2
12 * @global parserElement $_Element
```

```

13  /*
14  /**
15  * error - this DocBlock occurs before the global variable definition and will apply to the function,
16  * ignoring the global variable completely
17  */
18 $_Element = new parserElement;
19
20 function oopsie()
21 {
22 ...
23 }
24
25 /** example of correct @global declaration,
26 * even with many irrelevant things in between
27 * @global mixed $_GLOBALS["myvar"]
28 */
29 // this is OK
30 if ($pre)
31 {
32     $thisstuff = 'is fine too';
33 }
34 $_GLOBALS["myvar"] = array( "this" => 'works just fine');
35
36 /**
37 * example of using @name with @global
38 * the @name tag *must* have a $ in the name, or an error will be raised
39 * @global array $_GLOBALS['neato']
40 * @name $neato
41 */
42 $_GLOBALS['neato'] = 'This variable\'s name is documented as $neato, and not as
$_GLOBALS['neato'];

```

Here's an example of documenting the use of a global variable in a function/method:

```

1 /**
2 * Used to showcase linking feature of function @global
3 */
4 class test
5 {
6 }
7
8 /**
9 * @global test $_GLOBALS['baz']
10 * @name $bar
11 */
12 $_GLOBALS['bar'] = new test
13
14 /**
15 * example of basic @global usage in a function
16 * assume global variables "$foo" and "$bar" are already documented
17 * @global bool used to control the weather
18 * @global test used to calculate the division tables
19 * @param bool $baz
20 * @return mixed
21 */
22 function function1($baz)
23 {

```

```
24 global$foo,$bar;  
25 // note that this also works as:  
26 // global $foo;  
27 // global $bar;  
28 if($baz)  
29 {  
30     $a = 5;  
31 } else  
32 {  
33     $a = array(1,4);  
34 }  
35 return$a;  
36 }
```

# @ignore

*Prevent documentation of an element*

## Description

Use @ignore to prevent phpDocumentor from documenting an element, such as a duplicate element.

## Example

Here is an example of how to use @ignore:

```
1  if($ostest)
2  {
3      /**
4      * This define will either be 'Unix' or 'Windows'
5      */
6      define("OS"           , "Unix"           );
7  } else
8  {
9      /**
10     * @ignore
11     */
12     define("OS"           , "Windows"        );
13 }
```

# @internal

*Mark documentation as private, internal to the software project*

## Description

Use @internal to facilitate the creation of two sets of documentation, one for advanced developers or for internal company use, and the other for the general PHP public. Use this tag or its cousin, [inline {@internal}}](#)

@internal responds to the command-line [-pp, --parseprivate](#), as both @internal and --parseprivate share the same purpose. In both cases, the intent is to allow the creation of two sets of documentation, one for public use, and the other for a subset of developers, either the project developers or those who wish to extend the project. In essence, it allows the creation of both user-level and programmer-level documentation from the same source, and is one of phpDocumentor's most useful features.

## Example

Here is an example of how to use @internal and {@internal}:

```
1  /**
2  * This class does things
3  *
4  * {@internal To access the company guidelines on modifying this class,
5  * see {@link http://www.example.com Company Guidelines}, or contact
6  * your supervisor}}
7  * Using this class can be very helpful for several reasons. etc. etc.
8  * @internal the class uses the private methods {@link _foo()} and
9  * {@link _bar()} to do some wacky stuff
10 */
11 class Doesthings
```

# @license

*Display a hyperlink to a URL for a license*

## Description

You may use the @license tag to document any element (include, page, class, function, define, method, variable)

URL is any valid Uniform Resource Locator that points to the full text of a license. The description for the license is optional

## Example

Here's an example:

```
1  /**
2   * Displays <a href="http://opensource.org/licenses/gpl-license.php">GNU Public
3   * License</a>
4   * @license http://opensource.org/licenses/gpl-license.php GNU Public License
5   */
6  class opensource_class {...}
```

# @link

*Display a hyperlink to a URL in the documentation*

## Description

You may use the @link tag to document any element (include, page, class, function, define, method, variable). The guidelines for giving the element name to @link are the same as outlined in the [@see](#) manual page.

### Caution

Unless linking to an element, @link assumes the arguments passed are fully-formed URLs.  
Generally speaking, if you want to link to an element's documentation, use [@see](#) or [inline {@link}](#)...  
you can use @link, but the other options are better.

URL is any valid Uniform Resource Locator (<http://www.example.com>, <telnet://example.com>, <ftp://ftp.example.com>, <mailto:email@example.com>, etc.)

Note that as of version 1.2.0, it is possible to specify a short description for the link to use as the link text instead of the URL.

## Example

Here's an example:

```
1  /**
2  * Displays <a href="http://www.example.com">http://www.example.com</a>
3  * @link http://www.example.com
4  */
5  define("TEST_CONSTANT" ,3);
6 /**
7  * Displays <a href="http://www.example.com">Hello</a>
8  * @link http://www.example.com Hello
9  */
10 define("TEST_CONSTANT2" ,3);
11 /**
12 * You SHOULD use @see here instead of @link, but here's how it works:
13 * displays <a href=".//MyDocs/MyPackage/MyClass.html">MyClass</a>
14 * @link MyClass
15 */
16 define("TEST_CONSTANT3" ,3);
```

# @method

*"Magic" Method of a class.*

## Description

@method shows a "magic" method that is found inside the class.

returntype should be a valid PHP type or "mixed." phpDocumentor will display the optional description unmodified, and defaults to "void" if the returntype is not present.

## Example

```
1  /**
2   * show off @method
3   *
4   * @method int borp() borp(int $int1, int $int2) multiply two integers
5   */
6  class Magician
7  {
8      function __call($method $params)
9      {
10         if ($method == 'borp') {
11             if (count($params) == 2) {
12                 return $params[0] * $params[1];
13             }
14         }
15     }
16 }
```

Using this "magic" tag will result in the tag info being listed in the "infomational" tag listing for the class itself. It does NOT generate actual "method" or "function" entities in the document.

In the example above, when using PhpDocumentor with the HTML:Smarty:HandS converter, you'll see the "\_\_call()" method shown in the "Method Summary" listing of your class methods, but you will NOT see "borp()" there, because the latter is not an explicitly declared method in your code. You WILL see "borp()" in the class-level's "API Tags" listing, thanks to your "@method" tag in the class-level docblock.

# @name

*Specify an alias to use for a procedural page or global variable in displayed documentation and linking*

## Description

phpDocumentor recognizes the @name tag in global variable DocBlocks (in conjunction with [@global](#)), and uses it to rename a global variable for documentation to make it more readable. A global variable name MUST begin with a dollar sign (\$) or phpDocumentor will ignore the tag and raise an error.

## Example

Here's an examples:

```
1 /**
2 * Now, when @global is used in a function, it will link to $baz
3 * @name $baz
4 * @global array $GLOBALS['baz']
5 */
6 $GLOBALS['baz'] = array('foo','bar');
7
8 /**
9 * @global array used for stuff
10 */
11 function mine()
12 {
13     global$baz;
14 }
```

# @package

*Specify package to group classes or functions and defines into*

## Description

@package can only be used to document procedural pages or classes.

**Packages** are used to help you logically group related elements. You write classes to group related functions and data together, and phpDocumentor represents the contents of files (functions, defines, and includes) as "Procedural Pages." A package is used to group classes and procedural pages together in the same manner that a directory groups related files together.

If found in a page-level DocBlock, packagename specifies the package that all functions, includes, and defines in the php file belong to. If found in a class-level DocBlock, packagename specifies the package that the class belongs to.

### Caution

If not present, a class's package is set to **default**, or the value of the -dn command-line switch, **even if the page-level package is set to another value**. This may be changed by the command-line option -dn or --defaultpackagename. PACKAGES MUST BE ONE WORD WITH NO SPACES OR NEWLINES CONTAINING ONLY LETTERS, DIGITS, and "\_", "-", "[" or "]".

@package groups php files together for documentation.

There are two ways a @package tag is parsed:

<ol role = "l">

- page-level package (defines, functions, includes/requires)
- class-level package (class, all its variables and methods)

If a file contains functions and defines, they will be packaged by a page-level DocBlock. A page-level DocBlock is a DocBlock at the top of a file that precedes another DocBlock. In other words, it is not paired with any phpDocumentor element (include, function, define, class, class function, class variable).

## Example

phpDocumentor parses a DocBlock as a page-level DocBlock if it precedes another DocBlock like this:

```
1  /**
2  * Page-Level DocBlock example.
3  * This DocBlock precedes another DocBlock and will be parsed as the page-level.
4  * Put your @package and @subpackage tags here
```

```

5  * @package pagelevel_package
6  */
7 /**
8  * function bluh
9  */
10 function bluh()
11 {
12 ...
13 }

```

A page is documented as a procedural page if it is parsed, regardless of its content (**NEW in 1.0**).

A class-level DocBlock is any DocBlock that precedes a class definition in a php file.

```

1  <?php
2 /**
3  * class bluh
4  * class-level DocBlock example.
5  * @package applies_to_bluh
6  */
7 class bluh
8 {
9 /**
10  * This variable is parsed as part of package applies_to_bluh
11 */
12 var $foo
13 /**
14  * So is this function
15 */
16 function bar()
17 {
18 }
19 }
20 ?>

```

If no @package tag is specified, the package named "default" will be used, with one major exception. If a class extends a parent class that has a @package tag, it will inherit the package from the parent class. This can be very useful in small projects. HOWEVER, it is highly recommended that a @package tag is used before every class that should have that package to avoid name collisions. What is a name collision? If you are documenting two related but separate packages, and both packages contain classes with the same name. For example, the tests package included with phpDocumentor contains several files used to verify the eradication of bugs from phpDocumentor. Several of these files contain classes with the same name as classes in phpDocumentor. phpDocumentor cannot determine the correct parent class automatically and will not inherit package at all to avoid a wrong guess. WE HIGHLY RECOMMEND THAT YOU USE @package IN THE DOCBLOCK OF EVERY CLASS OR PROCEDURAL PAGE YOU WISH TO BE IN A PACKAGE.

Elements can also be grouped into subpackages using [@subpackage](#)

See also [@subpackage](#)

# @param

*Document a function parameter*

## Description

NOTE: as of 0.4.1, @param can document phpdoc.de-style, with optional \$paramname

The datatype should be a valid PHP type (int, string, bool, etc), a class name for the type of object, or simply "mixed". Further, you can list multiple datatypes for a single parameter by delimiting them with the pipe (e.g. "@param int|string \$p1"). You may document parameters listed or any optional parameters that will be parsed by standard PHP functions func\_num\_args()/get\_func\_arg(). Recommended name format for parameters listed with func\_get\_arg() is:

- **\$paramname** if there is only one parameter
- **\$paramname,...** if the number of parameters is unlimited

phpDocumentor will display the optional description unmodified.

Note that the **\$paramname,...** will be shown in the output docs in both the parameter listing AND the function signature. If you are not indicating in the actual code that the parameter is optional (via "\$paramname = 'a default value'"), then you should mention in the parameter's description that the parameter is optional.

## Example

Here's an example:

```
1  /**
2  * example of basic @param usage
3  * @param bool $baz
4  * @return mixed
5  */
6  function function1($baz)
7  {
8      if ($baz)
9      {
10          $a = 5;
11      } else
12      {
13          $a = array(1,4);
14      }
15      return $a;
16  }
17
18  class class1
19  {
20      /**
```

```

21   * example of documenting a method, and using optional description with @return
22   * @return string de-html_entitied string (no entities at all)
23   */
24 function bar$foo()
25 {
26     return strtr($foo,array_flip(get_html_translation_table(HTML_ENTITIES)));
27 }
28 }
29 /**
30  * Example of documenting multiple possible datatypes for a given parameter
31  * @param bool|string $foosometimes a boolean, sometimes a string (or, could have just used
32  "mixed")
33  * @param bool|int $barsometimes a boolean, sometimes an int (again, could have just used
34  "mixed")
35 /**
36 function function2($foo, $bar)
37 {
38     if (!$foo)
39     {
40         // definitely not a string, and not a boolean TRUE... could ONLY be a boolean FALSE
41     }
42     if (!$bar)
43     {
44         // could ONLY be a boolean FALSE or an integer "0"
45     }
46 }
47 /**
48  * Example of documenting undetermined function arguments
49  * (notice how $foo_desc is NOT part of the actual function signature in the code, but still gets
50  documented)
51  * @param string $foo
52  * @param mixed $foo_descoptional description of foo
53 /**
54 function function3($foo)
55 {
56     echo $foo
57     if(func_num_args == 2)
58     {
59         echo 'Description: '.func_get_arg(1);
60     }
61 }
62 /**
63  * Example of unlimited parameters.
64  * Returns a formated var_dump for debugging purposes
65  * (since the recurrences of $v are not listed in the actual function signature in the code,
66  * you may wish to highlight they are "optional" in their description)
67  * @param string $sstring to display
68  * @param mixed $vvariable to display with var_dump()
69  * @param mixed $v,...unlimited OPTIONAL number of additional variables to display with var_dump()
70  */
71 function fancy_debug($s,$v)
72 {
73     print $s."<blockquote>\n";
74     var_dump( $v);
75     if(func_num_args() > 2)
76 
```

```
77  {
78  for($i=2;$i< func_num_args(); $i++)
79  {
80      $a = func_get_arg( $i);
81      var_dump( $a);
82      print "<br>\n" ;
83  }
84  print "</blockquote>\n" ;
85 }
```

# @property

"Magic" Property of a class.

## Description

@property shows a "magic" property variable that is found inside the class.

datatype should be a valid PHP type or "mixed." phpDocumentor will display the optional description unmodified, and defaults to "mixed" if the datatype is not present.

The property is presumed to be available for both read and write operations. If the property is read-only, you should use the @property-read tag instead. If the property is write-only, use @property-write.

## Example

```
1  /**
2  * show off @property, @property-read, @property-write
3  *
4  * @property mixed $regular regular read/write property
5  * @property-read int $foo the foo prop
6  * @property-write string $bar the bar prop
7  */
8  class Magician
9  {
10     private $_thingy;
11     private $_bar;
12
13     function __get($var)
14     {
15         switch($var) {
16             case 'foo':
17                 return 45;
18             case 'regular':
19                 return $this->_thingy;
20         }
21
22     function __set($var, $val)
23     {
24         switch($var) {
25             case 'bar':
26                 $this->_bar = $val;
27                 break;
28             case 'regular':
29                 if (is_string($val)) {
30                     $this->_thingy = $val;
31                 }
32     }
```

Using these "magic" tags will result in the tag info being listed in the "infomational" tag listing for the class itself. It does NOT generate actual "property" or "variable" entities in the document.

In the example above, when using PhpDocumentor with the HTML:Smarty:HandS converter and the "--parseprivate on" setting, you'll see "\$\_thingy" and "\$\_bar" shown in the "Property Summary" listing of your class variables, but you will NOT see "\$regular", "\$foo", or "\$bar" there, because the latter three are not explicitly declared variables in your code. You WILL see "\$regular", "\$foo" and "\$bar" in the class-level's "API Tags" listing, thanks to your "@property" tags in the class-level docblock.

# @return

*Specify the return type of a function or method*

## Description

The @return tag is used to document the return value of functions or methods. @returns is an alias for @return to support tag formats of other automatic documentors

The datatype should be a valid PHP type (int, string, bool, etc), a class name for the type of object returned, or simply "mixed". If you want to explicitly show multiple possible return types, list them pipe-delimited without spaces (e.g. "@return int|string"). If a class name is used as the datatype in the @return tag, phpDocumentor will automatically create a link to that class's documentation. In addition, if a function returns multiple possible values, separate them using the | character, and phpDocumentor will parse out any class names in the return value. phpDocumentor will display the optional description unmodified.

## Example

Here's an example:

```
1  /**
2   * example of basic @return usage
3   * @return mixed
4   */
5  function function1($baz)
6  {
7    if($baz)
8    {
9      $a = 5;
10   } else
11   {
12     $a = array(1,4);
13   }
14   return$a;
15 }
16 /**
17 * example of showing multiple possible return types
18 * @return int|string could be an int, could be a string
19 */
20
21 function function2($foo)
22 {
23   if($foo)
24   {
25     return 0;
26   }
27   else
28   {
29     return "zero"
30   }
31 }
```

```
31 }
32
33 class class1
34 {
35 /**
36 * example of documenting a method, and using optional description with @return
37 * @return string de-html_entitied string (no entities at all)
38 */
39 function bar($foo)
40 {
41     return strtr($foo,array_flip(get_html_translation_table(HTML_ENTITIES)));
42 }
43
44 /**
45 * example of using @return with a class name
46 * @param integer even or odd integer
47 * @return Parser|false phpDocumentor Parser object or error
48 */
49 function& factory($number)
50 {
51     $returnval= true;
52     if($number% 2)
53     {
54         $returnval= new Parser;
55     } else
56     {
57         $returnval= false;
58     }
59     return$returnval;
60 }
61 }
```

# @see

*Display a link to the documentation for an element*

## Description

The @see tag may be used to document any element (global variable, include, page, class, function, define, method, variable)

### Caution

@see only displays links to element documentation. If you want to display a hyperlink, use [@link](#) or [inline {@link}](#)

**New in version 1.2:** You can link to any defined function in the current php version using the function's name. This linking is done using the standard php function [http://www.php.net/get\\_defined\\_functions](http://www.php.net/get_defined_functions), and so relies on the version of php that is used to execute phpDocumentor. A benefit of this method is that the function highlighting will automatically upgrade with a php upgrade without any change to the underlying code. You may also link directly to any portion of the php website using the fake package override PHP MANUAL (as in PHP MANUAL#get\_defined\_functions, or PHP MANUAL#support.php)

Along with [inline {@link}](#), the @see tag is among the most useful features of phpDocumentor. With this tag, you can create a link to any element (except include/require) in the documentation with a very wide range of options. The @see parser can be told exactly where to look using some basic punctuation:

- :: -- This is the class scope override. Use it to tell the @see parser which class to look in for the element specified, like **classname::methodname()** or **classname::\$variablename**.
- () -- When present at the end of elementname, like **elementname()**, this tells the @see parser to search for a function or method.
- \$ -- When present at the beginning of elementname, like **\$elementname**, this tells the @see parser to search for a variable in the current class.

However, @see is also intelligent enough to recognize shorthand. If @see receives an elementname with no punctuation, it will search for an element in this order:

1. is elementname the name of a class?
  2. is elementname the name of a procedural page? (file.ext)
  3. is elementname the name of a define?
  4. if the DocBlock containing the @see is in a class:
    1. is elementname a method?
    2. is elementname a variable?
- is elementname a function?

@see parsing is slightly slower when passed an elementname with no punctuation, especially if the elementname is a function, so use it sparingly in large projects (500+ elements with @sees in their DocBlocks). The best use for punctuation-less elementname is in a project whose classnames are in flux.

## Example

Here's an example of valid @see syntax:

```
1  /**
2  * class 1
3  *
4  * example of use of the :: scope operator
5  * @see subclass::method()
6  */
7 class main_class
8 {
9  /**
10  * example of linking to same class, outputs <u>main_class::parent_method()</u>
11  * @see parent_method
12  */
13  var foo = 3;
14
15 /**
16  * subclass inherits this method.
17  * example of a word which is either a constant or class name, in this case a classname
18  * @see subclass
19  * @see subclass::$foo
20  */
21
22 function parent_method()
23 {
24     if($this>  foo== 9) die;
25 }
26 }
27
28 /**
29 * this class extends main_class.
30 * example of linking to a constant, and of putting more than one element on the same line
31 * @see main_class, TEST_CONST
32 */
33 subclass extends main_class
34 {
35 /**
36  * example of same class lookup - see will look through parent hierarchy to
37  * find the method in { @link main_class}
38  * the above inline link tag will parse as <u>main_class</u>
39  * @see parent_method()
40  */
41  var $foo= 9;
42 }
43
44 define("TEST_CONST" , "foobar" );
```

# @since

*Document when (at which version) an element was first added to a package*

## Description

The @since tag may be used to document the release version of any element that can be documented (global variable, include, constant, function, define, class, variable, method, page). phpDocumentor will display the version/info string unaltered.

Use @since to document revisions, as in "This function has been a part of this package since version 2.0."

## Example

Here's an example:

```
1  /**
2  * Page-level DocBlock
3  * @package BigImportantProjectWithLotsofVersions
4  * @version 72.5
5  */
6 /**
7  * function datafunction
8  * @since Version 21.1
9  */
10 function datafunction()
11 {
12 ...
13 }
```

# @static

*Document a static property or method*

## Description

Use the @static tag to declare a variable or method to be static. Static elements can be called without reference to an instantiated class object, as in class::variable and class::method().

## Example

Here's an example:

```
1  /**
2   * example of a class
3   * with a single static variable
4   * and method
5   */
6  class myclass
7  {
8      /**
9       * a static variable
10      * @static
11      */
12     public static $astaticvar= 0;
13     /**
14      * normal variable
15      */
16     public $anormalvar= 1;
17     /**
18      * a static function
19      * @static
20      */
21     function mystaticfunction()
22     {
23     ...
24     }
25     /**
26      * normal function
27      */
28     function mynormalfunction()
29     {
30     ...
31     }
32 }
33 // example of usage of static methods
34 myclass::mystaticvar;
35 myclass::mystaticfunction();
```

Just using the **static** keyword in your code is enough for PhpDocumentor on PHP5 to recognize static variables and methods, and PhpDocumentor will mark them as static.

However, using the static tag will **also** result in PhpDocumentor marking the variable or method as static, even if the PHP code does **not** use the static keyword. So, using the tag but not the keyword actually means your code behavior will not match your API doc... so handle with care. The good news is that using both the keyword and the tag will not result in a double "static" in the resulting doc.

Ideally, the static keyword in your code is sufficient for the resulting docs to show your members are static. This tag appears to be a "helpful" tag for use in PHP4 code, allowing you to hint that you want the member to be treated as static, even if PHP4 doesn't have the ability to make it act that way. Most likely, you'll use this tag as a reminder that your intention with this member, once you move the code from PHP4 to PHP5, will be for that member to be static.

# @staticvar

*Document a static variable's use in a function/method*

## Description

Datatype should be a valid PHP type or "mixed."

phpDocumentor will display the optional description unmodified

## Example

Here's an example:

```
1  /**
2  * example of basic @staticvar usage in a function
3  * @staticvar integer used to calculate the division tables
4  * @staticvar array $bar is used to make me happy. Note that $bar is part of the description
5  * @param bool $baz
6  * @return mixed
7  */
8  function function1($baz)
9  {
10    static $foo= 6;$bar= array();
11    // note that this works as:
12    // static $foo = 6;
13    // static $bar = array();
14    if ($baz)
15    {
16      $a = 5;
17    } else
18    {
19      $a = array(1,4);
20    }
21    return $a;
22 }
```

# @subpackage

*Specify sub-package to group classes or functions and defines into. Requires [@package tag](#)*

## Description

@subpackage works with [@package](#) to group php files together for documentation.

If found in a page-level DocBlock, specifies the sub-package that all functions and defines in the php file belong to. If found in a class-level DocBlock, specifies the sub-package that the class belongs to.

### Caution

If @package is not present in the same DocBlock, the @subpackage tag is ignored. A SUBPACKAGE NAME MUST BE ONE WORD WITH NO SPACES OR NEWLINES CONTAINING ONLY LETTERS, DIGITS, and "\_", "-", "[" or "]"

A @subpackage tag is applied to documented elements in the same way that @package is applied.

If a file contains functions and defines, they will be packaged by a page-level DocBlock. A page-level DocBlock is a DocBlock that is not paired with any phpDocumentor element (include, function, define, class, class function, class variable).

NOTE: The @subpackage tag is intended to help categorize the elements that are in an actual @package value. Since PHP itself doesn't allow you to have two functions with the same name in the same script, PhpDocumentor also requires all names in an @package to be unique... meaning, @subpackage does not allow further "naming separation" inside that @package. What it does do is allow a level of visual grouping/separation of the elements inside that @package.

## Example

phpDocumentor parses a DocBlock as a page-level DocBlock if it precedes another DocBlock like this:

```
1  /**
2  * Page-Level DocBlock example.
3  * This DocBlock precedes another DocBlock and will be parsed as the page-level.
4  * Put your @package and @subpackage tags here
5  * @package pagelevel_package
6  * @subpackage data
7  */
8 /**
9 * function datafunction
10 */
11 function datafunction()
12 {
```

```
13 ...
14 }
```

A class-level DocBlock is any DocBlock that precedes a class definition in a php file.

```
1 <?php
2 /**
3 * class bluh
4 * class-level DocBlock example.
5 * @package applies_to_bluh
6 * @subpackage bluh
7 */
8 class bluh
9 {
10 /**
11 * This variable is parsed as part of package applies_to_bluh, and subpackage bluh
12 */
13 var $foo
14 /**
15 * So is this function
16 */
17
18 function bar()
19 {
20 }
21 ?>
```

See also: [@package](#)

# @todo

*Document changes that will be made in the future*

## Description

Use @todo to document planned changes to an element that have not been implemented. phpDocumentor will display the information string unaltered.

The @todo tag may be used to document future changes to almost any element that can be documented (global variable, constant, function, define, class, variable, method, page). **Since v1.3.0**, @todo can no longer be used on an "include" element.

## Example

Here's an example:

```
1  /**
2  * Page-level DocBlock
3  * @package unfinished
4  * @todo finish the functions on this page
5  */
6 /**
7  * function datafunction
8  * @todo make it do something
9  */
10 function datafunction()
11 {
12 }
```

# @tutorial

*Display a link to the documentation for a tutorial*

## Description

The @tutorial tag may be used to document any element (global variable, include, page, class, function, define, method, variable)

### Caution

@tutorial only displays links to tutorials/extended documentation. If you want to display a hyperlink or link to a documented element, use [@see](#) or [@link](#) or [inline {@link}](#)

Along with [inline {@tutorial}](#), the @tutorial tag is among the most useful features of phpDocumentor. With this tag, you can create a link to any tutorial or extended documentation. The @tutorial parser can be told exactly where to look using a format similar to a URI:

- **package/** -- Preface to specify a tutorial in a specific package
- **subpackage/** -- Preface to specify a tutorial in a specific package/subpackage
- **#section** -- Link to a specific section of the documentation package/subpackage/tutorial.ext#section
- **.subsection** -- Link to a specific sub-section of the documentation package/subpackage/tutorial.ext#section.subsection

However, @tutorial, like @see, is also intelligent enough to recognize shorthand. The search order for a tutorial is identical to what is specified in [@see](#)

## Example

Here's an example of valid @tutorial syntax:

```
1  /**
2   * This will link to the phpDocumentor.pkg tutorial if it is unique, regardless
3   * of its location
4   * @tutorial phpDocumentor.pkg
5   */
6  class test1
7  {
8  }
9
10 /**
11  * This will link to the phpDocumentor.pkg tutorial if and only if it is in the
12  * phpDocumentor package, any subpackage.
13  * @tutorial phpDocumentor/phpDocumentor.pkg
14  */
15 class test2
16 {
```

```
17 }
18 /**
19 * This will link to the phpDocumentor.pkg tutorial if and only if it is in the
20 * phpDocumentor package, info subpackage.
21 * @tutorial phpDocumentor/info/phpDocumentor.pkg
22 */
23
24 class test3
25 {
26 }
27
28 /**
29 * This will link to the phpDocumentor.pkg tutorial, section1 and subsection one
30 * @tutorial phpDocumentor.pkg#section1, phpDocumentor.pkg#section1.subsection1
31 */
32 class test4
33 {
34 }
```

# @uses

*Display a link to the documentation for an element, and create a backlink in the other element's documentation to this*

## Description

### Caution

Since version 1.2.0beta3, the syntax for @uses has changed. It differs in that @uses now contains a description of how the element is used and does not allow a comma-delimited list of values any longer.

The @uses tag may be used to document any element (global variable, include, page, class, function, define, method, variable)

### Caution

@uses only displays links to element documentation. If you want to display a hyperlink, use [@link](#) or [inline {@link}](#)

@uses is very similar to [@see](#), see the documentation for @see for details on format and structure. The @uses tag differs from @see in two ways. @see is a one-way link, meaning the documentation containing a @see tag contains a link to other documentation. The @uses tag automatically creates a virtual @usedby tag in the other documentation that links to the documentation containing the @uses tag. In other words, it is exactly like @see, except a return link is added automatically.

The other difference is that @uses only accepts one element to link to, and a description of that element.

## Example

Here's an example of valid @uses syntax:

```
1  /**
2  * class 1
3  *
4  */
5 class main_class
6 {
7  /**
8  * @var integer
9  */
10 var foo = 3;
11
12 /**
13 * subclass inherits this method.
14 * example of a word which is either a constant or class name,
15 * in this case a classname
16 * @uses subclass sets a temporary variable
```

```

17 * @uses subclass::$foo this is compared to TEST_CONST
18 * @uses TEST_CONST compared to subclass::$foo, we
19 *       die() if not found
20 */
21
22 function parent_method()
23 {
24     if($this> foo== 9)die;
25     $test= new subclass;
26     $a = $test> foo;
27     if($a == TEST_CONST)die;
28 }
29 }
30
31 /**
32 * this class extends main_class.
33 */
34 subclass extendsmain_class
35 {
36     /**
37      * @var integer
38     */
39     var $foo= 9;
40 }
41
42 define("TEST_CONST" , "foobar");

```

This will parse as if it were:

```

1  /**
2  * class 1
3  *
4  */
5 class main_class
6 {
7 /**
8  * @var integer
9  */
10 var foo = 3;
11
12 /**
13  * subclass inherits this method.
14  * example of a word which is either a constant or class name,
15  * in this case a classname
16  * @uses subclass sets a temporary variable
17  * @uses subclass::$foo this is compared to TEST_CONST
18  * @uses TEST_CONST compared to subclass::$foo, we
19  *       die() if not found
20 */
21
22 function parent_method()
23 {
24     if($this> foo== 9)die;
25     $test= new subclass;
26     $a = $test> foo;
27     if($a == TEST_CONST)die;
28 }

```

```
29 }
30
31 /**
32 * this class extends main_class.
33 * @usedby main_class::parent_method() sets a temporary variable
34 */
35 subclass extendsmain_class
36 {
37 /**
38 * @var integer
39 * @usedby main_class::parent_method() this is compared to TEST_CONST
40 */
41 var $foo= 9;
42 }
43
44 /**
45 * @usedby main_class::parent_method() compared to subclass::$foo, we
46 * die() if not found
47 */
48 define("TEST_CONST" , "foobar" );
```

# @var

*Document the data type of a class variable*

## Description

You may use the @var tag to document the data type of class variables.

The datatype should be a valid PHP type (int, string, bool, etc), a class name for the type of object, or simply "mixed". phpDocumentor will display the optional description unmodified, and defaults to "mixed" if the datatype is not present

## Example

Here's an example:

```
1  class class1
2  {
3      /**
4      * example of documenting a variable's type
5      * @var string
6      */
7      var $variable
8      /**
9      * example of documenting a variable's type
10     * @var string contains class1 information
11    */
12    var $variable_with_desc
13    /**
14     * this variable is documented as type "mixed" since no @var tag is present
15    */
16    var $mixed_variable
17 }
```

# @version

*Version of current element*

## Description

Documents the version of any element, including a page-level block

**NEW v1.2** - @version is now inherited by child classes from a parent class, see [inline {@inheritDoc}](#).

## Example

phpDocumentor displays a @version string without modification, so it may be used in any way you'd like:

```
1  /**
2  * example of @version tag with CVS tag
3  * @version $Id: tags.version.pkg,v 1.2 2006-04-29 04:08:27 cellog Exp $;
4  */
5  function datafunction()
6  {
7  }
8
9 /**
10 * Custom version string
11 * @version customversionstring1.0
12 */
13 class blah
14 {
15 }
```

# phpDocumentor Inline tags

*How to use Inline tags in DocBlocks and in Tutorials*

## Inline Tags Manual

Welcome to the phpDocumentor Inline Tags Manual

Unlike regular tags (see [phpDocumentor tags](#)), Inline tags display in the text flow where they appear. As of version 1.2, there are several inline tags. The allowed inline tag list is different for tutorials and regular in-code documentation.

Inline tags allowed in both tutorials and DocBlocks include:

- [inline {@link}](#)
- [inline {@tutorial}](#)

Inline tags allowed in only DocBlocks include:

- [inline {@source}](#)
- [inline {@inheritdoc}](#)

Inline tags allowed in only tutorials include:

- [inline {@id}](#)
- [inline {@toc}](#)

The example below will display the text "this function works heavily with [foo\(\)](#) to rule the world" where [foo\(\)](#) represents a hyperlink that links to the function foo()'s documentation.

```
1  /**
2  * inline tags demonstration
3  *
4  * this function works heavily with {@link foo()} to rule the world. If I want
5  * to use the characters "{@link" in a docblock, I just use "{@}link." If
6  * I want the characters "{@ *}" I use "{@}*"
7  */
8  function bar()
9  {
10 }
11
12 function foo()
```

# inline {@example}

*Display source code of an example file inline*

## Description

Like the [{@example}](#) tag, the inline example tag can be used to parse an example file for syntax highlighting and linking to documentation. This versatile tag attempts to read the file from the full path specified, and will accept any path that <http://www.php.net/fopen> will accept. phpDocumentor checks the path to ensure that the file retrieved has a valid .php extension as defined in phpDocumentor.ini, and then opens the file. It will parse the file, and output syntax-highlighted source with line numbers, links to documentation and will then add a link to the documentation to that file.

If given an absolute path, phpDocumentor will not search for the example file any further. If given a relative path (no leading c:\ or /) phpDocumentor searches for examples files first in the directory specified by the [-ed](#), [-examplesdir](#) command-line, if present. As of phpDocumentor 1.2.1, it will next search for the file in an examples/ subdirectory of the current file's directory. Otherwise, it will search for a subdirectory named "examples" in the top-level parsing directory, and if not found, in the top-level directory.

The top-level parsing directory is simply the directory that every file parsed has in common.

The [inline {@source}](#) tag serves a similar purpose, but instead of parsing a separate file, it parses the current function or method's source.

The {@example} tag has two optional parameters, the starting line number to display and the ending line number to display. If only the first parameter is present, {@example} will print the source code starting on that line number to the end of the function source code. If both parameters are present, {@example} will print an excerpt of the source code beginning on the starting line number, and concluding with the ending line number.

## Example

Here's an example:

```
1  /**
2  * My function
3  *
4  * Here is an inline example:
5  * <code>
6  * <?php
7  * echo strlen\('6'\);;
8  * ?>
9  * </code>
10 * and using an external example file:
11 * {@example /path/to/example.php}
12 *
13 * Display only the first 2 lines:
14 * {@example /path/to/example.php 0 2}
```

```
15 * @example /path/to/example.php How to use this function
16 * @example anotherexample.inc This example is in the "examples" subdirectory
17 */
18 function mine()
19 {
20 }
```

# inline {@id}

*Used to control XML tutorial ids in refsects*

## Description

The {@id} inline tag is used only in tutorials/extended documentation. It is very simply designed to solve the problem of identifying sections for hyperlinking. For instance, in the DocBook Converter, id's are in the format package.packagename[.subpackagename].file[.sectionname], where file is either the classname, function/method name, or summary. In the other Converters, the id is not nearly as restrictive, but this inline tag allows for the potential of future converters that are equally exacting. Use <refsect1 id="{@id idname}"> in the tutorial (tutorialname.pkg/tutorialname.cls/tutorialname.proc) to do this.

The {@id} tag is context-sensitive, and will pre-pend the id of the current refsect1, refsect2, or refsect3 to any {@id}, allowing further error control.

## Caution

The sectionname may only consist of **lower-case** letters, numbers, and hyphens "-." No other characters are allowed

Here is an example of how {@id} must be used

```
1  <refentry id="      {@id}"      >
2  <refnamediv>
3  <refname>    Main Title</refname>
4  <refpurpose>  Description of tutorial</refpurpose>
5  </refnamediv>
6  <refsynopsisdiv>
7  <refsynopsisdivinfo>
8  <author>
9   Gregory Beaver
10 </author>
11 <copyright>  Copyright 2002, Gregory Beaver</copyright>
12 </refsynopsisdivinfo>
13 </refsynopsisdiv>
14 <refsect1 id="      {@id section}"      >
15 <title>    First Section Title</title>
16 <para>
17 Note that the id parameter must be within quotes
18 and there must be no whitespace on either side.
19 id=" {@id section}" will break the id parsing
20 </para>
21 <para>
22 This section's name is
23 "packagesubpackagetutorial.ext.section" by default.
24 The HTML/CHM converters name it "section," the XML
25 converter names it "package.subpackage.tutorial-ext.section";
26 </para>
```

```
27 <refsect2 id="@id subsection">
28   <title> Subsection Title</title>
29   <para>
30     This section inherit's the refsect1's id, and is named
31     "packagesubpackage/tutorial.ext.sectionsubsection"; or
32     "sectionsubsection"; for HTML/CHM, and
33     "packagesubpackage/tutorial-ext.sectionsubsection"; in XML
34   </para>
35 </refsect2>
36 </refsect1>
37 </refentry>
```

# inline {@internal} }

*Mark documentation as private, internal to the software project in text flow*

## Description

### Caution

Unlike other inline tags, {@internal} may contain other inline tags due to its purpose. To terminate an {@internal} inline tag, you must use two ending brackets "}}"

Use {@internal} to facilitate the creation of two sets of documentation, one for advanced developers or for internal company use, and the other for the general PHP public. Use this inline tag or its cousin, [@internal](#)

# inline {@inheritDoc}

Used to directly inherit the long description from the parent class in child classes

## Description

The {@inheritDoc} inline tag is used in the DocBlocks of classes, methods and class variables of child classes. phpDocumentor will automatically inherit the [@author tag](#), [@version tag](#), and [@copyright tag](#) from a parent class. In addition, if there is no documentation comment present, it will inherit the parent's documentation.

Note that if the {@inheritDoc} inline tag is not present, and a child class is undocumented, the child class will still attempt to inherit documentation from the parent class as is. {@inheritDoc} allows flexibility of where to put documentation from the parent class in a child class's documentation.

## Example

In some cases, the child class may want to simply augment the parent class's documentation. This is the primary purpose {@inheritDoc} was designed for. For example:

```
1  /**
2  * Makes bars
3  *
4  * This class generates bars using the main algorithm.
5  */
6  class bar
7  {
8  }
9
10 /**
11 * Makes chocolate bars
12 *
13 * There are two aspects to this class.
14 * {@inheritDoc} In addition, the foo class
15 * makes the bars chocolate
16 */
17 class foo extends bar
18 {
19 }
```

This source code will parse as if it was:

```
1  /**
2  * Makes bars
3  *
4  * Generates bars using the main algorithm.
5  */
6  class bar
7 {
```

```
8  }
9
10 /**
11  * Makes chocolate bars
12  *
13  * There are two aspects to this class.
14  * This class generates bars using the main algorithm. In addition, the foo class
15  * makes the bars chocolate
16  */
17 class foo extends bar
18 {
19 }
```

# inline {@link}

*Display a link to a URL, or link to an element's documentation in the the flow of descriptive text*

## Description

The inline {@link} tag is used to document any element (page, global variable, include, class, function, define, method, variable).

**New in version 1.2:** You can link to any defined function in the current php version using the function's name. This linking is done using the standard php function [http://www.php.net/get\\_defined\\_functions](http://www.php.net/get_defined_functions), and so relies on the version of php that is used to execute phpDocumentor. A benefit of this method is that the function highlighting will automatically upgrade with a php upgrade without any change to the underlying code. You may also link directly to any portion of the php website using the fake package override PHP MANUAL (as in PHP MANUAL#get\_defined\_functions, or PHP MANUAL#support.php).

### Caution

inline {@link} displays links directly in the natural text flow of a DocBlock. If you want to display links prominently after descriptive text, use [@see](#) or [@link](#).

For parameter information, see [@see](#) or [@link](#). The guidelines for giving the element name to inline @link are the same as outlined in the [@see](#) manual page.

## Example

The inline {@link} differs from ordinary non-inline tags. Inline tags parse and display their output directly in the documentation. In other words, this DocBlock:

```
1  /**
2  * Text with a normal @see and @link tag
3  * (the parentheses in "element()" are only necessary
4  * because it is a function)
5  * @see element()
6  * @link http://www.example.com
7  */
8  function element()
9  {
10 }
```

Parses as (with -o HTML:frames:default):

```
1  <H3>    element</H3>
2  <P><B>        element ( )</B></P>
3  <!-- ===== Info from phpDoc block ===== -->
```

```

4  <b>      </b>
5  <pre>
6  Text with a normal @see and @link tag (the parentheses in "element()" are only necessary
because it is a function)
7  </pre>
8  <DL>
9    <DT>    Function Parameters:</DT>
10
11
12    <DT>    Function Info:</DT>
13    <DD><b>        See</b>      - <CODE><a
href="../default/_fake_page_php.html#element"          > element()</a></CODE></DD>
14    <DD><b>        Link</b>      - <CODE><a
href="http://www.example.com"           > www.example.com</a></CODE></DD>
15
16  </DL>
17
18  <HR>
```

```

1  /**
2  * Text with an {@link http://www.example.com Inline Link to a Hyperlink} and an inline
3  * link to {@link element()} displays without a break in the flow
4  * (again, the parentheses in "element()" are only necessary
5  * because it is a function)
6  */
7  function element()
8  {
9 }
```

Parses as (with -o HTML:frames:default):

```

1  <H3>    element</H3>
2  <P><B>        element ()</B></P>
3  <!-- ===== Info from phpDoc block ===== -->
4  <b>      </b>
5  <pre>
6  Text with an <a href="http://www.example.com"          > Inline Link to a Hyperlink</a> and
7  an inline link to <a href="../default/_fake_page_php.html#element"          > element()</a>
8  displays without a break in the flow (again, the parentheses in "element()" are only necessary
9  because it is a function)
10 </pre>
11 <DL>
12   <DT>    Function Parameters:</DT>
13
14
15   <DT>    Function Info:</DT>
16
17 </DL>
18
19 <HR>
```

# inline {@source}

*Display source code of a function or method in the long description*

## Description

The {@source} inline tag is used in the DocBlock of functions or methods to extract source code of the function/method for display in generated documentation. This tag is designed to allow detailed line-by-line description of how a function works without having to cut and paste from the source. This allows modification to the source and automatic update of documentation so that it remains current.

## Example

The inline {@source} differs from all ordinary tags. Inline tags parse and display their output directly in the documentation. In other words, this DocBlock:

```
1  /**
2  * Text with a normal tag, @copyright
3  * @copyright Copyright 2002, Greg Beaver
4  */
5  function element()
6  {
7 }
```

will display normal documentation.

This DocBlock:

```
1  /**
2  * Text with an inline source tag:
3  *
4  * {@source }
5  * displays without a break in the flow
6  */
7  function element($pages)
8  {
9      if(empty( $pages))
10     {
11         die("<b>ERROR</b>: nothing parsed" );
12     }
13 }
```

will display the complete source inside the documentation as if we had typed it out

```
1  /**
2  * Text with an inline source tag:
```

```

3  *
4  * <code>
5  * function element($pages)
6  {
7  *   if (empty($pages))
8  *   {
9  *     die("<b>ERROR</b>: nothing parsed");
10 *   }
11 * }
12 * </code>
13 * displays without a break in the flow
14 */

```

The {@source tag has two optional parameters, the starting line number to display and the ending line number to display. If only the first parameter is present, {@source} will print the source code starting on that line number to the end of the function source code. If both parameters are present, {@source} will print an excerpt of the source code beginning on the starting line number, and concluding with the ending line number.

```

1 /**
2 * Text with an inline source tag:
3 *
4 * {@source 3}
5 * displays without a break in the flow
6 */
7 function element($pages)
8 {
9   if(empty( $pages))
10  {
11    die("<b>ERROR</b>: nothing parsed"
12 );
13   foreach($test as $hello)
14   {
15     echo "      I love  $hello      ";
16   }
17 }

```

Parses as if we had coded:

```

1 /**
2 * Text with an inline source tag:
3 *
4 * <code>
5 *   if (empty($pages))
6 *   {
7 *     die("<b>ERROR</b>: nothing parsed");
8 *   }
9 *   foreach($test as $hello)
10 *   {
11 *     echo "I love $hello";
12 *   }
13 * }
14 * </code>
15 * displays without a break in the flow
16 */

```

```
1  /**
2  * Text with an inline source tag:
3  * {@source 3 1}
4  * displays without a break in the flow
5  */
6 function element($pages)
7 {
8     if(empty( $pages))
9     {
10         die("<b>ERROR</b>: nothing parsed" );
11     }
12     foreach($test as $hello)
13     {
14         echo "    I love  $hello    ";
15     }
16 }
```

Parses as if we had coded:

```
1  /**
2  * Text with an inline source tag:
3  *
4  * <code>
5  *   if (empty($pages))
6  * </code>
7  * displays without a break in the flow
8  */
```

# inline {@toc}

*Display a table of contents of all {@id}s declared in a file*

## Description

The inline {@toc} tag is used as a sub-element of the <refentry> tag to display a table of contents of the current tutorial. This table of contents is generated directly by a call to [Converter::formatTutorialTOC\(\)](#).

## Example

This sample tutorial:

```
1  < refentry id="@id"          >
2  < refnamediv>
3  < refname>@{toc} example</ refname>
4  < repurpose> Uses {@toc} to generate a table of contents</ repurpose>
5  </ refnamediv>
6  {@toc}
7  < refsect1 id="@id one">
8  < title> First section</ title>
9  </ refsect1>
10 < refsect1 id="@id two">
11 < title> Second section</ title>
12 < refsect2 id="@id sub">
13 < title> Sub-section</ title>
14 </ refsect2>
15 </ refsect1>
16 < refsect1 id="@id three">
17 < title> Third section</ title>
18 </ refsect1>
19 </ refentry>
```

Will parse as if it were

```
1  < refentry id="@id"          >
2  < refnamediv>
3  < refname>@{toc} example</ refname>
4  < repurpose> Uses {@toc} to generate a table of contents</ repurpose>
5  </ refnamediv>
6
7  < refsect1>
8  < itemizedlist>
9  < listitem>< para>@ tutorial mytutorial#one</para></listitem>
10 < listitem>< para>@ tutorial mytutorial#two</para>
11 < itemizedlist>
12 < listitem>< para>@ tutorial mytutorial#two.sub</para></listitem>
13 </ itemizedlist>
14 </ para></ listitem>
```

```
15  < listitem>< para>{@ tutorial mytutorial#three}</para></listitem>
16 </ refsect1>
17
18 < refsect1 id="{@id one}"          >
19 < title> First section</ title>
20 </ refsect1>
21 < refsect1 id="{@id two}"          >
22 < title> Second section</ title>
23 < refsect2 id="{@id sub}"          >
24 < title> Sub-section</ title>
25 </ refsect2>
26 </ refsect1>
27 < refsect1 id="{@id three}"        >
28 < title> Third section</ title>
29 </ refsect1>
30 </ refentry>
```

The format of the table of contents is Converter-dependent, and for the converters supplied with phpDocumentor, can be set up through the use of tutorial\_toc.tpl in the template for tremendous customization control

# inline {@tutorial}

*Display a link to a tutorial in the flow of descriptive text*

## Description

The inline {@tutorial} tag is used in any context to link to the documentation for a tutorial.

### Caution

inline {@tutorial} displays links directly in the natural text flow of a DocBlock. If you want to display links to a tutorial prominently after descriptive text, use [{@tutorial}](#)

## Example

The inline {@tutorial} differs from ordinary non-inline tags. Inline tags parse and display their output directly in the documentation. In other words, this DocBlock:

```
1  /**
2  * Text with a normal @tutorial tag
3  * @tutorial phpDocumentor/phpDocumentor.pkg
4  */
5  function element()
6  {
7  }
```

Parses as (with -o HTML:frames:default):

### Example:

```
role = "html"<H3>element</H3>
<P><B>element ()</B></P>
<!-- ===== Info from phpDoc block ===== -->
<b></b>
<pre>
Text with a normal @tutorial tag
</pre>
<DL>
  <DT>Function Parameters:</DT>

  <DT>Function Info:</DT>
  <DD><b>Tutorial</b> - <CODE><a
  href="../phpDocumentor/tutorial_phpDocumentor.pkg.html">phpDocumentor 1.2.2
  Tutorial</a></CODE></DD>

</DL>
```

<HR>

```
1  /**
2   * Text with an inline link to {@tutorial phpDocumentor/phpDocumentor.pkg} displays
3   * without a break in the flow
4   */
5  function element()
6  {
7 }
```

Parses as (with -o HTML:frames:default):

**Example:**

```
role = "html"<H3>element</H3>
<P><B>element ()</B></P>
<!-- ===== Info from phpDoc block ===== -->
<b></b>
<pre>
Text with an an inline link to <a
href="../phpDocumentor/tutorial_phpDocumentor.pkg.html">phpDocumentor 1.2.2
Tutorial</a> displays without a break in the flow
</pre>
<DL>
<DT>Function Parameters:</DT>

<DT>Function Info:</DT>

</DL>
<HR>
```

# Writing a New Converter

*Overview of how to write a new Converter*

**Authors:** Joshua Eichorn

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

Gregory Beaver

[cellog@sourceforge.com](mailto:cellog@sourceforge.com)

## Introduction to Converters

This documentation deals only with the advanced programming topic of creating a new output converter. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to extend an existing Converter, read [Converter Manual](#)

## Basic Concepts

### Abstract Parsing Data

#### *Source Code Elements*

A Converter's job is to take abstract data from parsing and create documentation. What is the abstract data? phpDocumentor is capable of documenting tutorials, and a php file and its re-usable or structurally important contents. In other words, phpDocumentor can document:

- A XML DocBook-based tutorial (see [phpDocumentor Tutorials](#))
- Procedural Page (PHP source file)
- Include Statements
- Define Statements
- Global Variables
- Functions
- Classes
- Class Variables
- Class Methods

phpDocumentor represents these PHP elements using classes:

- [parserTutorial](#)
- [parserData](#), [parserPage](#)
- [parserInclude](#)
- [parserDefine](#)
- [parserGlobal](#)
- [parserFunction](#)
- [parserClass](#)
- [parserVar](#)

- [parserMethod](#)

This relationship between the source code and abstract data is quite clear, and very simple. The data members of each abstract representation correspond with the information needed to display the documentation. All PHP elements contain a DocBlock, and this is represented by a class as well, [parserDocBlock](#). The elements of a DocBlock are simple:

- Short Description, containing any number of inline tags
- Long Description, containing any number of inline tags
- Tags, some containing any number of inline tags in their general description field

phpDocumentor represents these elements using classes as well:

- [parserDesc](#) for both short and long descriptions
- [parserInlineTag](#) for inline tags
- [parserTag](#) for regular tags

### *HTML-specific issues*

There are some other issues that Converters solve. In HTML, a link is represented by an `<a>` tag, but in the PDF Converter, it is represented by a `<c:ilink>` tag. How can we handle both cases? Through another abstract class, the [abstractLink](#) and its descendants:

- [tutorialLink](#)
- [pageLink](#)
- [defineLink](#)
- [globalLink](#)
- [functionLink](#)
- [classLink](#)
- [varLink](#)
- [methodLink](#)

Note the absence of an "includeLink" class - this is intentional, as only re-usable elements need to be linked. An include statement is always attached to the file that it is in, and cannot be anywhere else.

These abstract linking classes contain the information necessary to differentiate between the location of any of the element's documentation. They are only used to allow linking to documentation, and not to source code. A link is then converted to the appropriate text representation for the output format by [Converter::returnSee\(\)](#) (a href=link for html, c:ilink:link for pdf, link linkend=link in xml:docbook, and so on).

The other issues solved by a converter involves html in a DocBlock. To allow better documentation, html is allowed in DocBlocks to format the output. Unfortunately, this complicates output in other formats. To solve this issue, phpDocumentor also parses out all allowed HTML (see [phpDocumentor Tutorial](#) for more detailed information) into abstract structures:

- `<b>` -- emphasize/bold text
- `<br>` -- hard line break, may be ignored by some converters
- `<code>` -- Use this to surround php code, some converters will highlight it
- `<i>` -- italicize/mark as important
- `<li>` -- list item
- `<ol>` -- ordered list

- <p> -- If used to enclose all paragraphs, otherwise it will be considered text
- <pre> -- Preserve line breaks and spacing, and assume all tags are text (like XML's CDATA)
- <ul> -- unordered list

is mapped to classes:

- [parserB](#)
- [parserBr](#)
- [parserCode](#)
- [parserI](#)
- [parserList](#) - both types of lists are represented by this object, and each <li> is represented by an array item
- [parserPre](#)

<p> is represented by partitioning text into an array, where each array item is a new paragraph. (as in [parserDocBlock::\\$processed\\_desc](#))

With these simple structures and a few methods to handle them, the process of writing a new converter is straightforward.

### *Separation of data from display formatting*

phpDocumentor has been designed to keep as much formatting out of the source code as possible. For many converters, there need be no new code written to support the conversion, as all output-specific information can be placed in template files. However, the complexity of generating class trees does require the insertion of some code into the source, so at the bare minimum, the [getRootTree\(\)](#) method must be overridden.

## **Methods that must be overridden**

Creating a new converter can be challenging, but should not be too complicated. You need to override one data structure, [Converter::\\$leftIndex](#), to tell the Converter which of the individual element indexes your Converter will use. You will also need to override a few methods for the Converter to work. The most important relate to linking and output.

A Converter must override these core methods:

- [Convert\(\)](#) - take any descendant of parserElement or a parserPackagePage and convert it into output
- [returnSee\(\)](#) - takes a abstract link and returns a string that links to an element's documentation
- [returnLink\(\)](#) - takes a URL and text to display and returns an internet-enabled link
- [Output\(\)](#) - generate output, or perform other cleanup activities
- [Convert\\_RIC\(\)](#) - Converts README/INSTALL/CHANGELOG file contents for inclusion in documentation
- [ConvertErrorLog\(\)](#) - formats errors and warnings from [\\$phpDocumentor\\_errors](#). see [HTMLframesConverter::ConvertErrorLog\(\)](#)
- [getFunctionLink\(\)](#) - for all of the functions below, see [HTMLframesConverter::getFunctionLink\(\)](#) for an example
- [getClassLink\(\)](#)
- [getDefineLink\(\)](#)
- [getGlobalLink\(\)](#)
- [getMethodLink\(\)](#)
- [getVarLink\(\)](#)

A Converter may optionally implement these abstract methods:

- [endPage\(\)](#) - do any post-processing of procedural page elements, possibly output documentation for the page
- [endClass\(\)](#) - do any post-processing of class elements, possibly output documentation for the class
- [formatIndex\(\)](#) - format the \$elements array into an index see [HTMLframesConverter::generateElementIndex\(\)](#)
- [formatPkgIndex\(\)](#) - format the \$pkg\_elements array into an index see [HTMLframesConverter::generatePkgElementIndex\(\)](#)
- [formatLeftIndex\(\)](#) - format the \$elements array into an index see [HTMLframesConverter::formatLeftIndex\(\)](#)
- [formatTutorialTOC\(\)](#) - format the output of a {@toc} tag, see [HTMLframesConverter::formatTutorialTOC\(\)](#)
- [getRootTree\(\)](#) - generates an output-specific tree of class inheritance
- [SmartyInit\(\)](#) - initialize a [Smarty](#) template object
- [writeSource\(\)](#) - write out highlighted source code for a parsed file
- [writeExample\(\)](#) - write out highlighted source code for an example
- [unmangle\(\)](#) - do any post-processing of class elements, possibly output documentation for the class

The following methods may need to be overridden for proper functionality, and are for advanced situations.

- [checkState\(\)](#) - used by the [parserStringWithInlineTags::Convert\(\)](#) cache to determine whether a cache hit or miss occurs
- [getState\(\)](#) - used by the [parserStringWithInlineTags::Convert\(\)](#) cache to save state for the next Convert() call
- [type\\_adjust\(\)](#) - used to enclose type names in proper tags as in [XMLDocBookConverter::type\\_adjust\(\)](#)
- [postProcess\(\)](#) - called on all converted text, so that any illegal characters may be escaped. The HTML Converters, for example, pass all output through <http://www.php.net/htmlentities>
- [getTutorialId\(\)](#) - called by the {@id} inline tag to get the Converter's way of representing a document anchor

## Converter methods an extended converter should use

- [getSortedClassTreeFromClass\(\)](#) -- generating class trees by package
- [hasTutorial\(\)](#) -- use this to retrieve a tutorial, or determine if it exists
- [getTutorialTree\(\)](#) -- use this to retrieve a hierarchical tree of tutorials that can be used to generate a table of contents for all tutorials
- [vardump\\_tree\(\)](#) -- use this to assist in debugging large tree structures of tutorials

# Converter Default Template Variables

*Basic Template Variables Available for Use by a New Converter*

## Introduction to Converter Template Variables

As of phpDocumentor version 1.2.0rc1, all Converters share certain template variables in common. This is made possible by the common use of the Smarty template engine. Smarty makes it possible to truly separate data from its display, allowing ultimate control of the display through the templates.

### Smarty

What is Smarty? This is a versatile compiling template engine, and has its home page at <http://smarty.php.net>, please read the manual and the tutorials if you are unfamiliar with Smarty. This tutorial only describes how to use the templates, and will be confusing if you have never used Smarty before. In particular, you must understand fully how to use the {section} and {foreach} tags. Please read <http://smarty.php.net/manual/en/language.function.section.php> and <http://smarty.php.net/manual/en/language.function.foreach.php>

All template files have available to them these smarty variables:

- {\$date} string - the current date and time, as formatted by [date\("r",time\)](#)
- {\$maintitle} string - the contents of the [phpDocumentor Tutorial](#) command-line switch
- {\$package} string - the package of the current template item
- {\$subpackage} string - the subpackage of the current template item
- {\$phpdocversion} string - the version number of phpDocumentor
- {\$phpdocwebsite} string - the official phpDocumentor website (<http://www.phpdoc.org>)

## Common Source Code Element Template Variables

Source code elements are described in [Documentable PHP Elements](#). For most converters, documented elements are split up into two categories: class and procedural page. The template variables that contain this information are also grouped in this manner.

The source code elements (function, include, global variable, define, class, class variable, method) all have a few template variables in common. They are:

- {\$\$desc} string - Summary from the DocBlock
- {\$\$desc} string - Long description from the DocBlock
- {\$tags} array - Tags found in the DocBlock, array('keyword' => tagname, 'data' => description). access in this manner:
  - {section name=sec loop=\$tags}{\$tags[sec].keyword} = {\$tags[sec].data}{/section}
  - or
  - {foreach from=\$tags item=tag}{\$tag.keyword} = {\$tag.data}{/foreach}
- {\$api\_tags} array - Tags found in the DocBlock useful in documenting the API, array('keyword' => tagname, 'data' => description). Access in the same manner as {\$tags}
- {\$info\_tags} array - Tags found in the DocBlock that are purely informational in nature, offering no particular use in documenting the actual software API, array('keyword' => tagname, 'data' => description). Access in the same manner as {\$tags}

- `[$utags]` array - Unknown tags found in the DocBlock, array('keyword' => tagname, 'data' => description). Access in the same manner as `[$tags]`
- `[$line_number]` integer - The line number in the source code that this element appears
- `[$id]` string - EXCEPT for include, all elements have a unique ID. This should be used to create a document anchor that will allow hyperlinking to that element's documentation.

## Procedural Page Conversion

### *Procedural Page Converter variables*

The Converter::ConvertPage() method sets up several important variables that are used by the other conversion methods. These are:

- `Converter::$page_data` - the Smarty template that contains all procedural page contents
- `Converter::$path` - the full path to the procedural page
- `Converter::$page` - the converter-safe name of the procedural page, can be used to uniquely name output
- `Converter::$curpage` - the [parserPage](#) representation of the current procedural page

### *Procedural Page template variables*

There are many template variables assigned to the procedural page template:

- `[$name]` string - the name of the file "file.php"
- `[$source_location]` string - the relative path of the file, needed for an include statement
- `[$classes]` array - an array of classes defined in the file, see [Converter::getClassesOnPage\(\)](#)
- `[$tutorial]` string|false - a link to the tutorial associated with this page, or false if none
- `[$sdesc]` string - summary of the procedural page from the Page-level DocBlock
- `[$desc]` string - long description of the procedural page from the Page-level DocBlock
- `[$tags]` array - array of all tags, array('keyword' => tagname, 'data' => description)
- `[$utags]` array - array of all unknown tags, array('keyword' => tagname, 'data' => description). This variable is unused in the templates packaged with phpDocumentor, and should be accessed in exactly the same manner that `[$tags]` is accessed.
- functions, includes, defines, globals array - these arrays contain all of the procedural elements defined in the file

## Function template variables

The function and method template variables are nearly identical. The only template variable that functions use that methods do not is `[$function_conflicts]`, as methods cannot have conflicts. The function template variables are:

- `[$function_name]` string - simple name of the function
- `[$params]` array - Function parameters, array('var' => parameter name, 'datatype' => parameter type, 'data' => parameter description). The last two items are pulled from a [@var](#) tag.
- `[$function_return]` string - data type that the function returns
- `[$function_conflicts]` array - array('conflictype' => 'function', 'conflicts' => array(link to docs of conflicting function 1, ...))
- `[$function_call]` string - pre-formatted function declaration, use as is
- `[$function_call]` array - special unformatted function declaration, use to customize for special purposes. All of the templates distributed with phpDocumentor use this template variable. Writing template code for this variable is complex. It may be best to copy over code from existing templates and modify it as needed.

The `[$function_call]` template variable is formatted by [parserFunction::getIntricateFunctionCall\(\)](#), and the documentation for that method describes its format in detail.

All functions are appended to the {\$functions} template variable of the procedural page that contains them. In version 2.0, there will be greater flexibility possible with locating template variables.

## Global variable template variables

Global variables templates are pretty simple. They contain the standard element template variables plus:

- {\$global\_name} string - Name of the global variable
- {\$global\_type} string - Data type of the global variable
- {\$global\_value} string - Initial value of the global variable as parsed from the source, if any
- {\$global\_conflicts} array|false - array('conflicttype' => 'global variables', 'conflicts' => array(link to docs of conflicting global variable 1, ...))

All global variables are appended to the {\$globals} template variable of the procedural page that contains them. In version 2.0, there will be greater flexibility possible with locating template variables.

## Define constants template variables

Define constant templates are also pretty simple. They contain the standard element template variables plus:

- {\$define\_name} string - Name of the constant
- {\$define\_value} string - Value of the constant as parsed from the source
- {\$define\_conflicts} array|false - array('conflicttype' => 'defines', 'conflicts' => array(link to docs of conflicting define 1, ...))

All define constants are appended to the {\$defines} template variable of the procedural page that contains them. In version 2.0, there will be greater flexibility possible with locating template variables.

## Include statements template variables

Include statement templates are the simplest of all. They contain the standard element template variables plus:

- {\$include\_name} string - the kind of include (include/require/include\_once/require\_once)
- {\$include\_value} string - The file included. If the file is parsed, a link to its documentation will be passed instead of just the name of the file.

All include statements are appended to the {\$includes} template variable of the procedural page that contains them. In version 2.0, there will be greater flexibility possible with locating template variables.

## Class Conversion

### *Class Converter Variables*

Like ConvertPage(), the Converter::ConvertClass() method sets up a few important variables that are used by the other conversion methods. These are:

- [Converter::\\$class\\_data](#) - the Smarty template that contains all class contents for this class
- [Converter::\\$class](#) - the name of the current class

## *Class Template Variables*

There are many template variables assigned to the procedural page template aside from the standard element template variables:

- {\$class\_name} string - the name of the class
- {\$package} string - the package of the class
- {\$source\_location} string - the include path to the file that contains this class
- {\$children} array - array('link' => link to class that extends this class, 'sdesc' => summary of child class, ...)
- {\$class\_tree} string - fully-formatted class tree (Converter-specific). Use unmodified.
- {\$conflict} array - array('conflictytype' => 'classes', 'conflicts' => array(link to docs of conflicting class 1, ...))
- {\$ivars} array - complex array of all inherited variables, see [Converter::getFormattedInheritedVars\(\)](#)
- {\$imethods} array - complex array of all inherited variables, see [Converter::getFormattedInheritedMethods\(\)](#)
- vars, methods array - these template variables will contain each of the class variables and method template variables.

## **Class variable template variables**

The Class variable template variables are:

- {\$var\_name} string - simple name of the class variable
- {\$var\_default} string - Default value of the class variable, if any
- {\$var\_type} string - data type of the variable, from the [@var](#) tag.
- {\$var\_overrides} array - array('link' => link to documentation for overridden class variable, 'sdesc' => summary of overridden class variable)

The {\$function\_call} template variable is formatted by [parserFunction::getIntricateFunctionCall\(\)](#), and the documentation for that method describes its format in detail.

All class variables are appended to the {\$vars} template variable of the class that contains them. In version 2.0, there will be greater flexibility possible with locating template variables.

## **Function template variables**

The function and method template variables are nearly identical. See [Function template variables](#) for details.

The template variables that methods use and functions do not use are:

- {\$constructor} boolean - true/false, true if the method is a constructor
- {\$descmethod} array - array('link'=>link to documentation for overriding method, 'sdesc'=> summary of the overriding method,...)
- {\$method\_overrides} array - array('link'=>link to documentation for overridden method, 'sdesc'=> summary of the overridden method)

All methods are appended to the {\$methods} template variable of the class that contains them. In version 2.0, there will be greater flexibility possible with locating template variables.

## **Tutorial template variables**

- {\$contents} string - Converted contents of the tutorial
- {\$title} string - Title of the tutorial
- {\$nav} string - true/false, Whether any navigation links to other tutorials exist
- {\$up} string - link to the parent tutorial of this tutorial
- {\$uptitle} string - title of the parent tutorial
- {\$prev} string - link to the previous tutorial in the tutorial hierarchy
- {\$prevtitle} string - title of the previous tutorial

- {\$next} string - link to the next tutorial in the tutorial hierarchy
- {\$nexttitle} string - title of the next tutorial

# Writing a Converter, Methods

*Learn what methods need to be defined in a new Converter*

## Introduction to Converter Methods

This Tutorial describes the methods that a New Converter must use to become fully functional. A great deal of functionality is handled by the parent Converter class, and a child converter's only job is to convert sorted data from abstract classes into full-fledged output.

There are literally no restrictions on how a converter does its job. If the predefined [Converter::walk\(\)](#) method does not do anything a converter needs to do, all one needs to do is override walk() and perform any necessary processing directly on the parsed elements. Be forewarned that this is a bit like trying to fix a problem in a program by tinkering with the operating system, and we cannot be even in the slightest responsible for any suffering you might incur.

Having said that nasty disclaimer, let's begin with the good news. The only thing a standard converter needs to know is that it must define a few methods. The only crucial methods to define are [Convert\(\)](#), [returnSee\(\)](#), and [returnLink\(\)](#).

The Convert() method expects it will be passed all parsed data. Data will be passed either by file or by package, but in either case, all data will pass through Convert(), and so must be handled by it. Good programming practice suggests having one method for every parsed element, and that is what the bundled Converters do.

returnSee() is the core of phpDocumentor's advanced linking [@see](#), [@tutorial](#), [inline {@tutorial}](#) and [inline {@link}](#). returnLink() handles hyperlinks to URLs on the internet.

Beyond these three essential methods, there are a slew of methods that assist phpDocumentor in creating its advanced source highlighting feature, in creating an error log and todo list, and a few advanced functions that speed up conversion through a cache. Beyond this, there are many helper functions that assist in the creation of indexes, class trees, inheritance and conflict information, and so on. phpDocumentor does not restrict their use, to allow for future possibilities.

For most cases, it is best to start writing a new converter by copying the code from one of the existing converters, and then begin modifying methods to generate the appropriate output. In addition, this will leverage the existing use of the Smarty template engine to separate specifics of output formatting from the data source. In theory, one could create a single converter and use many templates for creating different output, but this is often not possible because templates are still organized into files, and this is invariant. In the future of phpDocumentor, we may have a base converter from which all other converters can extend, greatly simplifying development. For now, development is still relatively easy, after the basic concepts behind phpDocumentor are grasped.

## Converter core methods that must be overridden

## Convert()

The [Converter::Convert\(\)](#) method is called by [Converter::walk\(\)](#) to process all Converter classes:

- [parserData](#) - representation of a file
- [parserInclude](#) - representation of include statements
- [parserDefine](#) - representation of define statements
- [parserFunction](#) - representation of functions
- [parserGlobal](#) - representation of global variables
- [parserClass](#) - representation of a class
- [parserVar](#) - representation of class variables
- [parserMethod](#) - representation of class methods
- [parserPage](#) - representation of old Package Pages (deprecated)
- [parserTutorial](#) - representation of tutorials (like what you are reading now)

It is up to this method to distribute processing of these elements, or do any post-processing that is necessary. All of the converters included with phpDocumentor process elements by passing them to individual Convert\*() methods like [HTMLframesConverter::ConvertClass\(\)](#), and one can simply copy this style, or write a completely new method.

Data is passed to the Converter organized by file, procedural elements first followed by class elements, unless [Converter::\\$sort\\_absolutely\\_everything](#) is set to true. Then data is passed organized by package, with all files and procedural elements first, followed by all class elements. The PDFdefaultConverter set \$sort\_absolutely\_everything = true, and HTML converters set \$sort\_absolutely\_everything = false

## returnSee()

This method takes a [abstractLink](#) class descendant and converts it to an output-format compatible link to an element's documentation. HTML converters convert the link to an <a href=""> tag, the XML converter changes it to a <link linkend=""> tag, etc. This function is also responsible for distinguishing between sections of documentation. All information needed to distinguish between elements is included in the data members of a link class, it is up to the returnSee() method to translate that into a unique string identifier. A good strategy is to write a function that takes a link class and returns a unique identifier as in [XMLDocBookConverter::getId\(\)](#), and then reference this function to grab identification strings for defining anchors in the generated output.

## returnLink()

This method takes a URL and converts it to an external link to that URL in the appropriate output format. In HTML, the link will be a standard <a href= tag.

## Output()

This method is called at the end of the walk() method. It may be used to generate output or do any necessary cleanup. Nothing is required of Output, and it may do nothing as it does in the HTML converters, which write output continuously as converting, or perform special operations as in [CHMdefaultConverter::Output\(\)](#).

## Convert\_RIC()

This method is called to format the contents of README, INSTALL, CHANGELOG, NEWS, and FAQ files grabbed from the base parse directory. This function allows standard distribution files to be embedded in generated documentation. A Converter should format these files in a monospace font, if possible.

## ConvertErrorLog()

This method is called at the end of parsing to convert the error log into output for viewing later by the developer. Error output is very useful for finding mistakes in documentation comments. A simple solution is to copy the [HTMLframesConverter::ConvertErrorLog\(\)](#) function and the errors.tpl template file to the new converter. The error log should not be embedded in generated output, as no end-user wants to see that information.

## **getFunctionLink()**

This function should call Converter::getFunctionlink(), and then use [returnSee\(\)](#) to return a string containing a converter-specific link. Code can literally be copied from any of the converters. This function is called by the [parserFunction::getLink\(\)](#) method.

## **getClassLink()**

This function should call Converter::getClasslink(), and then use [returnSee\(\)](#) to return a string containing a converter-specific link. Code can literally be copied from any of the converters. This function is called by the [parserClass::getLink\(\)](#) method.

## **getDefineLink()**

This function should call Converter::getDefinelink(), and then use [returnSee\(\)](#) to return a string containing a converter-specific link. Code can literally be copied from any of the converters. This function is called by the [parserDefine::getLink\(\)](#) method.

## **getGlobalLink()**

This function should call Converter::getGloballink(), and then use [returnSee\(\)](#) to return a string containing a converter-specific link. Code can literally be copied from any of the converters. This function is called by the [parserGlobal::getLink\(\)](#) method.

## **getMethodLink()**

This function should call Converter::getMethodlink(), and then use [returnSee\(\)](#) to return a string containing a converter-specific link. Code can literally be copied from any of the converters. This function is called by the [parserMethod::getLink\(\)](#) method.

## **getVarLink()**

This function should call Converter::getVarlink(), and then use [returnSee\(\)](#) to return a string containing a converter-specific link. Code can literally be copied from any of the converters. This function is called by the [parserVar::getLink\(\)](#) method.

# **Abstract Methods that must be overridden**

## **endPage()**

This method is called by the [Converter::walk\(\)](#) method when all procedural and class elements on a page have been processed. The purpose is to allow the converter to reset any state variables that apply to the page. `endPage()` is not called by [Converter::walkEverything\(\)](#) and so if your converter sets [Converter::\\$sort\\_absolutely\\_everything = true](#), you will not need to implement this method.

## **endClass()**

This method is called by the [Converter::walk\(\)](#) method when all class elements (methods, variables) in a class have been processed. The purpose is to allow the converter to reset any state variables that apply to the page. `endPage()` is not called by [Converter::walk everything\(\)](#) and so if your converter sets [Converter::\\$sort\\_absolutely\\_everything = true](#), you will not need to implement this method.

## **formatIndex()**

This method is called before processing any elements, and is not required to do anything. The intent is to allow processing of a global element index to occur in a separate method, which logically separates activities performed by the Converter. See the [HTMLframesConverter::formatIndex\(\)](#) method for details on one possible implementation

## **formatPkgIndex()**

Like `formatIndex()`, this method is called before processing any elements, and is not required to do anything. The intent is to allow processing of a package-level index to occur in a separate method, which logically separates activities performed by the Converter. See the [HTMLframesConverter::formatPkgIndex\(\)](#) method for details on one possible implementation

## **formatLeftIndex()**

Like `formatIndex()`, this oddly-named method is called before processing any elements, and is not required to do anything. The name comes from the original JavaDoc design of putting an index in the lower left frame of output. The indexes needed by this function are automatically generated based on the value of [Converter::\\$leftindex](#). These indexes are arrays of links organized by package and subpackage. Left indexes can be generated for files (procedural pages), functions, classes, constants (define statements), and global variables.

## **formatTutorialTOC()**

This method is used to format the automatically generated table of contents from an [inline {@toc}](#) in a tutorial. The data structure passed is an array generated by [parserTocInlineTag::Convert\(\)](#) that contains each entry. All that `formatTutorialTOC` needs to do is structure the inherent hierarchy of the original DocBook tutorial source according to the requirements of the output format. This is remarkably simple, and can often be implemented simply by passing the entire array to a template engine, as `HTMLframesConverter::formatTutorialTOC()` does.

## **getRootTree()**

This is a very fast non-recursive method that generates a string containing class trees. In earlier `phpDocumentor` versions, the `getRootTree()` method was recursive, and both consumed too much memory and was very slow. It was also easier to understand. The data structure processed in the current version of `phpDocumentor` still contains the same information, but requires a precise algorithm to process. The good news is that we have worked out this algorithm for you, and to implement this method in a new converter, you need only copy the code from one of the existing converters. See `Converter::getRootTree()` for low-level details.

## **SmartyInit()**

The primary template engine for `phpDocumentor` is the [Smarty](#) template engine. This utility function is used to initialize a new Smarty template object with pre-initialized variables. The `Converter::newSmarty` initializes several variables, see [Converter::newSmarty\(\)](#) for details. If your converter needs to have variables available to every template, extend this function, and use [Smarty::assign\(\)](#) to assign values to variables. Note that the

function must use code that returns a reference like:

```
function &SmartyInit(&$templ)
{
    $somevalue = "whatever you want, babe";
    $templ->assign("variable", $somevalue);
    return $templ;
}
```

## **writeSource()**

## **writeExample()**

## **unmangle()**

This function is used by the pre-PHP 4.2.3 [inline {@source}](#) tag's parserInlineSourceTag::Convert() method to process output from [http://www.php.net/highlight\\_string](http://www.php.net/highlight_string). In non-HTML converters, it should return an empty string. For advanced source highlighting, implement highlightSource(), highlightDocBlockSource().

# **Methods that may optionally be overridden**

## **checkState()**

## **getState()**

## **type\_adjust()**

This method is passed type names from [@param](#)/[@return](#)/[@var](#) tags and can be used to enclose types in special formatting. The only converter that really uses this capability is the DocBook converter. Since the type name can be a link class, it is possible to determine what kind of element the type is, a constant, class, or anything that can be documented. For example, the DocBook converter encloses classes in `<classname>` tags, constants in `<constant>` tags, variables in `<varname>` tags, and also will enclose functions in `<function>` tags.

## **postProcess()**

This method takes a string and should "escape" any illegal characters. For instance, in HTML, all "<" characters must be escaped to "<" entities. This is to prevent documentation comments from being interpreted as text modifiers in the output format.

## **getTutorialId()**

This method is used to pass back a valid ID for a tutorial section. In HTML, most of the id is coded in the file containing the data, and only the section and subsection need to be used to differentiate sections. In DocBook, the id is packages.categoryname.packagename.subpackagename.class/filename.sectionsubsection, a much more involved scenario.

## Utility methods

### **getSortedClassTreeFromClass()**

This method is used by [getRootTree\(\)](#) to create class trees for a package. Also see the docs in [Converter::getSortedClassTreeFromClass\(\)](#)

### **hasTutorial()**

This helper method is used to determine whether a particular tutorial has been parsed, and can be used to perform association between parsed elements and their tutorials

### **getTutorialTree()**

This helper method returns a recursive data structure containing all tutorials in the table-of-contents style hierarchy that they should have in final output. This is used to create a table of contents for tutorials in the left frame of the HTML Converters, and in the PDF Converter, to determine indentation level in the top-level table of contents.

### **vardump\_tree()**

This function is a utility function. It performs what looks like a [http://www.php.net/var\\_dump](http://www.php.net/var_dump) of any recursive data structure, but if it encounters an object, it only prints out its class and name if the \$name data member exists. This allows checking of the tree structure without having to wade through pages of class information, and was very useful for debugging, so we added it for your use.

# Converter Manual

*Learn how to use and extend a Converter*

## [phpDocumentor](#) Converters Documentation

Detailed Documentation of the tag specification is available at [phpDocumentor tags](#). Detailed information on phpDocumentor and a tutorial on basic usage is at: [phpDocumentor Tutorial](#)

If you are wondering how to extend the capabilities of phpDocumentor's output, this is the documentation you need. This documentation does not discuss how to document your code or project, and only deals with the specific requirements of extending phpDocumentor.

## **Internals: how phpDocumentor takes source code and generates documentation**

phpDocumentor is divided into three logical components: [Parser](#), [IntermediateParser](#) and [Converter](#). These three divisions communicate through special method interfaces.

### **Parser internals**

The Parser component does the work of reading the actual files, both source code and tutorial/manual files. For a detailed discussion of how to write a tutorial, see [phpDocumentor Tutorials](#), or read the source files for phpDocumentor's manual in the tutorials/ subdirectory. The Parser encapsulates input into abstract classes, all of which are defined in seven files:

- [DescHTML.inc](#)
- [DocBlockTags.inc](#)
- [InlineTags.inc](#)
- [ParserData.inc](#)
- [ParserElements.inc](#)
- [ParserDocBlock.inc](#)
- [PackagePageElements.inc](#)

The abstract classes are then passed to the Intermediate Parser.

The IntermediateParser class organizes and processes abstract data passed in with the assistance of the [ProceduralPages](#) and [Classes](#) classes. All data is sorted alphabetically and then passed to the converters specified on the command-line one by one. Traditionally, the next step would be to generate output by passing the abstract data to a template engine. phpDocumentor has one more layer of separation to accomodate different output formats called "Converters."

What is a Converter? To understand this, one must first understand the problem that Converters solve. Documentation is not always best viewed as a web page. Sometimes a user may want to print it out, or view it in XML with an interface program that can search or perform other advanced functions. Printing html does not work very well, and may look very different on different user's screens. To solve this problem, we need an interface between the parser and the template engine, hence the Converter.

On an even more basic level, the linking performed by phpDocumentor requires that the program pass over the data at least twice to set up relationships and create the @see, @tutorial, {@tutorial} and {@link} hyperlinks that make it such an effective tool. In earlier versions of phpDocumentor, the parser passed over the data twice, with significant redundancy and slow parsing times. In addition, the linking pass had to be first, and the order of parsing was important. In other words, if file A contains class B extends foo, and file B contains class foo, if file A is parsed before file B, no inheritance occurs between the classes (you can try this with phpDocumentor 0.4.2a and earlier, it breaks the parser). The Converter works with temporary data structures (all defined in ParserData.inc, ParserElements.inc, and ParserDocBlock.inc if you want a peek), and allows the linking pass to occur in-memory after parsing, with a significant speedup (almost twice as fast as earlier versions).

## Adding a template

In most cases, it is possible to leverage the existing code to create spectacular output simply by writing a new set of templates for one of the existing Converters. To learn how to write templates for each specific converter, choose from the list below:

- [Writing templates for the HTMLframes Converter](#)
- [Writing templates for the HTMLSmarty Converter](#)
- [Writing templates for the PDFdefault Converter](#)
- [Writing templates for the XMLDocBook Converter](#)
- [Writing templates for the CHMdefault Converter](#)

## Extending an existing Converter by writing your own Converter Class

If the existing converters work except for a small detail that could be fixed by adding a small patch to the source code of an output converter, then you need to extend a Converter. There are several straightforward rules that allow you to easily extend a Converter.

### Rules for extending a Converter

First, the extended converter must be defined by a single word that would work as a valid filename. For an example, if the Converter extends the HTMLframesConverter, and provides search functionality in the generated documentation through the insertion of special php code into the generated files, it might be named "Search."

Once a name has been chosen, a subdirectory of the parent converter (HTMLframesConverter is in phpDocumentor/Converters/HTML/frames/) must be created with the same name as the child converter. So, for our "Search" Converter, the directory phpDocumentor/Converters/HTML/frames/Search/ must be created. Put your new Converter in a file called "**HTMLframes SearchConverter.inc**". Note the insertion of our new Converter's name before the word "Converter" in the filename.

Next, create a class declaration in the new HTMLframesSearchConverter.inc file as below:

```
1  /**
2  * HTML output converter.
3  *
4  * This Converter takes output from the {@link Parser} and converts it to HTML-ready output for use with
```

```

{@link Smarty}.
5  * Add any extra documentation about the search capabilities here
6  * @package Converters
7  * @subpackage HTMLframes
8  * @author Your Name <example@example.com>
9  */
10 class HTMLframesSearchConverter extends Converter
11 {
12 /**
13  * frames/Search Converter wants elements sorted by type as well as alphabetically
14  * @see Converter::$sort_page_contents_by_type
15  * @var boolean
16  */
17 var $sort_page_contents_by_type = true;
18 /** @var string */
19 var $outputformat = 'HTML';
20 /** @var string */
21 var $name = 'frames/Search';

```

Note the var \$name is set to the relative subdirectory. This is very important. Do not set the name to be "framesSearch", as the [Converter::setTemplateDir\(\)](#) method uses this variable to set the template directory.

If the existing templates work fine, and there is some other change needed, set the name to be the same as the parent, and phpDocumentor will use the parent templates.

After extending the methods the new Search Converter will need, place the templates in a subdirectory named templates/ (as in phpDocumentor/Converters/HTML/frames/Search/templates/), just as they are used in the converters that come with phpDocumentor.

## Using an extended Converter with phpDocumentor

After all this complexity of setting up the Converter, using it is straightforward. simply pass -o HTML:frames/Search:templatename to phpDocumentor's web interface or commandline!

## Writing a Converter for a new output format

This topic is very large, and a separate manual entry is devoted entirely to this subject, [Writing a New Converter](#)

# Using the PDFParser XML templating language

*Learn how the PDFParser XML templating language is constructed and used*

## PDFParser Introduction

This documentation deals only with the PDFParser XML templating language. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to write a new Converter, read [Converter Manual](#)

## Overview

The PDF:default Converter uses the powerful Cezpdf class library written by Wayne Munro. Although the Cezpdf library is very versatile and useful, it is designed to be used as PHP code, and does not allow access to its methods from the text passed to it. The PDFParser class was written to solve this problem. Through the use of a few basic XML-based tags, and the versatile <pdffunction /> tag, one can access any feature of the Cezpdf library and do anything that it can do. This is what makes it possible to use Smarty templates without modifying any code in phpDocumentor, a primary goal of the 1.2.0 release.

## PDFParser XML tags

### <text>

The <text> tag is a block tag (requires a closing tag). All output that is not a PDFParser tag must be enclosed in this tag. These are the parameters to the text tag:

- size (**required**) - font size
- any legal parameter to [Cezpdf::ezText\(\)](#) (all optional):
- left - indentation from the left margin
- right - indentation from the right margin
- aleft - absolute indentation from the left side of the page
- aright - absolute indentation from the right side of the page
- justification - one of left, right, center/centre, full
- either leading or spacing

Examples might include:

- <text size="11">Hello World!</text>
- <text size="11" justification="right">Hello World!</text>
- <text size="11" left="10" right="10" justification="center">Hello World!</text>

### <font>

Unlike HTML, the <font /> tag is a self-enclosed tag, and is only used to select the font face (and future

versions will also select color). Syntax is <font face="fontface" /> The fontface must be present in the Converters/PDF/default/templates/fonts/ subdirectory as a fontface-php.afm and fontface.afm file, otherwise Cezpdf will cause all kinds of PHP warnings. This is an issue with Cezpdf that we are assisting the author in fixing, and will not be a problem in the future.

## <newpage>

This is the simplest tag, use it as <newpage /> to tell Cezpdf to jump to the top of the next page in output.

## <pdffunction>

The <pdffunction> tag is the most versatile of the PDFParser tags. This tag can be used to arbitrarily call any Cpdf or Cezpdf method. Arguments can be passed to the function both as literal values and using dynamic variables, both class variables and temporary variables. The method chosen is selected as if pdffunction were a namespace rather than a

Here are the parameters to the pdffunction:

- return (**optional**) set a temporary variable to the value returned from the function. This parameter allows a later reference to the temporary variable like: <pdffunction:getYPlusOffset return="newy" offset="0" /> <pdffunction:addJpegFromFile arg="logo.jpg" x="250" y=\$newy />

any other parameter represents an argument. Since the parameters passed are simply passed by value to the method, the names can be anything that is useful to understanding how they work.

<pdffunction:ezStartPageNumbers x="500" y="28" size="10" pos="" pattern="{ldelim}PAGENUM{rdelim} of {ldelim}TOTALPAGENUM{rdelim}" num="1" /> could just as easily be <pdffunction:ezStartPageNumbers arg="500" foo="28" bar="10" mylord="" goodnessgracious="{ldelim}PAGENUM{rdelim} of {ldelim}TOTALPAGENUM{rdelim}" num="1" /> and the code would still work splendidly. The first way is obviously easier to handle, and so is recommended.

Most arguments to the method will be literals, like numbers or strings, but some arguments may need to be more dynamic, based on the values returned from other pdffunction tags. To do this, use the return parameter as described above, and reference the temporary variable as if it were a php variable like <pdffunction:whatever arg=\$tempvar />. It may even be necessary to access a data member of the pdf class. In this case, use \$this->datamember as in <pdffunction:whatever arg=\$this->datamember />

# Writing templates for the PDFdefault Converter

*Learn which template variables are available for use in an HTML:frames template*

**Authors:** Joshua Eichorn

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

Gregory Beaver

[cellog@sourceforge.com](mailto:cellog@sourceforge.com)

## PDF:default:\* Introduction

This documentation deals only with adding a new template to the PDFdefault Converter. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to write a new Converter, read [Converter Manual](#)

# Writing templates for the CHMdefault Converter

*Learn which template variables are available for use in an CHM:default template*

**Authors:** Joshua Eichorn

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

Gregory Beaver

[cellog@sourceforge.com](mailto:cellog@sourceforge.com)

## CHM:default:\* Introduction

This documentation deals only with adding a new template to the CHMdefault Converter. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to write a new Converter, read [Converter Manual](#)

To generate a CHM file, go to <http://go.microsoft.com/fwlink/?LinkId=14188> and download Microsoft's Help Workshop.

# Writing templates for the XMLDocBook Converter

*Learn which template variables are available for use in an XML:DocBook template*

**Authors:** Joshua Eichorn

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

Gregory Beaver

[cellog@sourceforge.com](mailto:cellog@sourceforge.com)

## XML:DocBook: \* Introduction

This documentation deals only with adding a new template to the XMLDocBook Converter. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to write a new Converter, read [Converter Manual](#)

# Writing templates for the HTMLSmarty Converter

*Learn which template variables are available for use in an HTML:Smarty template*

**Authors:** Joshua Eichorn

[jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)

Gregory Beaver

[cellog@sourceforge.com](mailto:cellog@sourceforge.com)

## HTML:Smarty: \* Introduction

This documentation deals only with adding a new template to the HTMLSmarty Converter. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to write a new Converter, read [Converter Manual](#)

# Writing templates for the HTMLframes Converter

*Learn which template variables are available for use in an HTML:frames template*

## **HTML:frames:\*** Introduction

This documentation deals only with adding a new template to the HTMLframes Converter. To learn how to use phpDocumentor, read the [phpDocumentor Guide to Creating Fantastic Documentation](#). To learn how to write a new Converter, read [Converter Manual](#)

## Overview

The HTML:frames converter uses all of the values from the generic Converter (see [Converter Default Template Variables](#)), and adds a few more.

All templates have the standard smarty template variables available to them, as well as these:

- {\$subdir} string - the combination of "../" that will lead to the base output directory, used for linking to other generated documentation.
- {\$packageindex} array - a list of all the packages and their index pages in the format array('link' => link to the index page, 'title' => package name)

## Template Variables

### **Source Code formatting - filesource.tpl**

When source code is generated via the [@filesource](#) tag, the output is passed into these templates variables:

- {\$source} string - the fully formatted source code, use as is
- {\$package} string - the package name of the source file
- {\$subpackage} string - the subpackage name of the source file
- {\$name} string - the name of the source file file.ext
- {\$source\_loc} string - the relative path to the file, used for include() statements
- {\$docs} string - a link to the documentation for this file

### **Example Code formatting - examplesource.tpl**

When an example file is specified via the [@example](#) tag, the output is passed into these templates variables:

- {\$source} string - the fully formatted source code, use as is
- {\$file} string - the full path to the file
- {\$title} string - the title of this example, from the @example tag

### **Left Frame formatting - left\_frame.tpl**

The HTML:frames Converter places a table of contents in the left frame. This table of contents contains links to class trees, alphabetical indexes, tutorials, and elements organized by category (class/procedural code).

### *Template variables passed to left\_frame.tpl*

Each left\_frame.tpl consists of one package's documentation. Basic template variables passed to left\_frame.tpl include:

- {\$info} array - the table of contents, see [The {\\$info} template variable](#) for contents
- {\$package} string - package this table of contents documents
- {\$hastutorials} boolean - true if this package has external tutorials (see [phpDocumentor Tutorials](#))
- {\$hastodos} boolean - true if there are any [@todo](#) tags in this package
- {\$todolink} string - file.html name of the todo list.
- {\$classtreepage} string - name of the file containing class trees, minus .html
- {\$elementindex} string - name of the file containing the alphabetical index for this package, minus .html
- {\$packagedoc} string|false - link to package-level documentation, or false if there is none

### *The {\$info} template variable*

The {\$info} variable is an array of arrays, each indexed by subpackage. Even packages with no explicit [@subpackage](#) tags have a subpackage named " (empty string). Elements in the main package can be detected using the smarty code:

```
{section name=p loop=$info}{if $info[p].subpackage == ""}...
```

The {\$info} variable contains these sub-variables (assuming that {section name=p loop=\$info} was used):

- {\$info[p].package} string - package name
- {\$info[p].subpackage} string - subpackage name
- {\$info[p].packagedoc} string - link to old format .html-based package-level documentation (deprecated)
- {\$info[p].packagetutorial} string - link to new package-level documentation, or subpackage-level documentation for subpackages
- {\$info[p].packagetutorialhoa} string - link to new package-level documentation, or subpackage-level documentation for subpackages, without enclosing <a href="link to docs"> HTML tag. Use this to do special link handling (opening a new window, opening in the \_right frame, etc.)
- {\$info[p].packagetutorialtitle} string - title of package-level documentation
- {\$info[p].files}/{\$info[p].classes}/{\$info[p].functions} array - links to documentation array('link' => html link, 'title' => name of element)
- {\$info[p].tutorials} string - Formatted tutorial table of contents, see [Tutorial table of contents formatting - tutorial\\_tree.tpl](#)

### *Tutorial table of contents formatting - tutorial\_tree.tpl*

The tutorial table of contents is formatted using the recursive tutorial\_tree.tpl template. This template has only a few variables:

- {\$name} string - abstract name of the tutorial (tutorial.pkg becomes tutorialpkg)
- {\$parent} string|false - {\$name} from parent tutorial, or false if no parent tutorial
- {\$main.title} string - the title of the tutorial. Use as {main.title|striptags}
- {\$main.link} string - link to the tutorial's documentation
- {\$haskids} boolean - true if this tutorial has sub-tutorials
- {\$kids} string - fully formatted list of child tutorials
- {\$subtree} boolean - true if this is a sub-tutorial tree. Use to modify formatting for sub-tutorial listings.

The tutorial\_tree.tpl file should represent the hierarchy of tutorials through some form of a nested html tag, such as a nested list (like the default converter), or a nested DHTML menu (like the phpedit or earthli templates).

## Class trees - classtrees.tpl

Class trees are a visual representation of Class inheritance to help facilitate understanding of how a project is organized. Class trees have only a few template variables:

- {\$package} string - the package that base classes are found in
- {\$classtrees} array - a pre-formatted list of class trees, organized by base class. Format is: array('class' => name of class, 'classtree' => fully formatted class tree)

## Package index - top\_frame.tpl

The top frame is used to display links to the highest level of documentation, the package table of contents, and the project-wide README/CHANGELOG/FAQ/NEWS/INSTALL files.

- {\$packages} array - A listing of all packages documented, in format array('link' => name of the left\_frame.tpl-generated package table of contents, 'title' => name of the package)
- {\$ric} array - all of the README/INSTALL/FAQ/NEWS/CHANGELOG files found, in format array('file' => name of the generated output, 'name' => displayable name of the parsed file)

## Alphabetical Indexes - elementindex.tpl/pkgelementindex.tpl

phpDocumentor's generated alphabetical element indexes are organized into an index of all elements in all packages (elementindex.tpl), and indexes of the elements in each individual package (pkgelementindex.tpl). Their variables are the same, except for the addition of the {\$package} variable in pkgelementindex.tpl. Here are the template variables:

- {\$index} array - the index items, see [{\\$index} contents](#)
- {\$letters} array - a listing of each letter heading that should be used in the index

### *{\$index} contents*

The {\$index} variable is an array of index listings organized by their first letter. All documented elements whose name begins with the letter A will be grouped into the sub-array 'a' => array(listing,listing,...). The listing is organized as follows:

- {\$index[xxx].name} string - name of the documented element
- {\$index[xxx].title} string - One of Variable, Global, Method, Function, Constant, Page, Include
- {\$index[xxx].link} string - hyperlink to the element's documentation, or the file included for include statements
- {\$index[xxx].listing} string - A fully-formatted index listing in English.

## Miscellaneous Templates - ric.tpl/todolist.tpl/errors.tpl

These specialized templates will only be used in certain cases.

- ric.tpl - This template is only used when files such as README, INSTALL, or CHANGELOG are detected in the base parse directory.
- todolist.tpl - This template is only used when a [@todo](#) tag is used in documentation
- errors.tpl - This template is always generated, but only contains content when there are parsing syntax errors or logical conflicts in the parsed documentation

### *ric.tpl - README/INSTALL/CHANGELOG display*

This is a very simple template.

- {\$name} string - name of the file (README, for example)
- {\$contents} string - contents of the file, ready for display (no modification needed)

### *todolist.tpl - listings of @todo tags*

This is also a very simple template, and requires only a {section} tag to process the input.

- {\$todos} array - Format: array(packagename => array('link' => string element containing @todo tag, 'todos' => array(todo item 1, todo item 2,...),...))

### *errors.tpl - phpDocumentor parsing/conversion warnings and errors*

This file is a tool used to help find mistakes in your own documentation, and greatly simplifies determining whether there is a bug in your documentation or in phpDocumentor itself.

The structure of the errors array passed to errors.tpl is somewhat complex to learn, but easy to use once learned. Every error or warning is grouped by the file that triggered it. In the case of post-parsing errors, they are grouped into the pseudo-file "Post-parsing". Within each file, warnings are grouped separately from errors. Hence, the structure looks something like:

```
1  array('Post-parsing' => array(  'warnings' => array(
2      array('name' =>  name of warning,
3          'listing' =>  description of warning),...
4      )
5      'errors' => array(
6          array('name' =>  name of error,
7              'listing' =>  description of error),...
8          )
9      ),
10     'file1.php' => array(  same structure  as Post-parsing),...
11 );
```

- {\$all} array - This is the structure listed above
- {\$files} array - a listing of all the files, useful for making a table of contents
- {\$title} string - this is hard-coded in English. Override by simple using {assign var="title" value="whatever you want"} before including header.tpl

## **PHP element templates - differences from Standard Converter**

For the basic PHP elements, define, include, function, global variable, class, and so on (see [Documentable PHP Elements](#)), there are very few differences between the templates packaged with the HTMLframesConverter and the basic Converter.

### *Class variables - var.tpl*

The only additional variable is:

- {\$var\_dest} string - html anchor used to pinpoint this class variable's documentation in the class file's documentation. Use as in: <a name="#{\$var\_dest}"></a>

### *Class methods - method.tpl*

The only additional variable is:

- {\$method\_dest} string - html anchor used to pinpoint this class method's documentation in the class file's documentation. Use as in: <a name="#{\$method\_dest}"></a>

### *Functions - function.tpl*

The only additional variable is:

- {\$function\_dest} string - html anchor used to pinpoint this function's documentation in the procedural file's documentation. Use as in: <a name="#{\$function\_dest}"></a>

### *Defines - define.tpl*

The only additional variable is:

- {\$define\_link} string - html anchor used to pinpoint this define statement's documentation in the procedural file's documentation. Use as in: <a name="#{\$define\_link}"></a>

### *Global Variables - global.tpl*

The only additional variable is:

- {\$global\_link} string - html anchor used to pinpoint this global variable's documentation in the procedural file's documentation. Use as in: <a name="#{\$global\_link}"></a>

### *Include Statements - include.tpl*

The only additional variable is:

- {\$include\_file} string - html anchor used to pinpoint this include statement's documentation in the procedural file's documentation. Use as in: <a name="#{\$include\_file}"></a>



# Package Converters Procedural Elements

## Converter.inc

### Base class for all Converters

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2001-2006 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Author** Greg Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: Converter.inc 287891 2009-08-30 08:08:00Z ashnazg \$
- **Copyright** 2001-2006 Gregory Beaver
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- Since 1.0rc1
- Filesource [Source Code for this file](#)
- License [LGPL](#)

*string* function adv\_htmlentities(\$s) [line [5404](#)]

**Function Parameters:**

- *string* \$s

### **smart htmlentities, doesn't entity the allowed tags list**

Since version 1.1, this function uses htmlspecialchars instead of htmlentities, for international support This function has been replaced by functionality in [ParserDescCleanup.inc](#)

- **Deprecated** As of v1.2, No longer needed, as valid tags are parsed out of the source, and everything else is [Converter::postProcess\(\)](#) handled

include\_once "phpDocumentor/Smarty-2.6.0/libs/Smarty.class.php" [line [43](#)]

## **Smarty template files**

# Package Converters Classes

## Class Converter

[line 74]

### Base class for all output converters.

The Converter marks the final stage in phpDocumentor. phpDocumentor works in this order:

Parsing => Intermediate Parsing organization => Conversion to output

A Converter takes output from the [phpDocumentor IntermediateParser](#) and converts it to output. With version 1.2, phpDocumentor includes a variety of output converters:

- [HTMLframesConverter](#)
- [HTMLSmartyConverter](#)
- [PDFdefaultConverter](#)
- [CHMdefaultConverter](#)
- CSVdia2codeConverter
- [XMLDocBookConverter](#)

{@ and using [walk\(\)](#) or [walk everything](#) (depending on the value of [\\$sort absolutely everything](#)) it "walks" over an array of phpDocumentor elements.}}

- **Package** Converters
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: Converter.inc 287891 2009-08-30 08:08:00Z ashnazg \$
- **Abstract Element**
- **Since** 1.0rc1

### Converter::\$all\_packages

array = [line 366]

## All packages encountered in parsing

- See [phpDocumentor\\_IntermediateParser::\\$all\\_packages](#)

### Converter::\$class

*string|false = false [line 105]*

**set to a classname if currently parsing a class, false if not**

### Converter::\$classes

*Classes = [line 319]*

**All class information, organized by path, and by package**

### Converter::\$class\_contents

*array = array() [line 269]*

**alphabetical index of all methods and vars in a class by package/subpackage**

The class itself has a link under ###main

- **Var Format:** array(package =>  
    array(subpackage =>  
        array(path =>  
            array(class =>  
                array([abstractLink](#) descendant 1, ...  
            )  
            )  
        )  
    )  
)
- See [Converter::formatLeftIndex\(\)](#)

### Converter::\$class\_data

*Smarty = [line 165]*

**template for the class currently being processed**

**Converter::\$class\_elements**

*array = array() [line [208](#)]*

**alphabetized index of classes by package**

- Var Format: array(package => array(subpackage => array([classLink 1](#),[classLink 2](#),...))
- See [Converter::\\$leftindex](#)

**Converter::\$curfile**

*mixed = [line [313](#)]*

**full path of the current file being converted**

**Converter::\$curpage**

*parserPage = [line [171](#)]*

**current procedural page being processed**

**Converter::\$define\_elements**

*array = array() [line [201](#)]*

**alphabetized index of defines by package**

- Var Format: array(package => array(subpackage => array([defineLink 1](#),[defineLink 2](#),...)))
- See [Converter::\\$leftindex](#)

**Converter::\$elements**

*array = array() [line [187](#)]*

**alphabetical index of all elements**

- **Var** Format: array(first letter of element name => array( [parserElement](#) or [parserPage](#),...))
- **See** [Converter::formatIndex\(\)](#), [HTMLframesConverter::formatIndex\(\)](#)

### **Converter::\$function\_elements**

*array = array() [line [222](#)]*

#### **alphabetized index of functions by package**

- **Var** Format: array(package => array(subpackage => array( [functionLink](#) 1,[functionLink](#) 2,...))
- **See** [Converter::\\$leftindex](#)

### **Converter::\$global\_elements**

*array = array() [line [215](#)]*

#### **alphabetized index of global variables by package**

- **Var** Format: array(package => array(subpackage => array( [globalLink](#) 1,[globalLink](#) 2,...)))
- **See** [Converter::\\$leftindex](#)

### **Converter::\$highlightingSource**

*boolean = false [line [325](#)]*

#### **Flag used to help converters determine whether to do special source highlighting**

### **Converter::\$leftindex**

*array = array('classes' => true, 'pages' => true, 'functions' => true, 'defines' => true, 'globals' => true) [line [384](#)]*

#### **Controls which of the one-element-only indexes are generated.**

Generation of these indexes for large packages is time-consuming. This is an optimization feature. An example of how to use this is in [HTMLframesConverter::\\$leftindex](#), and in [HTMLframesConverter::formatLeftIndex\(\)](#). These indexes are intended for use as navigational aids through documentation, but can be used for anything by converters.

- See [Converter::formatLeftIndex\(\)](#)
- See [Converter::\\$class\\_elements](#), [Converter::\\$page\\_elements](#),  
[Converter::\\$function\\_elements](#), [Converter::\\$define\\_elements](#), [Converter::\\$global\\_elements](#)

### **Converter::\$outputformat**

*string = 'Generic' [line 90]*

#### **output format of this converter**

in Child converters, this will match the first part of the -o command-line as in -o HTML:frames:default "HTML"

- [Tutorial phpDocumentor Tutorial](#)

### **Converter::\$package**

*string = 'default' [line 95]*

#### **package name currently being converted**

### **Converter::\$packagecategories**

*array = [line 359]*

#### **Packages associated with categories**

Used by the XML:DocBook/peardoc2 converter, and available to others, to group many packages into categories

- See [phpDocumentor\\_IntermediateParser::\\$packagecategories](#)

### Converter::\$package\_elements

*array = array() [line [180](#)]*

**alphabetical index of all elements sorted by package, subpackage, page, and class.**

- **Var Format:** array(package => array(subpackage => array('page'|'class' => array(path|classname => array(element, element,...)))))
- **Uses [Converter::\\$sort\\_absolutely\\_everything](#)** - if true, then \$package\_elements is used, otherwise, the ParserData::\$classelements and ParserData::\$pageelements variables are used

### Converter::\$package\_output

*mixed = [line [141](#)]*

**set to value of -po commandline**

- **Tutorial [phpDocumentor Tutorial](#)**

### Converter::\$package\_parents

*array = [line [349](#)]*

#### **Hierarchy of packages**

Every package that contains classes may have parent or child classes in other packages. In other words, this code is legal:

1

In this case, package one is a parent of package two

- See [phpDocumentor\\_IntermediateParser::\\$package\\_parents](#)

### Converter::\$page

*string = [line [147](#)]*

**name of current page being converted**

### Converter::\$page\_contents

*array = array() [line [238](#)]*

**alphabetical index of all elements on a page by package/subpackage**

The page itself has a link under ###main

- **Var Format:** array(package => array(subpackage => array(path => array(descendant 1, ...)))) [abstractLink](#)
- See [Converter::formatLeftIndex\(\)](#)

### Converter::\$page\_data

*Smarty = [line [159](#)]*

**template for the procedural page currently being processed**

### Converter::\$page\_elements

*array = array() [line [194](#)]*

**alphabetized index of procedural pages by package**

- **Var Format:** array(package => array(subpackage => array([pageLink 1](#), [pageLink 2](#), ...)))
- See [Converter::\\$leftindex](#)

**Converter::\$parseprivate**

*bool = [line 276]*

**controls processing of elements marked private with @access private**  
defaults to false. Set with command-line --parseprivate or -pp

**Converter::\$path**

*string = [line 153]*

**path of current page being converted**

**Converter::\$pkg\_elements**

*array = array() [line 229]*

**alphabetical index of all elements, indexed by package/subpackage**

- Var Format: array(first letter of element name => array( [parserElement](#) or [parserPage](#),...))
- See [Converter::formatPkgIndex\(\)](#), [HTMLframesConverter::formatPkgIndex\(\)](#)

**Converter::\$processSpecialRoots**

*boolean = false [line 81]*

**This converter knows about the new root tree processing**  
In order to fix PEAR Bug #6389

**Converter::\$quietmode**

*bool = [line 283]*

**controls display of progress information while parsing.**  
defaults to false. Set to true for cron jobs or other situations where no visual output is necessary

**Converter::\$smarty\_dir**

*string = " [line 301]*

## Directory that the smarty templates are in

### Converter::\$sort\_absolutely\_everything

*mixed = false [line 251]*

**This is used if the content must be passed in the order it should be read, i.e. by package, procedural then classes**

This fixes bug 637921, and is used by [PDFdefaultConverter](#)

- Used by [Converter::\\$package\\_elements](#) - if true, then \$package\_elements is used, otherwise, the ParserData::\$classelements and ParserData::\$pageelements variables are used

### Converter::\$sort\_page\_contents\_by\_type

*boolean = false [line 245]*

**This determines whether the [\\$page\\_contents](#) array should be sorted by element type as well as alphabetically by name**

- See [Converter::sortPageContentsByElementType\(\)](#)

### Converter::\$sourcePaths

*array = array() [line 372]*

**A list of files that have had source code generated**

### Converter::\$subpackage

*string = " [line 100]*

**subpackage name currently being converted**

### **Converter::\$targetDir**

*mixed = " [line [289](#)]*

**directory that output is sent to. -t command-line sets this.**

- Tutorial [phpDocumentor Tutorial](#)

### **Converter::\$templateDir**

*string = " [line [295](#)]*

**Directory that the template is in, relative to phpDocumentor root directory**

### **Converter::\$templateName**

*string = " [line [308](#)]*

**Name of the template, from last part of -o**

- Tutorial [phpDocumentor Tutorial](#)

### **Converter::\$template\_options**

*array = [line [399](#)]*

**Options for each template, parsed from the options.ini file in the template base directory**

- Tutorial [phpDocumentor Tutorials](#)

### **Converter::\$title**

*string = 'Generated Documentation' [line [392](#)]*

- See [phpDocumentor\\_IntermediateParser::\\$title](#)

## Converter::\$todoList

*array = array() [line [429](#)]*

### List of all @todo tags and a link to the element with the @todo

Format: array(package => array(link to element, array(todo [parserTag](#),...))),...)

- Tutorial [@todo](#)

Constructor void function Converter::Converter(&\$allp, &\$packp, &\$classes, &\$procpages, \$po, \$pp, \$qm, \$targetDir, \$template, \$title) [line [452](#)]

#### Function Parameters:

- array &\$allp [\\$all\\_packages](#) value
- array &\$packp [\\$package\\_parents](#) value
- [Classes](#) &\$classes [\\$classes](#) value
- [ProceduralPages](#) &\$procpages \$proceduralpages value
- array \$po [\\$package\\_output](#) value
- boolean \$pp [\\$parseprivate](#) value
- boolean \$qm [\\$quietmode](#) value
- string \$targetDir [\\$targetDir](#) value
- string \$template [\\$templateDir](#) value
- string \$title (@link \$title} value

## Initialize Converter data structures

string function Converter::AttrToString(\$tag, \$attr, [\$unmodified = false]) [line [1307](#)]

#### Function Parameters:

- string \$tag tag name
- attribute \$attr array
- boolean \$unmodified if true, returns attrname="value"...

**Convert the attribute of a Tutorial docbook tag's attribute list  
to a string based on the template options.ini**

*string* function Converter::Bolden(\$para) [line [931](#)]

**Function Parameters:**

- *string* \$para

**Used to convert the contents of <b> in a docblock**

- Usedby [parserB::Convert\(\)](#)

*string* function Converter::Br(\$para) [line [1001](#)]

**Function Parameters:**

- *string* \$para

**Used to convert <br> in a docblock**

- Usedby [parserBr::Convert\(\)](#)

*void* function Converter::checkState(\$state) [line [5366](#)]

**Function Parameters:**

- *mixed* \$state

**Compare parserStringWithInlineTags::Convert() cache state to \$state**

- See [parserStringWithInlineTags::Convert\(\)](#)
- Abstract Element

*void* function Converter::cleanup() [[line 5077](#)]

### Finish up parsing/cleanup directories

*void* function Converter::Convert(&\$element) [[line 3982](#)]

#### Function Parameters:

- *mixed* &\$element [parserElement](#) descendant or [parserPackagePage](#) or [parserData](#)

### Convert all elements to output format

This will call ConvertXxx where Xxx is <http://www.php.net/ucfirst>(\$element->type). It is expected that a child converter defines a handler for every element type, even if that handler does nothing. phpDocumentor will terminate with an error if a handler doesn't exist.

- **Throws** PDERROR\_NO\_CONVERT\_HANDLER

*void* function Converter::convertClass(&\$element) [[line 4046](#)]

#### Function Parameters:

- &\$element

### Default Class Handler

#### Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. Sets up the class template. {@, [getFormattedConflicts](#), [getFormattedInheritedMethods](#), and [getFormattedInheritedVars](#) are called to complete vital template setup.}}

*void* function Converter::convertConst(&\$element, [\$additions = array()]) [line [4217](#)]

**Function Parameters:**

- *parserConst* &\$element
- \$additions

**Converts class constants for template output.**

Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template.

This function must be called by a child converter with any extra template variables needed in the parameter \$addition

*void* function Converter::convertDefine(&\$element, [\$addition = array()]) [line [4334](#)]

**Function Parameters:**

- *parserDefine* &\$element
- array \$addition any additional template variables should be in this array

**Converts defines for template output**

Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. This function must be called by a child converter with any extra template variables needed in the parameter \$addition {@, this method also uses utility functions [getGlobalValue\(\)](#), [getFormattedConflicts\(\)](#)}

- **Uses** [Converter::postProcess\(\)](#) - on define\_value template value, makes it displayable

*void* function Converter::ConvertErrorLog() [line [2426](#)]

**Convert the phpDocumentor parsing/conversion error log**

- **Abstract Element**

`void function Converter::convertFunction(&$element, [$addition = array()]) [line 4391]`

**Function Parameters:**

- *parserFunction* &\$element
- \$addition

### Converts function for template output

#### Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. This function must be called by a child converter with any extra template variables needed in the parameter \$addition

- See [Converter::prepareDocBlock\(\)](#)

`void function Converter::convertGlobal(&$element, [$addition = array()]) [line 4300]`

**Function Parameters:**

- *parserGlobal* &\$element
- array \$addition any additional template variables should be in this array

### Converts global variables for template output

#### Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. This function must be called by a child converter with any extra template variables needed in the parameter \$addition {@, this method also uses utility functions [getGlobalValue\(\)](#), [getFormattedConflicts\(\)](#)}

- **Uses** [Converter::postProcess\(\)](#) - on global\_value template value, makes it displayable

*void function Converter::convertInclude(&\$element, [\$addition = array()]) [line [4363](#)]*

**Function Parameters:**

- [parserInclude](#) &\$element
- \$addition

## Converts includes for template output

### Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. This function must be called by a child converter with any extra template variables needed in the parameter \$addition

- See [Converter::prepareDocBlock\(\)](#)

*void function Converter::convertMethod(&\$element, [\$additions = array()]) [line [4113](#)]*

**Function Parameters:**

- [parserMethod](#) &\$element
- \$additions

## Converts method for template output

### Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. This function must be called by a child converter with any extra template variables needed in the parameter \$addition

*void function Converter::convertPage(&\$element) [line [4254](#)]*

**Function Parameters:**

- [parserPage](#) &\$element

## Default Page Handler

Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. {@, this class uses getSourceLocation() and [getClassesOnPage\(\)](#) to set template variables. Also used is getPageName(), to get a Converter-specific name for the page.}}

*string* function Converter::ConvertTitle(\$tag, \$attr, \$title, \$cdata) [line [1396](#)]

**Function Parameters:**

- *string* **\$tag** tag name
- *array* **\$attr**
- *string* **\$title** title text
- *string* **\$cdata**

## Convert the title of a Tutorial docbook tag section

to a string based on the template options.ini

*void* function Converter::ConvertTodoList() [line [2434](#)]

**Convert the list of all @todo tags**

- **Abstract Element**

*void* function Converter::convertTutorial(&\$element) [line [4010](#)]

**Function Parameters:**

- [parserTutorial](#) &\$element

## Default Tutorial Handler

Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template. Sets up

the tutorial template, and its prev/next/parent links {@ and uses the parserTutorial::prev, parserTutorial::next, parserTutorial::parent links to set up those links.})

*void function Converter::convertVar(&\$element, [\$additions = array()]) [line [4173](#)]*

**Function Parameters:**

- *parserVar* **&\$element**
- **\$additions**

### **Converts class variables for template output.**

#### Conversion Handlers

All of the convert\* handlers set up template variables for the Smarty template.

This function must be called by a child converter with any extra template variables needed in the parameter \$addition

*void function Converter::Convert\_RIC(\$name, \$contents) [line [3962](#)]*

**Function Parameters:**

- *README|INSTALL|CHANGELOG* **\$name**
- *string* **\$contents** contents of the file

### **Convert README/INSTALL/CHANGELOG file contents to output format**

- **Abstract Element**

*void function Converter::copyFile(\$file, [\$subdir = "]) [line [5341](#)]*

**Function Parameters:**

- *string* **\$file**
- **\$subdir**

**Copies a file from the template directory to the target directory**  
thanks to Robert Hoffmann for this fix

*void* function Converter::createParentDir(\$dir) [[line 5257](#)]

**Function Parameters:**

- *string* \$dir

**Recursively creates all subdirectories that don't exist in the \$dir path**

*string* function Converter::EncloseList(\$list, \$ordered) [[line 888](#)]

**Function Parameters:**

- *string* \$list
- *\$ordered*

**Used to convert the contents of <ol> or <ul> in a docblock**

- **Used by** [parserList::Convert\(\)](#) - enclose the list

*string* function Converter::EncloseParagraph(\$para) [[line 917](#)]

**Function Parameters:**

- *string* \$para

**Used to enclose a paragraph in a docblock**

*void* function Converter::endClass() [[line 507](#)]

**Called by**[walk\(\)](#) **while converting, when the last class element has been parsed.**

A Converter can use this method in any way it pleases. HTMLframesConverter uses it to complete the template for the class and to output its documentation

- See [HTMLframesConverter::endClass\(\)](#)
- Abstract Element

*void* function Converter::endPage() [*line 521*]

**Called by**[walk\(\)](#) **while converting, when the last procedural page element has been parsed.**

A Converter can use this method in any way it pleases. HTMLframesConverter uses it to complete the template for the procedural page and to output its documentation

- See [HTMLframesConverter::endClass\(\)](#)
- Abstract Element

*string* function Converter::exampleProgramExample(\$example, [\$tutorial = false], [\$inlinesourceparse = null], [\$class = null], [\$linenum = null], [\$filesourcepath = null]) [*line 773*]

**Function Parameters:**

- *string* **\$example**
- *boolean* **\$tutorial** true if this is to highlight a tutorial <programlisting>
- **\$inlinesourceparse**
- **\$class**
- **\$linenum**
- **\$filesourcepath**

**Used to convert the {@example} inline tag in a docblock.**

By default, this just wraps ProgramExample

- See XMLDocBookpearDoc2Converter::exampleProgramExample

*string* function Converter::flushHighlightCache() [*line 609*]

## Return the close text for the current token

*void* function Converter::formatIndex() [line [545](#)]

### Called bywalk() while converting.

This method is intended to be the place that \$elements is formatted for output.

- See [HTMLframesConverter::formatIndex\(\)](#)
- Abstract Element

*void* function Converter::formatLeftIndex() [line [559](#)]

### Called bywalk() while converting.

This method is intended to be the place that any of \$class elements, \$function elements, \$page elements, \$define elements, and \$global elements is formatted for output, depending on the value of \$leftindex

- See [HTMLframesConverter::formatLeftIndex\(\)](#)
- Abstract Element

*void* function Converter::formatPkgIndex() [line [533](#)]

### Called bywalk() while converting.

This method is intended to be the place that \$pkg\_elements is formatted for output.

- See [HTMLframesConverter::formatPkgIndex\(\)](#)
- Abstract Element

*string* function Converter::formatTutorialTOC(\$toc) [line [1030](#)]

#### Function Parameters:

- array **\$toc** format: array(array('tagname' => section, 'link' => returnsee link, 'id' =>

anchor name, 'title' => from title tag),...)

## Creates a table of contents for a {@toc} inline tag in a tutorial

This function should return a formatted table of contents. By default, it does nothing, it is up to the converter to format the TOC

- **Abstract Element**

- **Used by** [parserTocInlineTag::Convert\(\)](#) - passes an array of format:

*void* function Converter::generateChildClassList(\$class) [[line 4794](#)]

**Function Parameters:**

- [parserClass \\$class](#) class variable

**returns a list of child classes**

- **Uses** [parserClass::getChildClassList\(\)](#)

*string* function Converter::generateFormattedClassTree(\$class) [[line 4641](#)]

**Function Parameters:**

- [parserClass \\$class](#) class variable

**returns an array containing the class inheritance tree from the root object to the class.**

This method must be overridden, or phpDocumentor will halt with a fatal error

- **Abstract Element**

array function Converter::getClassesOnPage(&\$element) [*line 4593*]

**Function Parameters:**

- [parserData](#) &\$element

**gets a list of all classes declared on a procedural page represented by**

\$element, a [parserData](#) class

mixed function Converter::getClassLink(\$expr, \$package, [\$file = false], [\$text = false]) [*line 3179*]

**Function Parameters:**

- *string* \$expr class name
- *string* \$package package name
- \$file
- \$text

**return false or [aClassLink](#) to \$expr**

- See [classLink](#)

mixed function Converter::getConstLink(\$expr, \$class, \$package, [\$file = false], [\$text = false]) [*line 3354*]

**Function Parameters:**

- *string* \$expr constant name
- *string* \$class class name
- *string* \$package package name
- \$file
- \$text

**return false or [aConstLink](#) to \$expr in \$class**

- See [constLink](#)

*string* function Converter::getConverterDir() [[line 5162](#)]

**Get the absolute path to the converter's base directory**

*string* function Converter::getCurrentPageLink() [[line 1115](#)]

- **Abstract Element**

*string* function Converter::getCurrentPageURL(\$pathinfo) [[line 1095](#)]

**Function Parameters:**

- *string* **\$pathinfo**

**Return the path to the current**

*mixed* function Converter::getDefineLink(\$expr, \$package, [\$file = false], [\$text = false]) [[line 3227](#)]

**Function Parameters:**

- *string* **\$expr** constant name
- *string* **\$package** package name
- **\$file**
- **\$text**

**return false or [defineLink](#)to \$expr**

- See [defineLink](#)

*string* function Converter::getFileName(\$path, \$pathinfo) [line [1070](#)]

**Function Parameters:**

- *string* **\$pathinfo**
- **\$path**

**Translate the path info into a unique file name for the highlighted source code.**

*string* function Converter::getFilePath(\$base) [line [1083](#)]

**Function Parameters:**

- *string* **\$base** Path is relative to this folder

**Return the fixed path to the source-code file folder.**

*array* function Converter::getFormattedConflicts(&\$element, \$type) [line [4694](#)]

**Function Parameters:**

- *mixed* &**\$element** [parserClass](#), [parserFunction](#), [parserDefine](#) or [parserGlobal](#)
- *string* **\$type** type to display. either 'class','function','define' or 'global variable'

- **Uses** [parserGlobal::getConflicts\(\)](#)
- **Uses** [parserFunction::getConflicts\(\)](#)
- **Uses** [parserDefine::getConflicts\(\)](#)
- **Uses** [parserClass::getConflicts\(\)](#)

*array* function Converter::getFormattedDescMethods(&\$element) [line [4714](#)]

**Function Parameters:**

- [parserMethod](#) &**\$element**

**Get a list of methods in child classes that override this method**

- **Uses [parserMethod::getOverridingMethods\(\)](#)**

array function Converter::getFormattedDescVars(&\$element) [[line 4735](#)]

**Function Parameters:**

- **[parserVar](#) &\$element**

**Get a list of vars in child classes that override this var**

- **Uses [parserVar::getOverridingVars\(\)](#)**

string function Converter::getFormattedImplements(\$el) [[line 4657](#)]

**Function Parameters:**

- **[parserClass](#) \$el** class variable

**returns an array containing the class inheritance tree from the root object to the class.**

This method must be overridden, or phpDocumentor will halt with a fatal error

- **Abstract Element**

array function Converter::getFormattedInheritedConsts(\$child) [[line 4999](#)]

**Function Parameters:**

- **[parserConst](#) \$child** class constant

### **Return template-enabled list of inherited class constants**

uses parserConst helper function getInheritedConsts and generates a template-enabled list using getClassLink()

- See [Converter::getClassLink\(\)](#), parserMethod::getInheritedConsts()

array function Converter::getFormattedInheritedMethods(\$child) [line [4911](#)]

**Function Parameters:**

- [parserMethod](#) \$child class method

### **Return template-enabled list of inherited methods**

uses parserMethod helper function getInheritedMethods and generates a template-enabled list using getClassLink()

- See [Converter::getClassLink\(\)](#), parserMethod::getInheritedMethods()

array function Converter::getFormattedInheritedVars(\$child) [line [4840](#)]

**Function Parameters:**

- [parserVar](#) \$child class var

### **Return template-enabled list of inherited variables**

uses parserVar helper function getInheritedVars and generates a template-enabled list using getClassLink()

- See [Converter::getClassLink\(\)](#), [parserVar::getInheritedVars\(\)](#)

array/false function Converter::getFormattedMethodImplements(&\$element) [line [4773](#)]

**Function Parameters:**

- [parserMethod](#) &\$element

**Get the method this method(s) implemented from an interface, if any**

- Uses [parserMethod::getImplements\(\)](#)

array/false function Converter::getFormattedOverrides(&\$element) [line [4756](#)]

**Function Parameters:**

- [parserMethod](#) &\$element

**Get the method this method overrides, if any**

- See [parserMethod::getOverrides\(\)](#)

mixed function Converter::getFunctionLink(\$expr, \$package, [\$file = false], [\$text = false]) [line [3203](#)]

**Function Parameters:**

- **\$expr** function name
- **\$package** package name
- **\$file**
- **\$text**

**return false or [functionLink](#) to \$expr**

- See [functionLink](#)

*mixed* function Converter::getGlobalLink(\$expr, \$package, [\$file = false], [\$text = false]) [line [3251](#)]

**Function Parameters:**

- *string* **\$expr** global variable name (with leading \$)
- *string* **\$package** package name
- **\$file**
- **\$text**

**return false or [globalLink](#) to \$expr**

- See [defineLink](#)

*string* function Converter::getGlobalValue(\$value) [line [5186](#)]

**Function Parameters:**

- *string* **\$value** default value of a global variable.

### Parse a global variable's default value for class initialization.

If a global variable's default value is "new class" as in:

<sup>1</sup> `$globalvar = new Parser`

This method will document it not as "new Parser" but instead as "new [Parser](#)". For examples, see [phpdoc.inc](#). Many global variables are classes, and phpDocumentor links to their documentation

*void* function Converter::getHighlightState() [line [588](#)]

*string* function Converter::getId(&\$link) [line [3952](#)]

**Function Parameters:**

- [abstractLink](#) &\$link

**take `abstractLink` descendant and text \$eltext and return a unique ID in the format needed for the Converter**

- **Abstract Element**

*string* function Converter::getIncludeValue(\$value, \$ipath) [[line 5227](#)]

**Function Parameters:**

- *string* **\$value** file included by include statement.
- *string* **\$ipath** path of file that has the include statement

### **Parse an include's file to see if it is a file documented in this project**

Although not very smart yet, this method will try to look for the included file file.ext:

```
1     include ("file.ext" );
```

If it finds it, it will return a link to the file's documentation. As of 1.2.0rc1, phpDocumentor is smarty enough to find these cases:

- absolute path to file
- ./file.ext or ../file.ext
- relpath/to/file.ext if relpath is a subdirectory of the base parse directory

For examples, see [Setup.inc.php](#) includes. Every include auto-links to the documentation for the file that is included

*mixed* function Converter::getLink(\$expr, [\$package = false], [\$packages = false]) [[line 3565](#)]

**Function Parameters:**

- *string* **\$expr** expression to search for a link
- *string* **\$package** package to start searching in
- *array* **\$packages** list of all packages to search in

### **The meat of the @see tag and inline {@link} tag**

\$expr is a string with many allowable formats:

1. proceduralpagename.ext
2. constant\_name
3. classname::function()
4. classname::constantname
5. classname::\$variablename
6. classname
7. object classname
8. function functionname()
9. global \$globalvarname
10. packagename#expr where expr is any of the above

New in version 1.1, you can explicitly specify a package to link to that is different from the current package. Use the # operator to specify a new package, as in tests#bug-540368.php (which should appear as a link like: "[tests#bug-540368.php](#)"). This example links to the procedural page bug-540368.php in package tests. Also, the "function" operator is now used to specifically link to a function instead of a method in the current class.

```

1   class myclass
2   {
3       // from inside the class definition, use "function conflict()" to refer to
procedural function "conflict()"
4       function conflict()
5       {
6       }
7   }
8
9   function conflict()
10  {
11  }
```

If classname:: is not present, and the see tag is in a documentation block within a class, then the function uses the classname to search for \$expr as a function or variable within classname, or any of its parent classes. given an \$expr without '\$', '::' or ')' getLink first searches for classes, procedural pages, constants, global variables, and then searches for methods and variables within the default class, and finally for any function

- See [pageLink](#), [functionLink](#), [defineLink](#), [classLink](#), [methodLink](#), [varLink](#)
- See [Converter::getPageLink\(\)](#), [Converter::getDefineLink\(\)](#), [Converter::getVarLink\(\)](#), [Converter::getFunctionLink\(\)](#), [Converter::getClassLink\(\)](#)

*mixed* function Converter::getMethodLink(\$expr, \$class, \$package, [\$file = false], [\$text = false]) [[line 3300](#)]  
**Function Parameters:**

- *string* \$expr method name

- *string* **\$class** class name
- *string* **\$package** package name
- **\$file**
- **\$text**

return false or [methodLink](#) to \$expr in \$class

- See [methodLink](#)

*mixed* function Converter::getPageLink(\$expr, \$package, [\$path = false], [\$text = false], [\$packages = false])  
[line [3275](#)]

**Function Parameters:**

- *string* **\$expr** procedural page name
- *string* **\$package** package name
- **\$path**
- **\$text**
- **\$packages**

return false or [pageLink](#) to \$expr

- See [pageLink](#)

*array* function Converter::getSortedClassTreeFromClass(\$class, \$package, \$subpackage) [line [2954](#)]

**Function Parameters:**

- *string* **\$class** class name
- *string* **\$package**
- *string* **\$subpackage**

### Return a tree of all classes that extend this class

The data structure returned is designed for a non-recursive algorithm, and is somewhat complex. In most cases, the array returned is:

```

array('#root' =>
    array('link' => classLink to $class,
        'parent' => false,
        'children' => array(array('class' => 'childclass1',
            'package' => 'child1package'),
            array('class' => 'childclass2',
                'package' => 'child2package'),...
        )
    ),
    'child1package#childclass1' =>
        array('link' => classLink to childclass1,
            'parent' => '#root',
            'children' => array(array('class' => 'kidclass',
                'package' => 'kidpackage'),...
            )
        ),
    'kidpackage#kidclass' =>
        array('link' => classLink to kidclass,
            'parent' => 'child1package#childclass1',
            'children' => array() // no children
        ),
    ...
)
)

```

To describe this format using language, every class in the tree has an entry in the first level of the array. The index for all child classes that extend the root class is childpackage#childclassname. Each entry in the array has 3 elements: link, parent, and children.

- link - a [classLink](#) to the current class
- parent - a [classLink](#) to the class's parent, or false (except for one special case described below)
- children - an array of arrays, each entry has a 'class' and 'package' index to the child class,  
used to find the entry in the big array

special cases are when the #root class has a parent in another package, or when the #root class extends a class not found by phpDocumentor. In the first case, parent will be a classLink to the parent class. In the second, parent will be the extends clause, as in:

```

1   class X extends Y
2   {
3   }
4 }
```

in this case, the #root entry will be array('link' => classLink to X, 'parent' => 'Y', children => array(...))

The fastest way to design a method to process the array returned is to copy `HTMLframesConverter::getRootTree()` into your converter and to modify the html to whatever output format you are going to use

- See [HTMLframesConverter::getRootTree\(\)](#)

*string* function Converter::getSourceLink(\$path) [/line [1105](#)]

**Function Parameters:**

- **\$path**

- **Abstract Element**
- **Used by** [parserFileSourceTag::getSourceLink\(\)](#)

*void* function Converter::getState() [/line [5355](#)]

**Return parserStringWithInlineTags::Convert() cache state**

- See [parserStringWithInlineTags::Convert\(\)](#)
- **Abstract Element**

*string* function Converter::getTutorialId(\$package, \$subpackage, \$tutorial, \$id) [/line [1418](#)]

**Function Parameters:**

- **\$package**
- **\$subpackage**
- **\$tutorial**
- **\$id**

**Return a converter-specific id to distinguish tutorials and their sections**

Used by {@id}

- **Usedby** [parserTocInLineTag::Convert\(\)](#) - retrieve the tutorial ID for
- **Usedby** [parserIdInLineTag::Convert\(\)](#) - retrieve converter-specific ID

*tutorialLink|string* function Converter::getTutorialLink(\$expr, [\$package = false], [\$subpackage = false], [\$packages = false]) [line [3417](#)]

**Function Parameters:**

- *string \$expr* the original expression
- *string \$package* package to look in first
- *string \$subpackage* subpackage to look in first
- *array \$packages* array of package names to search in if not found in parent packages.  
This is used to limit the search, phpDocumentor automatically searches all packages

### The meat of the @tutorial tag and inline {@tutorial} tag

Take a string and return an abstract link to the tutorial it represents. Since tutorial naming literally works like the underlying filesystem, the way to reference the tutorial is similar. Tutorials are located in a subdirectory of any directory parsed, which is named 'tutorials/' (we try to make things simple when we can). They are further organized by package and subpackage as:

tutorials/package/subpackage

and the files are named \*.cls, \*.pkg, or \*.proc, and so a link to a tutorial named file.cls can be referenced (depending on context) as any of:

```

1      * @tutorial package/subpackage/file.cls
2      * @tutorial package/file.cls
3      * @tutorial file.cls

```

The first case will only be needed if file.cls exists in both the current package, in another package/file.cls and in another package/subpackage/file.cls and you wish to reference the one in another package/subpackage. The second case is only needed if you wish to reference file.cls in another package and it is unique in that package. the third will link to the first file.cls it finds using this search method:

1. current package/subpackage
2. all other subpackages of current package
3. parent package, if this package has classes that extend classes in another package
4. all other packages

- Since 1.2

array function Converter::getTutorialTree(\$tutorial) [line [1974](#)]

**Function Parameters:**

- [parserTutorial](#)|array \$tutorial

### Get a tree structure representing the hierarchy of tutorials

Returns an array in format: array('tutorial' => [parserTutorial](#),  
'kids' => array( // child tutorials  
    array('tutorial' => child [parserTutorial](#),  
        'kids' => array(...)  
    )  
)  
)

- Tutorial [phpDocumentor Tutorials](#)

mixed function Converter::getVarLink(\$expr, \$class, \$package, [\$file = false], [\$text = false]) [line [3327](#)]

**Function Parameters:**

- string \$expr var name
- string \$class class name
- string \$package package name
- \$file
- \$text

return false or [varLink](#) to \$expr in \$class

- See [varLink](#)

*boolean* function Converter::hasSourceCode(\$path) [line [1159](#)]

**Function Parameters:**

- *string* **\$path** full path to the source code file

**Determine whether an element's file has generated source code, used for linking to line numbers of source.**

Wrapper for [\\$sourcePaths](#) in this version

*false|parserTutorial* function Converter::hasTutorial(\$type, \$name, \$package, [\$subpackage = "]) [line [490](#)]

**Function Parameters:**

- *pkg|cls|proc* **\$type** the tutorial type to search for
- *tutorial* **\$name** name
- *string* **\$package** package name
- *string* **\$subpackage** subpackage name, if any

*string* function Converter::highlightDocBlockSource(\$token, \$word, [\$preformatted = false]) [line [690](#)]

**Function Parameters:**

- *string* **\$token** name of docblock token type
- *string* **\$word** contents of token
- *boolean* **\$preformatted** whether the contents are preformatted or need modification

**Used to allow converters to format the source code of DocBlocks the way they'd like.**

default returns it unchanged. Mainly used by the HighlightParser

*string* function Converter::highlightSource(\$token, \$word, [\$preformatted = false]) [line [637](#)]

**Function Parameters:**

- *integer* **\$token** token value from [tokenizer constants](#)
- *string* **\$word** contents of token
- *boolean* **\$preformatted** whether the contents are preformatted or need modification

**Used to allow converters to format the source code the way they'd like.**  
default returns it unchanged. Mainly used by the HighlightParser

*string* function Converter::highlightTutorialSource(\$token, \$word, [\$preformatted = false]) [line [728](#)]

**Function Parameters:**

- *string* **\$token** name of docblock token type
- *string* **\$word** contents of token
- *boolean* **\$preformatted** whether the contents are preformatted or need modification

**Used to allow converters to format the source code of Tutorial XML the way they'd like.**

default returns it unchanged. Mainly used by the HighlightParser

*string* function Converter::Italicize(\$para) [line [945](#)]

**Function Parameters:**

- *string* **\$para**

**Used to convert the contents of <i> in a docblock**

- Used by [parser::Convert\(\)](#)

*string* function Converter::Kbdize(\$para) [line [973](#)]

**Function Parameters:**

- *string* **\$para**

**Used to convert the contents of <kbd> in a docblock**

- **Usedby** [parserKbd::Convert\(\)](#)

*string* function Converter::ListItem(\$item) [[line 874](#)]

**Function Parameters:**

- *string* **\$item**

**Used to convert the contents of <li> in a docblock**

- **Usedby** [parserList::Convert\(\)](#) - enclose each item of the list

*Smarty* function Converter::newSmarty() [[line 5051](#)]

**Return a Smarty template object to operate with**

This returns a Smarty template with pre-initialized variables for use. If the method "SmartyInit()" exists, it is called.

*void* function Converter::Output(\$title) [[line 5112](#)]

**Function Parameters:**

- **\$title**

**do all necessary output**

- See [Converter](#)
- [Abstract Element](#)
- **Usedby** [phpDocumentor\\_IntermediateParser::Convert\(\)](#)

*string* function Converter::postProcess(\$text) [line [1016](#)]

**Function Parameters:**

- **\$text**

### This version does nothing

Perform necessary post-processing of string data. For example, the HTML Converters should escape < and > to become &lt; and &gt;

- **Usedby** [Converter::convertDefine\(\)](#) - on define\_value template value, makes it displayable
- **Usedby** [Converter::convertGlobal\(\)](#) - on global\_value template value, makes it displayable

*array* function Converter::prepareDocBlock(&\$element, [\$names = array()], [\$nopackage = true]) [line [4456](#)]

**Function Parameters:**

- *mixed* **&\$element** any descendant of [parserElement](#), or [parserData](#)
- *array* **\$names** used to translate tagnames into other tags
- *boolean* **\$nopackage** set to false for pages and classes, the only elements allowed to specify @package

### convert the element's DocBlock for output

This function converts all tags and descriptions for output

*string* function Converter::PreserveWhiteSpace(\$string) [line [903](#)]

**Function Parameters:**

- *string* **\$string**

### Used to convert the contents of <pre> in a docblock

- **Usedby** [parserPre::Convert\(\)](#)

*string* function Converter::ProgramExample(\$example, [\$tutorial = false], [\$inlinesourceparse = null], [\$class = null], [\$linenum = null], [\$filesourcepath = null]) [line [785](#)]

**Function Parameters:**

- *string* **\$example**
- *boolean* **\$tutorial** true if this is to highlight a tutorial <programlisting>
- **\$inlinesourceparse**
- **\$class**
- **\$linenum**
- **\$filesourcepath**

**Used to convert the <code> tag in a docblock**

- Used by [parserCode::Convert\(\)](#)

*string* function Converter::returnLink(\$link, \$text) [line [3929](#)]

**Function Parameters:**

- *string* **\$link** URL
- *string* **\$text** text to display

**take URL \$link and text \$text and return a link in the format needed for the Converter**

- Abstract Element

*string* function Converter::returnSee(&\$link, [\$eltext = false]) [line [3941](#)]

**Function Parameters:**

- [abstractLink](#) &\$link

- *string \$eltext*

**take [abstractLink](#)descendant and text \$eltext and return a link  
in the format needed for the Converter**

- **Abstract Element**

*string* function Converter::Sampize(\$para) [[line 987](#)]

**Function Parameters:**

- *string \$para*

**Used to convert the contents of <samp> in a docblock**

- **Usedby** [parserSamp::Convert\(\)](#)

*void* function Converter::setSourcePaths(\$path) [[line 1174](#)]

**Function Parameters:**

- *string \$path* full path of source file

**Mark a file as having had source code highlighted**

*void* function Converter::setTargetDir(\$dir) [[line 5285](#)]

**Function Parameters:**

- *string \$dir* the output directory

## Sets the output directory for generated documentation

As of 1.3.0RC6, this also sets the compiled templates directory inside the target directory

`void function Converter::setTemplateBase($base, $dir) [/line 5123]`

### **Function Parameters:**

- *string* **\$base** template base directory
- *string* **\$dir** template name

## Set the template directory with a different template base directory

- **Tutorial** [phpDocumentor Tutorial](#)

`void function Converter::setTemplateDir($dir) [/line 5148]`

### **Function Parameters:**

- *string* **\$dir** subdirectory

## sets the template directory based on **\$outputformat** and **\$name**

Also sets [\\$templateName](#) to the \$dir parameter

`void function Converter::sortPageContentsByElementType(&$pages) [/line 2570]`

### **Function Parameters:**

- **&\$pages**

## sorts [\\$page\\_contents](#) by element type as well as alphabetically

- See [\\$sort\\_page\\_contents\\_by\\_element\\_type](#)

*string* function Converter::sourceLine(\$linenumber, \$line, [\$path = false]) [line [1130](#)]

**Function Parameters:**

- *integer* **\$linenumber** line number
- *string* **\$line** highlighted source code line
- *false|string* **\$path** full path to @filesource file this line is a part of, if this is a single line from a complete file.

### Return a line of highlighted source code with formatted line number

If the \$path is a full path, then an anchor to the line number will be added as well

*void* function Converter::startHighlight() [line [582](#)]

### Initialize highlight caching

*void* function Converter::TranslateEntity(\$name) [line [1184](#)]

**Function Parameters:**

- *string* **\$name** entity name

Used to translate an XML DocBook entity like &rdquo; from a tutorial by reading the options.ini file for the template.

- Used by [parserEntity::Convert\(\)](#) - convert contents to text

*string* function Converter::TranslateTag(\$name, \$attr, \$cdata, \$unconvertedcdata) [line [1214](#)]

**Function Parameters:**

- *string* **\$name** tag name
- *string* **\$attr** any attributes Format: array(name => value)
- *string* **\$cdata** the tag contents, if any
- *string* **\$unconvertedcdata** the tag contents, if any, unpost-processed

**Used to translate an XML DocBook tag from a tutorial by reading the options.ini file for the template.**

- **Used by** [parserXMLDocBookTag::Convert\(\)](#) - Calls this to enclose the contents of the DocBook tag based on the values in template options.ini file

*void* function Converter::TutorialExample(\$example) [[line 860](#)]

**Function Parameters:**

- *string* \$example

*string* function Converter::type\_adjust(\$typename) [[line 759](#)]

**Function Parameters:**

- *string* \$typename

**Called by** [parserReturnTag::Convert\(\)](#) to allow converters to change type names to desired formatting

Used by [XMLDocBookConverter::type\\_adjust\(\)](#) to change true and false to the peardoc2 values

*string* function Converter::unmangle(\$sourcecode) [[line 574](#)]

**Function Parameters:**

- *string* \$sourcecode output from highlight\_string() - use this function to reformat the returned data for Converter-specific output

**Called by** [parserSourceInlineTag::stringConvert\(\)](#) to allow converters to format the source code the way they'd like.

default returns it unchanged (html with xhtml tags)

- **Deprecated** in favor of tokenizer-based highlighting. This will be removed for 2.0
- **Usedby** [parserSourceInlineTag::stringConvert\(\)](#) - remove the extraneous stuff from [http://www.php.net/highlight\\_string](http://www.php.net/highlight_string)

*string* function Converter::vardump\_tree(\$tree, [\$indent = "]) [line [2202](#)]

**Function Parameters:**

- **\$tree**
- **\$indent**

### Debugging function for dumping \$tutorial\_tree

*string* function Converter::Varize(\$para) [line [959](#)]

**Function Parameters:**

- *string* **\$para**

### Used to convert the contents of <var> in a docblock

- **Usedby** [parserDescVar::Convert\(\)](#)

*void* function Converter::walk(&\$pages, &\$package\_pages) [line [1705](#)]

**Function Parameters:**

- **array &\$pages** Format: array(fullpath => [parserData](#) structure with full [parserData::\\$elements](#) and parserData::\$class\_elements.
- **array &\$package\_pages** Format: array([parserPackagePage](#) 1, [parserPackagePage](#) 2,...)

**called by** [phpDocumentor\\_IntermediateParser::Convert\(\)](#) **traverse the array of pages and their elements, converting them to the output format**

The walk() method should be flexible enough such that it never needs modification. walk() sets up all of the indexes, and sorts everything in logical alphabetical order. It then passes each element individually to [Convert\(\)](#), which then passes to the Convert\*()

methods. A child Converter need not override any of these unless special functionality must be added. see [Converter Default Template Variables](#) for details. {@ and the left indexes specified by \$leftIndexes, and then sorts them by calling sortIndexes().

Next, it converts all README/CHANGELOG/INSTALL-style files, using [Convert\\_RIC](#).

After this, it passes all package-level docs to Convert(). Then, it calls the index sorting functions [formatPkgIndex\(\)](#), [formatIndex\(\)](#) and [formatLeftIndex\(\)](#).

Finally, it converts each procedural page in alphabetical order. This stage passes elements from the physical file to Convert() in alphabetical order. First, procedural page elements [parserDefine](#), [parserInclude](#) [parserGlobal](#), and [parserFunction](#) are passed to Convert().

Then, class elements are passed in this order: [parserClass](#), then all of the [parserVars](#) in the class and all of the [parserMethods](#) in the class. Classes are in alphabetical order, and both vars and methods are in alphabetical order.

Finally, [ConvertErrorLog\(\)](#) is called and the data walk is complete.}}

- **Used by** [phpDocumentor\\_IntermediateParser::Convert\(\)](#) - passes \$pages and \$package\_pages
- **Uses** Converter::\_createPkgElements() - sets up [\\$elements](#) and [\\$pkg\\_elements](#) array, as well as \$links

`void function Converter::walk_everything() [line 2267]`

### **walk over elements by package rather than page**

This method is designed for converters like the PDF converter that need everything passed in alphabetical order by package/subpackage and by procedural and then class information

- See [Converter::walk\(\)](#)
- See [PDFdefaultConverter](#)

`void function Converter::writeExample($title, $path, $source) [line 1060]`

#### **Function Parameters:**

- *string \$title* example title
- *string \$path* example filename (no path)
- *string \$source* fully highlighted/linked source code of the file

## Write out the formatted source code for an example php file

This function provides the primary functionality for the [@example](#) tag.

- **Abstract Element**

*void* function Converter::writeFile(*\$file*, *\$data*, [*\$binary* = false]) *[line 5318]*

**Function Parameters:**

- *string \$file*
- *string \$data*
- *boolean \$binary* true if the data is binary and not text

## Writes a file to target dir

*void* function Converter::writeSource(*\$filepath*, *\$source*) *[line 1044]*

**Function Parameters:**

- *string \$filepath* full path to the file
- *string \$source* fully highlighted/linked source code of the file

## Write out the formatted source code for a php file

This function provides the primary functionality for the [@filesource](#) tag.

- **Abstract Element**
- **Usedby** [parserFileSourceTag::writeSource\(\)](#) - export highlighted file source

*void* function Converter::\_rmdir(\$directory) [line [5089](#)]

**Function Parameters:**

- *string* \$directory

## Completely remove a directory and its contents

*void* function Converter::\_setHighlightCache(\$type, \$token) [line [593](#)]

**Function Parameters:**

- \$type
- \$token

*string* function Converter::\_tutorial\_path(\$pkg, [\$subpkg = 0], [\$namepkg = 0]) [line [2081](#)]

**Function Parameters:**

- [\*parserTutorial\*](#) \$pkg
- [\*parserTutorial\*](#) \$subpkg
- [\*parserTutorial\*](#) \$namepkg

## Returns the path to this tutorial as a string

# CHMdefaultConverter.inc

**CHM (Compiled Help Manual) output converter for Smarty Template.**

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** CHMdefault
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: CHMdefaultConverter.inc 234145 2007-04-19 20:20:57Z ashnazg \$
- **Copyright** 2000-2006 Joshua Eichorn, Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.0rc1
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

## Class CHMdefaultConverter

[line [56](#)]

## **Generates files that MS HTML Help Workshop can use to create a MS Windows compiled help file (CHM)**

The free MS HTML Help compiler takes the project file (phpdoc.hhp) and reads the table of contents file specified in the project (which is always contents.hhc in phpDocumentor). When the converter reaches stable state, it will also output an index file index.hhk. The free download for MS HTML Help Workshop is available below

- **Package** Converters
- **Sub-Package** CHMdefault
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Revision: 234145 \$
- **Link** [MS HTML Help Workshop download](#)
- **Since** 1.0rc1

### **CHMdefaultConverter::\$base\_dir**

*string = [line [85](#)]*

**target directory passed on the command-line.**

[\\$targetDir](#) is malleable, always adding package/ and package/subpackage/ subdirectories onto it.

### **CHMdefaultConverter::\$class\_dir**

*string = [line [91](#)]*

**output directory for the current class being processed**

### **CHMdefaultConverter::\$current**

*array = [line [116](#)]*

**contains all of the template procedural page element loop data needed for the current template**

### **CHMdefaultConverter::\$currentclass**

*array = [line [122](#)]*

**contains all of the template class element loop data needed for the current template**

## **CHMdefaultConverter::\$juststarted**

*boolean = false [line [110](#)]*

### **controls formatting of parser informative output**

Converter prints: "Converting /path/to/file.php... Procedural Page Elements... Classes..." Since CHMdefaultConverter outputs files while converting, it needs to send a \n to start a new line. However, if there is more than one class, output is messy, with multiple \n's just between class file output. This variable prevents that and is purely cosmetic

## **CHMdefaultConverter::\$KLinks**

*array = array() [line [129](#)]*

### **Table of Contents entry for index.hhk**

- Used by [CHMdefaultConverter::addKLink\(\)](#) - tracks the index

## **CHMdefaultConverter::\$leftindex**

*array = array('classes' => true, 'pages' => true, 'functions' => true, 'defines' => false, 'globals' => false) [line [72](#)]*

### **indexes of elements by package that need to be generated**

## **CHMdefaultConverter::\$name**

*string = 'default' [line [67](#)]*

## **CHMdefaultConverter::\$outputformat**

*string = 'CHM' [line [65](#)]*

## **CHMdefaultConverter::\$package\_pages**

*array = array() [line [98](#)]*

### **array of converted package page names.**

Used to link to the package page in the left index

- **Var** Format: array(package => 1)

### **CHMdefaultConverter::\$page\_dir**

*string = [line [78](#)]*

**output directory for the current procedural page being processed**

### **CHMdefaultConverter::\$ric\_set**

*mixed = array() [line [124](#)]*

### **CHMdefaultConverter::\$sort\_page\_contents\_by\_type**

*boolean = true [line [63](#)]*

**CHMdefaultConverter wants elements sorted by type as well as alphabetically**

- See [Converter::\\$sort\\_page\\_contents\\_by\\_type](#)

### **CHMdefaultConverter::\$wrote**

*mixed = false [line [123](#)]*

Constructor void function CHMdefaultConverter::CHMdefaultConverter(&\$allp, &\$packp, &\$classes, &\$procpages, \$po, \$pp, \$qm, \$targetDir, \$templateDir, \$title) [line [135](#)]

#### **Function Parameters:**

- **&\$allp**
- **&\$packp**
- **&\$classes**
- **&\$procpages**
- **\$po**
- **\$pp**
- **\$qm**
- **\$targetDir**
- **\$templateDir**
- **\$title**

**sets [\\$base\\_dir](#) to [\\$targetDir](#)**

- See [Converter::Converter\(\)](#)

*void* function CHMdefaultConverter::addHHP(\$file) [*line 1700*]

**Function Parameters:**

- **\$file**
  - **Uses \$hhp\_files** - creates the array by adding parameter \$file

*void* function CHMdefaultConverter::addKLink(\$name, \$file, [\$bookmark = "], [\$group = "]) [*line 1794*]

**Function Parameters:**

- **string \$name** index entry name
- **string \$file** filename containing index
- **string \$bookmark** html anchor of location in file, if any
- **string \$group** group this entry with a string

## Add an item to the index.hhk file

- **Author** Andrew Eddie < [eddieajau@users.sourceforge.net](mailto:eddieajau@users.sourceforge.net)>
- **Uses [CHMdefaultConverter::\\$KLinks](#)** - tracks the index

*void* function CHMdefaultConverter::addSourceTOC(\$name, \$file, \$package, \$subpackage, [\$source = false]) [*line 1761*]

**Function Parameters:**

- **\$name**
- **\$file**
- **\$package**

- **\$subpackage**
- **\$source**

*void* function CHMdefaultConverter::addTOC(\$name, \$file, \$package, \$subpackage, [\$class = false], [\$tutorial = false]) [line [1772](#)]

**Function Parameters:**

- **\$name**
- **\$file**
- **\$package**
- **\$subpackage**
- **\$class**
- **\$tutorial**

*void* function CHMdefaultConverter::convertClass(&\$element) [line [828](#)]

**Function Parameters:**

- [\*parserClass\*](#) &\$element

## Converts class for template output

- See [Converter::getFormattedInheritedMethods\(\)](#), [Converter::getFormattedInheritedVars\(\)](#)
- See [Converter::prepareDocBlock\(\)](#), [Converter::generateChildClassList\(\)](#),  
[CHMdefaultConverter::generateFormattedClassTree\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function CHMdefaultConverter::convertConst(&\$element) [line [860](#)]

**Function Parameters:**

- [\*parserDefine\*](#) &\$element

## Converts class constants for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function CHMdefaultConverter::convertDefine(&\$element) [/line [909](#)]

**Function Parameters:**

- *parserDefine* &\$element

## Converts defines for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function CHMdefaultConverter::ConvertErrorLog() [/line [698](#)]

## Create errors.html template file output

This method takes all parsing errors and warnings and spits them out ordered by file and line number.

- **Global Variable Used** [ErrorTracker](#) \$phpDocumentor\_errors: We'll be using it's output facility

*void* function CHMdefaultConverter::convertFunction(&\$element) [/line [884](#)]

**Function Parameters:**

- *parserFunction* &\$element

## Converts function for template output

- See [Converter::prepareDocBlock\(\)](#), [parserFunction::getFunctionCall\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function CHMdefaultConverter::convertGlobal(&\$element) [line [920](#)]

**Function Parameters:**

- [\*parserGlobal\*](#) &\$element

## Converts global variables for template output

*void* function CHMdefaultConverter::convertInclude(&\$element) [line [897](#)]

**Function Parameters:**

- [\*parserInclude\*](#) &\$element

## Converts include elements for template output

- See [\*Converter::prepareDocBlock\(\)\*](#)

*void* function CHMdefaultConverter::convertMethod(&\$element) [line [872](#)]

**Function Parameters:**

- [\*parserDefine\*](#) &\$element

## Converts class methods for template output

- See [\*Converter::prepareDocBlock\(\)\*](#), [\*Converter::getFormattedConflicts\(\)\*](#)

*void* function CHMdefaultConverter::convertPackagepage(&\$element) [line [773](#)]

**Function Parameters:**

- [parserPackagePage](#) &\$element

**Converts package page and sets its package as used** [\\$package\\_pages](#)

*void* function CHMdefaultConverter::convertPage(&\$element) [*line 932*]

**Function Parameters:**

- [parserData](#) &\$element

**converts procedural pages for template output**

- See [Converter::prepareDocBlock\(\)](#), [Converter::getClassesOnPage\(\)](#)

*void* function CHMdefaultConverter::ConvertTodoList() [*line 644*]

*void* function CHMdefaultConverter::convertTutorial(&\$element) [*line 794*]

**Function Parameters:**

- [parserTutorial](#) &\$element

*void* function CHMdefaultConverter::convertVar(&\$element) [*line 848*]

**Function Parameters:**

- [parserDefine](#) &\$element

**Converts class variables for template output**

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function CHMdefaultConverter::Convert\_RIC(\$name, \$contents) [line [680](#)]

**Function Parameters:**

- **README|INSTALL|CHANGELOG \$name**
- **string \$contents** contents of the file

## Convert README/INSTALL/CHANGELOG file contents to output format

*void* function CHMdefaultConverter::copyMediaRecursively(\$media, \$targetdir, [\$subdir = "]) [line [1360](#)]

**Function Parameters:**

- **\$media**
- **\$targetdir**
- **\$subdir**

*void* function CHMdefaultConverter::endClass() [line [340](#)]

## Writes out the template file [\\$class](#) data and unsets the template to save memory

- See [Converter::endClass\(\)](#)
- See [registerCurrentClass\(\)](#)

*void* function CHMdefaultConverter::endPage() [line [363](#)]

## Writes out the template file [\\$page](#) data and unsets the template to save memory

- See [Converter::endPage\(\)](#)
- See [registerCurrent\(\)](#)

*void* function CHMdefaultConverter::formatIndex() [line [427](#)]

## CHMdefaultConverter uses this function to format template index.html and packages.html

This function generates the package list from [\\$all\\_packages](#), eliminating any packages that don't have any entries in their package index (no files at all, due to @ignore or other factors). Then it uses the default package name as the first package index to

display. It sets the right pane to be either a blank file with instructions on making package-level docs, or the package-level docs for the default package.

- **Global Variable Used** string \$phpDocumentor\_DefaultPackageName: Used to set the starting package to display

void function CHMdefaultConverter::formatLeftIndex() [[line 523](#)]

### Generate indexes for li\_package.html and classtree output files

This function generates the li\_package.html files from the template file left.html. It does this by iterating through each of the \$page\_elements, \$class\_elements and \$function\_elements arrays to retrieve the pre-sorted [abstractLink](#) descendants needed for index generation. Conversion of these links to text is done by [returnSee\(\)](#). The \$local parameter is set to false to ensure that paths are correct.

Then it uses [generateFormattedClassTrees\(\)](#) to create class trees from the template file classtrees.html. Output filename is classtrees\_packagename.html. This function also unsets [\\$elements](#) and [\\$pkg\\_elements](#) to free up the considerable memory these two class vars use

- See [Converter::\\$page\\_elements](#), [Converter::\\$class\\_elements](#), [Converter::\\$function\\_elements](#)

void function CHMdefaultConverter::formatPkgIndex() [[line 395](#)]

### CHMdefaultConverter chooses to format both package indexes and the complete index here

This function formats output for the elementindex.html and pkgelementindex.html template files. It then writes them to the target directory

- See [CHMdefaultConverter::generateElementIndex\(\)](#),  
[CHMdefaultConverter::generatePkgElementIndex\(\)](#)

*string* function CHMdefaultConverter::formatTutorialTOC(\$toc) [line [311](#)]

**Function Parameters:**

- **array \$toc** format: array(array('tagname' => section, 'link' => returnsee link, 'id' => anchor name, 'title' => from title tag),...)

**Use the template tutorial\_toc.tpl to generate a table of contents for HTML**

*void* function CHMdefaultConverter::generateElementIndex() [line [1321](#)]

**Generate alphabetical index of all elements**

- See [Converter::\\$elements](#), [Converter::walk\(\)](#)

*array* function CHMdefaultConverter::generateFormattedClassTree(\$class) [line [959](#)]

**Function Parameters:**

- ***parserClass \$class*** class variable

**returns an array containing the class inheritance tree from the root object to the class**

- Uses [parserClass::getParentClassTree\(\)](#)

*void* function CHMdefaultConverter::generateFormattedClassTrees(\$package) [line [1023](#)]

**Function Parameters:**

- ***string \$package*** package to generate a class tree for

**returns a template-enabled array of class trees**

- See \$roots, HTMLConverter::getRootTree()

void function CHMdefaultConverter::generateFormattedInterfaceTrees(\$package) [line [1077](#)]

**Function Parameters:**

- *string* **\$package** package to generate a class tree for

**returns a template-enabled array of interface inheritance trees**

- See \$roots, HTMLConverter::getRootTree()

string function CHMdefaultConverter::generateKLinks() [line [1815](#)]

**Get the table of contents for index.hhk**

- **Author** Andrew Eddie < [eddieajau@users.sourceforge.net](mailto:eddieajau@users.sourceforge.net)>

void function CHMdefaultConverter::generatePkgElementIndex(\$package) [line [1425](#)]

**Function Parameters:**

- *string* **\$package** name of a package

**Generate alphabetical index of all elements by package and subpackage**

- See [Converter::\\$pkg\\_elements](#), [Converter::walk\(\)](#),  
[CHMdefaultConverter::generatePkgElementIndexes\(\)](#)

*void* function CHMdefaultConverter::generatePkgElementIndexes() [*line 1476*]

- See [CHMdefaultConverter::generatePkgElementIndex\(\)](#)

*void* function CHMdefaultConverter::generateTOC() [*line 1709*]

- Usedby [bug698356\\_Output\(\)](#) - assigns to the toc template variable
- Usedby [CHMdefaultConverter::Output\(\)](#) - assigns to the toc template variable

*void* function CHMdefaultConverter::getCDATA(\$value) [*line 759*]

**Function Parameters:**

- **\$value**

*mixed* function CHMdefaultConverter::getClassLink(\$expr, \$package, [\$file = false], [\$text = false], [\$with\_a = true]) [*line 1516*]

**Function Parameters:**

- *string* **\$expr** name of class
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$with\_a** return just the URL, or enclose it in an html a tag

- See [Converter::getClassLink\(\)](#)

*mixed* function CHMdefaultConverter::getConstLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], 5) [*line 1631*]

**Function Parameters:**

- *boolean* 5 return just the URL, or enclose it in an html a tag
- *string* \$expr name of class constant
- *string* \$class class containing class constant
- *string* \$package package name
- *string* \$file full path to look in (used in index generation)
- *boolean* \$text deprecated

- See [Converter::getVarLink\(\)](#)

*void* function CHMdefaultConverter::getCurrentPageLink() [*line 289*]  
*mixed* function CHMdefaultConverter::getDefineLink(\$expr, \$package, [\$file = false], [\$text = false], 4) [*line 1548*]

**Function Parameters:**

- *boolean* 4 return just the URL, or enclose it in an html a tag
- *string* \$expr name of define
- *string* \$package package name
- *string* \$file full path to look in (used in index generation)
- *boolean* \$text deprecated

- See [Converter::getDefineLink\(\)](#)

*void* function CHMdefaultConverter::getExampleLink(\$path, \$title) [*line 199*]  
**Function Parameters:**

- \$path
- \$title

*mixed* function CHMdefaultConverter::getFunctionLink(\$expr, \$package, [\$file = false], [\$text = false], 4) [*line 1532*]

**Function Parameters:**

- *boolean* 4 return just the URL, or enclose it in an html a tag
- *string* \$expr name of function
- *string* \$package package name

- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getFunctionLink\(\)](#)

*mixed* function CHMdefaultConverter::getGlobalLink(\$expr, \$package, [\$file = false], [\$text = false], 4) [[line 1564](#)]

**Function Parameters:**

- *boolean 4* return just the URL, or enclose it in an html a tag
- *string \$expr* name of global variable
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getGlobalLink\(\)](#)

*void* function CHMdefaultConverter::getId(\$element, [\$fullpath = true]) [[line 591](#)]

**Function Parameters:**

- *\$element*
- *\$fullpath*

*array* function CHMdefaultConverter::getIndexInformation(\$elt) [[line 1204](#)]

**Function Parameters:**

- *parserElement \$elt* descendant of parserElement

## Generate indexing information for given element

- See [CHMdefaultConverter::generateElementIndex\(\)](#)

*mixed* function CHMdefaultConverter::getMethodLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], 5) [[line 1597](#)]

**Function Parameters:**

- *boolean* **5** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of method
- *string* **\$class** class containing method
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getMethodLink\(\)](#)

*mixed* function CHMdefaultConverter::getPageLink(\$expr, \$package, [\$path = false], [\$text = false], 4) [[line 1580](#)]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of procedural page
- *string* **\$package** package name
- *string* **\$path** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getPageLink\(\)](#)

*void* function CHMdefaultConverter::getPageName(&\$element) [[line 945](#)]

**Function Parameters:**

- **&\$element**

*string* function CHMdefaultConverter::getRootTree(\$tree, \$package, [\$noparent = false], \$nouknownparent) [line [1134](#)]

**Function Parameters:**

- **array** **\$tree** output from [getSortedClassTreeFromClass\(\)](#)
- **string** **\$package** package
- **boolean** **\$nouknownparent** if true, an object's parent will not be checked
- **\$noparent**

return formatted class tree for the Class Trees page

- See Classes::\$definitechild, [CHMdefaultConverter::generateFormattedClassTrees\(\)](#)

*string* function CHMdefaultConverter::getSourceAnchor(\$sourcefile, \$anchor, [\$text = ""], [\$link = false]) [line [220](#)]

**Function Parameters:**

- **string** **\$sourcefile** full path to source file
- **string** **\$anchor** name of anchor
- **string** **\$text** link text, if this is a link
- **boolean** **\$link** returns either a link or a destination based on this parameter

Retrieve a Converter-specific anchor to a segment of a source code file parsed via a [@filesource](#) tag.

*void* function CHMdefaultConverter::getSourceLink(\$path) [line [204](#)]

**Function Parameters:**

- **\$path**

*void* function CHMdefaultConverter::getTutorialId(\$package, \$subpackage, \$tutorial, \$id) [line [764](#)]

**Function Parameters:**

- **\$package**
- **\$subpackage**
- **\$tutorial**

- **\$id**

*mixed* function CHMdefaultConverter::getVarLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], 5) [line [1614](#)]

**Function Parameters:**

- *boolean* **5** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of var
- *string* **\$class** class containing var
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getVarLink\(\)](#)

*void* function CHMdefaultConverter::Output() [line [1859](#)]

**Create the phpdoc.hhp, contents.hhc files needed by MS HTML Help Compiler to create a CHM file**

The output function generates the table of contents (contents.hhc) and file list (phpdoc.hhp) files used to create a .CHM by the free MS HTML Help compiler. {@, a list of all separate .html files is created in CHM format, and written to phpdoc.hhp. This list was generated by [writefile](#).

Next, a call to the table of contents:

```
12      phpDocumentor\_out("NOTE: to create the documentation.chm file, you
must now run Microsoft Help Workshop on phpdoc.hhp\n" );
13      phpDocumentor\_out("To get the free Microsoft Help Workshop, browse to:
http://go.microsoft.com/fwlink/?LinkId=14188\n" );
finishes things off}}
```

- [Link <http://www.microsoft.com/downloads/release.asp?releaseid=33071>](#)
- **TODO** use to directly call html help compiler hhc.exe
- **Uses [CHMdefaultConverter::generateTOC\(\)](#)** - assigns to the toc template variable

*void* function CHMdefaultConverter::postProcess(\$text) [[line 297](#)]

**Function Parameters:**

- **\$text**

## Uses htmlspecialchars() on the input

*string* function CHMdefaultConverter::ProgramExample(\$example, [\$tutorial = false], [\$inlinesourceparse = null], [\$class = null], [\$linenum = null], [\$filesourcepath = null]) [[line 266](#)]

**Function Parameters:**

- **string \$example**
- **boolean \$tutorial**
- **\$inlinesourceparse**
- **\$class**
- **\$linenum**
- **\$filesourcepath**

## Used to convert the <code> tag in a docblock

*int* function CHMdefaultConverter::rcNatCmp(\$a, \$b) [[line 1645](#)]

**Function Parameters:**

- **mixed \$a**
- **mixed \$b**

## does a nat case sort on the specified second level value of the array

*int* function CHMdefaultConverter::rcNatCmp1(\$a, \$b) [[line 1661](#)]

**Function Parameters:**

- **mixed \$a**
- **mixed \$b**

## does a nat case sort on the specified second level value of the array.

this one puts constructors first

*string* function CHMdefaultConverter::returnLink(\$link, \$text) [[line 383](#)]

**Function Parameters:**

- *string* **\$link**
- *string* **\$text**

*void* function CHMdefaultConverter::returnSee(&\$element, [\$eltext = false], [\$with\_a = true], 3) [[line 561](#)]

**Function Parameters:**

- *boolean* **3** determines whether the returned text is enclosed in an <a> tag
- **[abstractLink](#) &\$element** a descendant of abstractlink should be passed, and never text
- ***string* \$eltext** text to display in the link
- ***boolean* \$with\_a** this parameter is not used, and is deprecated

**This function takes a[abstractLink](#)descendant and returns an html link**

*void* function CHMdefaultConverter::setTargetDir(\$dir) [[line 1405](#)]

**Function Parameters:**

- **\$dir**

**calls the converter setTargetDir, and then copies any template images and the stylesheet if they haven't been copied**

- See [Converter::setTargetDir\(\)](#)

*void* function CHMdefaultConverter::setTemplateDir(\$dir) [[line 1354](#)]

**Function Parameters:**

- **\$dir**

*void* function CHMdefaultConverter::SmartyInit(&\$templ) [[line 318](#)]

**Function Parameters:**

- **&\$templ**

*string* function CHMdefaultConverter::sourceLine(\$linenumber, \$line, [\$path = false]) [[line 242](#)]

**Function Parameters:**

- *integer* **\$linenumber** line number
- *string* **\$line** highlighted source code line
- *false/string* **\$path** full path to @filesource file this line is a part of, if this is a single line from a complete file.

### Return a line of highlighted source code with formatted line number

If the \$path is a full path, then an anchor to the line number will be added as well

*void* function CHMdefaultConverter::TutorialExample(\$example) [[line 275](#)]

**Function Parameters:**

- *string* **\$example**

*void* function CHMdefaultConverter::unmangle(\$sourcecode) [[line 144](#)]

**Function Parameters:**

- **\$sourcecode**
  - **Deprecated** in favor of PHP 4.3.0+ tokenizer-based source highlighting

*void* function CHMdefaultConverter::writeExample(\$title, \$path, \$source) [[line 179](#)]

**Function Parameters:**

- **\$title**
- **\$path**
- **\$source**

*void* function CHMdefaultConverter::writefile(\$file, \$contents) [line [1691](#)]

**Function Parameters:**

- **\$file**
- **\$contents**

**Write a file to disk, and add it to the \$hhp\_files list of files to include in the generated CHM**

```
1 function writefile($file,$contents)
2 {
3     $this> addHHP$this> targetDir . PATH DELIMITER .$file);
4     Converter::writefile( $file,$contents);
5 }
```

*void* function CHMdefaultConverter::writeNewPPage(\$key) [line [491](#)]

**Function Parameters:**

- **\$key**

*void* function CHMdefaultConverter::writeSource(\$path, \$value) [line [160](#)]

**Function Parameters:**

- *string* **\$path** full path to the source file
- *string* **\$value** fully highlighted source code

# HTMLframesConverter.inc

**HTML original framed output converter, modified to use Smarty Template.**

This Converter takes output from the [Parser](#) and converts it to HTML-ready output for use with [Smarty](#).

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** HTMLframes
- **Author** Gregory Beaver < [celfog@php.net](mailto:celfog@php.net)>
- **Version** CVS: \$Id: HTMLframesConverter.inc 234145 2007-04-19 20:20:57Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- Since 1.2
- Filesource [Source Code for this file](#)
- License [LGPL](#)

# HTMLSmartyConverter.inc

## HTML output converter for Smarty Template.

This Converter takes output from the [Parser](#) and converts it to HTML-ready output for use with [Smarty](#).

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** HTMLframes
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: HTMLSmartyConverter.inc 234145 2007-04-19 20:20:57Z ashnazg \$
- **Copyright** 2000-2006 Joshua Eichorn, Gregory Beaver
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- Since 1.0rc1
- Filesource [Source Code for this file](#)
- License [LGPL](#)

# Class HTMLframesConverter

[line 53]

## HTML output converter.

This Converter takes output from the [Parser](#) and converts it to HTML-ready output for use with [Smarty](#).

- **Package** Converters
- **Sub-Package** HTMLframes
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: HTMLframesConverter.inc 234145 2007-04-19 20:20:57Z ashnazg \$
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#), [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- **Since** 1.2

### HTMLframesConverter::\$base\_dir

*string = [line 88]*

#### target directory passed on the command-line.

[\\$targetDir](#) is malleable, always adding package/ and package/subpackage/ subdirectories onto it.

### HTMLframesConverter::\$class\_dir

*string = [line 94]*

#### output directory for the current class being processed

### HTMLframesConverter::\$current

*array = [line 119]*

**contains all of the template procedural page element loop data needed for the current template**

### HTMLframesConverter::\$currentclass

*array = [line 125]*

**contains all of the template class element loop data needed for the current template**

#### **HTMLframesConverter::\$juststarted**

*boolean = false [line [113](#)]*

#### **controls formatting of parser informative output**

Converter prints: "Converting /path/to/file.php... Procedural Page Elements... Classes..." Since HTMLdefaultConverter outputs files while converting, it needs to send a \n to start a new line. However, if there is more than one class, output is messy, with multiple \n's just between class file output. This variable prevents that and is purely cosmetic

#### **HTMLframesConverter::\$leftindex**

*array = array('classes' => true, 'pages' => true, 'functions' => true, 'defines' => false, 'globals' => false) [line [75](#)]*

#### **indexes of elements by package that need to be generated**

#### **HTMLframesConverter::\$name**

*string = 'frames' [line [70](#)]*

#### **HTMLframesConverter::\$outputformat**

*string = 'HTML' [line [68](#)]*

#### **HTMLframesConverter::\$package\_pages**

*array = array() [line [101](#)]*

#### **array of converted package page names.**

Used to link to the package page in the left index

- **Var Format:** array(package => 1)

#### **HTMLframesConverter::\$page\_dir**

*string = [line [81](#)]*

#### **output directory for the current procedural page being processed**

## **HTMLframesConverter::\$processSpecialRoots**

*boolean = true [line [60](#)]*

**This converter knows about the new root tree processing**

In order to fix PEAR Bug #6389

## **HTMLframesConverter::\$ric\_set**

*mixed = array() [line [127](#)]*

## **HTMLframesConverter::\$sort\_page\_contents\_by\_type**

*boolean = true [line [66](#)]*

**Smarty Converter wants elements sorted by type as well as alphabetically**

- See [Converter::\\$sort\\_page\\_contents\\_by\\_type](#)

## **HTMLframesConverter::\$wrote**

*mixed = false [line [126](#)]*

Constructor void function HTMLframesConverter::HTMLframesConverter(&\$allp, &\$packp, &\$classes, &\$procpages, \$po, \$pp, \$qm, \$targetDir, \$templateDir, \$title) [line [133](#)]

**Function Parameters:**

- **&\$allp**
- **&\$packp**
- **&\$classes**
- **&\$procpages**
- **\$po**
- **\$pp**
- **\$qm**
- **\$targetDir**
- **\$templateDir**
- **\$title**

**sets \$base\_dir to \$targetDir**

- See [Converter::Converter\(\)](#)

void function HTMLframesConverter::convertClass(&\$element) [[line 1030](#)]

**Function Parameters:**

- [parserClass](#) &\$element

### Converts class for template output

- See [Converter::getFormattedInheritedMethods\(\)](#), [Converter::getFormattedInheritedVars\(\)](#)
- See [Converter::prepareDocBlock\(\)](#), [Converter::generateChildClassList\(\)](#),  
[HTMLframesConverter::generateFormattedClassTree\(\)](#), [Converter::getFormattedConflicts\(\)](#)

void function HTMLframesConverter::convertConst(&\$element) [[line 1058](#)]

**Function Parameters:**

- [parserDefine](#) &\$element

### Converts class variables for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

void function HTMLframesConverter::convertDefine(&\$element) [[line 1099](#)]

**Function Parameters:**

- [parserDefine](#) &\$element

## Converts defines for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void function HTMLframesConverter::ConvertErrorLog() [line 914]*

### Create errors.html template file output

This method takes all parsing errors and warnings and spits them out ordered by file and line number.

- **Global Variable Used** [ErrorTracker](#) \$phpDocumentor\_errors: We'll be using it's output facility

*void function HTMLframesConverter::convertFunction(&\$element) [line 1078]*

#### **Function Parameters:**

- [parserFunction](#) &\$element

## Converts function for template output

- See [Converter::prepareDocBlock\(\)](#), [parserFunction::getFunctionCall\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void function HTMLframesConverter::convertGlobal(&\$element) [line 1108]*

#### **Function Parameters:**

- [parserGlobal](#) &\$element

## Converts global variables for template output

*void* function `HTMLframesConverter::convertInclude(&$element)` [*line 1089*]

**Function Parameters:**

- *parserInclude* `&$element`

**Converts include elements for template output**

- See [Converter::prepareDocBlock\(\)](#)

*void* function `HTMLframesConverter::convertMethod(&$element)` [*line 1068*]

**Function Parameters:**

- *parserDefine* `&$element`

**Converts class methods for template output**

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function `HTMLframesConverter::convertPackagepage(&$element)` [*line 989*]

**Function Parameters:**

- *parserPackagePage* `&$element`

**Converts package page and sets its package as used** [package\\_pages](#)

*void* function `HTMLframesConverter::convertPage(&$element)` [*line 1118*]

**Function Parameters:**

- [\*parserData\*](#) &\$element

## converts procedural pages for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getClassesOnPage\(\)](#)

*void* function HTMLframesConverter::ConvertTodoList() [[line 879](#)]  
*void* function HTMLframesConverter::convertTutorial(&\$element) [[line 1006](#)]

### **Function Parameters:**

- [\*parserTutorial\*](#) &\$element

*void* function HTMLframesConverter::convertVar(&\$element) [[line 1048](#)]

### **Function Parameters:**

- [\*parserDefine\*](#) &\$element

## Converts class variables for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function HTMLframesConverter::Convert\_RIC(\$name, \$contents) [[line 869](#)]

### **Function Parameters:**

- *README|INSTALL|CHANGELOG* \$name
- *string* \$contents contents of the file

## Convert README/INSTALL/CHANGELOG file contents to output format

*void* function `HTMLframesConverter::copyMediaRecursively($media, $targetdir, [$subdir = ''])` [line [1558](#)]

**Function Parameters:**

- `$media`
- `$targetdir`
- `$subdir`

*void* function `HTMLframesConverter::endClass()` [line [334](#)]

**Writes out the template file \$class data and unsets the template to save memory**

- See [Converter::endClass\(\)](#)
- See `registerCurrentClass()`

*void* function `HTMLframesConverter::endPage()` [line [356](#)]

**Writes out the template file \$page data and unsets the template to save memory**

- See [Converter::endPage\(\)](#)
- See `registerCurrent()`

*void* function `HTMLframesConverter::formatIndex()` [line [626](#)]

**HTMLDefaultConverter uses this function to format template index.html and packages.html**

This function generates the package list from [\\$all\\_packages](#), eliminating any packages that don't have any entries in their package index (no files at all, due to @ignore or other factors). Then it uses the default package name as the first package index to display. It sets the right pane to be either a blank file with instructions on making package-level docs, or the package-level docs for the default package.

- **Global Variable Used** string `$phpDocumentor_DefaultPackageName`: Used to set the starting package to display

*void* function **HTMLframesConverter::formatLeftIndex()** [*line 726*]

### **Generate indexes for li\_package.html and classtree output files**

This function generates the li\_package.html files from the template file left.html. It does this by iterating through each of the \$page\_elements, \$class\_elements and \$function\_elements arrays to retrieve the pre-sorted [abstractLink](#) descendants needed for index generation. Conversion of these links to text is done by [returnSee\(\)](#). The \$local parameter is set to false to ensure that paths are correct.

Then it uses [generateFormattedClassTrees\(\)](#) to create class trees from the template file classtrees.html. Output filename is classtrees\_packagename.html. This function also unsets [\\$elements](#) and [\\$pkg\\_elements](#) to free up the considerable memory these two class vars use

- See [Converter::\\$page\\_elements](#), [Converter::\\$class\\_elements](#), [Converter::\\$function\\_elements](#)

*void* function **HTMLframesConverter::formatPkgIndex()** [*line 597*]

### **HTMLdefaultConverter chooses to format both package indexes and the complete index here**

This function formats output for the elementindex.html and pkgelementindex.html template files. It then writes them to the target directory

- See [HTMLframesConverter::generateElementIndex\(\)](#),  
[HTMLframesConverter::generatePkgElementIndex\(\)](#)

*string* function **HTMLframesConverter::formatTutorialTOC(\$toc)** [*line 306*]

#### **Function Parameters:**

- **array \$toc** format: array(array('tagname' => section, 'link' => returnsee link, 'id' => anchor name, 'title' => from title tag),...)

**Use the template tutorial\_toc.tpl to generate a table of contents for HTML**

*void* function **HTMLframesConverter::generateElementIndex()** [line [1525](#)]

**Generate alphabetical index of all elements**

- See [Converter::\\$elements](#), [Converter::walk\(\)](#)

*array* function **HTMLframesConverter::generateFormattedClassTree(\$class)** [line [1143](#)]

**Function Parameters:**

- [parserClass \\$class](#) class variable

**returns an array containing the class inheritance tree from the root object to the class**

- Uses [parserClass::getParentClassTree\(\)](#)

*void* function **HTMLframesConverter::generateFormattedClassTrees(\$package)** [line [1207](#)]

**Function Parameters:**

- [string \\$package](#) package to generate a class tree for

**returns a template-enabled array of class trees**

- See [\\$roots](#), [HTMLConverter::getRootTree\(\)](#)

*void* function **HTMLframesConverter::generateFormattedInterfaceTrees(\$package)** [line [1261](#)]

**Function Parameters:**

- *string \$package* package to generate a class tree for

**returns a template-enabled array of interface inheritance trees**

- See \$roots, `HTMLConverter::getRootTree()`

*void* function `HTMLframesConverter::generatePkgElementIndex($package)` [[line 1623](#)]

**Function Parameters:**

- *string \$package* name of a package

**Generate alphabetical index of all elements by package and subpackage**

- See [Converter::\\$pkg\\_elements](#), [Converter::walk\(\)](#),  
`HTMLframesConverter::generatePkgElementIndexes()`

*void* function `HTMLframesConverter::generatePkgElementIndexes()` [[line 1674](#)]

- See [HTMLframesConverter::generatePkgElementIndex\(\)](#)

*void* function `HTMLframesConverter::getCData($value)` [[line 980](#)]

**Function Parameters:**

- **\$value**

*mixed* function `HTMLframesConverter::getClassLink($expr, $package, [$file = false], [$text = false], [$with_a = true])` [[line 1714](#)]

**Function Parameters:**

- *string \$expr* name of class
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated
- *boolean \$with\_a* return just the URL, or enclose it in an html a tag

- See [Converter::getClassLink\(\)](#)

*mixed* function `HTMLframesConverter::getConstLink($expr, $class, $package, [$file = false], [$text = false], 5)` [[line 1829](#)]

**Function Parameters:**

- *boolean 5* return just the URL, or enclose it in an html a tag
- *string \$expr* name of class constant
- *string \$class* class containing class constant
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getVarLink\(\)](#)

*void* function `HTMLframesConverter::getCurrentPageLink()` [[line 284](#)]

*mixed* function `HTMLframesConverter::getDefineLink($expr, $package, [$file = false], [$text = false], 4)` [[line 1746](#)]

**Function Parameters:**

- *boolean 4* return just the URL, or enclose it in an html a tag
- *string \$expr* name of define
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getDefineLink\(\)](#)

void function HTMLframesConverter::getExampleLink(\$path, \$title) [*line 194*]

**Function Parameters:**

- **\$path**
- **\$title**

mixed function HTMLframesConverter::getFunctionLink(\$expr, \$package, [\$file = false], [\$text = false], 4) [*line 1730*]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of function
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getFunctionLink\(\)](#)

mixed function HTMLframesConverter::getGlobalLink(\$expr, \$package, [\$file = false], [\$text = false], 4) [*line 1762*]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of global variable
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getGlobalLink\(\)](#)

*void* function `HTMLframesConverter::getId($element, [$fullpath = true])` [line [811](#)]

**Function Parameters:**

- **\$element**
- **\$fullpath**

*array* function `HTMLframesConverter::getIndexInformation($elt)` [line [1408](#)]

**Function Parameters:**

- [\*parserElement\*](#) **\$elt** descendant of `parserElement`

## Generate indexing information for given element

- See [`HTMLframesConverter::generateElementIndex\(\)`](#)

*mixed* function `HTMLframesConverter::getMethodLink($expr, $class, $package, [$file = false], [$text = false], 5)` [line [1795](#)]

**Function Parameters:**

- *boolean* **5** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of method
- *string* **\$class** class containing method
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [`Converter::getMethodLink\(\)`](#)

*mixed* function `HTMLframesConverter::getPageLink($expr, $package, [$path = false], [$text = false], 4)` [line [1778](#)]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag

- *string \$expr* name of procedural page
- *string \$package* package name
- *string \$path* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getPageLink\(\)](#)

*void* function `HTMLframesConverter::getPageName(&$element)` [[line 1129](#)]

**Function Parameters:**

- **&\$element**

*string* function `HTMLframesConverter::getRootTree($tree, $package, [$noparent = false], $nouknownparent)` [[line 1318](#)]

**Function Parameters:**

- *array \$tree* output from [getSortedClassTreeFromClass\(\)](#)
- *string \$package* package
- *boolean \$nouknownparent* if true, an object's parent will not be checked
- *\$noparent*

**return formatted class tree for the Class Trees page**

- See `Classes::$definitechild`, [HTMLframesConverter::generateFormattedClassTrees\(\)](#)

*string* function `HTMLframesConverter::getSourceAnchor($sourcefile, $anchor, [$text = ""], [$link = false])` [[line 215](#)]

**Function Parameters:**

- *string \$sourcefile* full path to source file
- *string \$anchor* name of anchor
- *string \$text* link text, if this is a link
- *boolean \$link* returns either a link or a destination based on this parameter

**Retrieve a Converter-specific anchor to a segment of a source code file parsed via a [@filesource](#) tag.**

*void* function `HTMLframesConverter::getSourceLink($path)` [[line 199](#)]

**Function Parameters:**

- `$path`

*void* function `HTMLframesConverter::getTutorialId($package, $subpackage, $tutorial, $id)` [[line 975](#)]

**Function Parameters:**

- `$package`
- `$subpackage`
- `$tutorial`
- `$id`

*void* function `HTMLframesConverter::getTutorialTree($tutorial, [$k = false])` [[line 548](#)]

**Function Parameters:**

- `$tutorial`
- `$k`

*mixed* function `HTMLframesConverter::getVarLink($expr, $class, $package, [$file = false], [$text = false], 5)` [[line 1812](#)]

**Function Parameters:**

- *boolean* `5` return just the URL, or enclose it in an html a tag
- *string* `$expr` name of var
- *string* `$class` class containing var
- *string* `$package` package name
- *string* `$file` full path to look in (used in index generation)
- *boolean* `$text` deprecated

- See [Converter::getVarLink\(\)](#)

*void* function `HTMLframesConverter::makeLeft()` [line [380](#)]  
*void* function `HTMLframesConverter::Output()` [line [1886](#)]

**This function is not used by `HTMLdefaultConverter`, but is required by Converter**

*void* function `HTMLframesConverter::postProcess($text)` [line [292](#)]

**Function Parameters:**

- `$text`

### Uses `htmlspecialchars()` on the input

*string* function `HTMLframesConverter::ProgramExample($example, [$tutorial = false], [$inlinesourceparse = null], [$class = null], [$linenum = null], [$filesourcepath = null])` [line [260](#)]

**Function Parameters:**

- `string $example`
- `boolean $tutorial`
- `$inlinesourceparse`
- `$class`
- `$linenum`
- `$filesourcepath`

### Used to convert the `<code>` tag in a docblock

*int* function `HTMLframesConverter::rcNatCmp($a, $b)` [line [1843](#)]

**Function Parameters:**

- `mixed $a`
- `mixed $b`

### does a nat case sort on the specified second level value of the array

*int* function `HTMLframesConverter::rcNatCmp1($a, $b)` [line [1859](#)]

**Function Parameters:**

- `mixed $a`
- `mixed $b`

**does a nat case sort on the specified second level value of the array.**

this one puts constructors first

*string* function `HTMLframesConverter::returnLink($link, $text)` [line [375](#)]

**Function Parameters:**

- *string* `$link`
- *string* `$text`

*void* function `HTMLframesConverter::returnSee(&$element, [$eltext = false], [$with_a = true], 3)` [line [781](#)]

**Function Parameters:**

- *boolean* `3` determines whether the returned text is enclosed in an `<a>` tag
- [\*\*abstractLink\*\*](#) &`$element` a descendant of abstractlink should be passed, and never text
- *string* `$eltext` text to display in the link
- *boolean* `$with_a` this parameter is not used, and is deprecated

**This function takes a[\*\*abstractLink\*\*](#)descendant and returns an html link**

*void* function `HTMLframesConverter::setTargetDir($dir)` [line [1603](#)]

**Function Parameters:**

- `$dir`

**calls the converter `setTargetDir`, and then copies any template images and the stylesheet if they haven't been copied**

- See [`Converter::setTargetDir\(\)`](#)

*void* function `HTMLframesConverter::SmartyInit(&$templ)` [line [313](#)]

**Function Parameters:**

- `&$templ`

*string* function `HTMLframesConverter::sourceLine($linenumber, $line, [$path = false])` [line [236](#)]

**Function Parameters:**

- *integer* **\$linenumber** line number
- *string* **\$line** highlighted source code line
- *false|string* **\$path** full path to @filesource file this line is a part of, if this is a single line from a complete file.

### Return a line of highlighted source code with formatted line number

If the \$path is a full path, then an anchor to the line number will be added as well

*void* function `HTMLframesConverter::TutorialExample($example)` [line [270](#)]

**Function Parameters:**

- *string* **\$example**

*void* function `HTMLframesConverter::unmangle($sourcecode)` [line [142](#)]

**Function Parameters:**

- **\$sourcecode**
- 
- **Deprecated** in favor of PHP 4.3.0+ tokenizer-based source highlighting

*void* function `HTMLframesConverter::writeExample($title, $path, $source)` [line [176](#)]

**Function Parameters:**

- **\$title**
- **\$path**
- **\$source**

*void* function `HTMLframesConverter::writeNewPPage($key)` [line [695](#)]

**Function Parameters:**

- **\$key**

*void* function **HTMLframesConverter::writeSource(\$path, \$value)** [[line 158](#)]

**Function Parameters:**

- *string* **\$path** full path to the source file
- *string* **\$value** fully highlighted source code

## Class **HTMLSmartyConverter** [[line 54](#)]

**HTML output converter.**

This Converter takes output from the [Parser](#) and converts it to HTML-ready output for use with [Smarty](#).

- **Package** Converters
- **Sub-Package** [HTMLSmarty](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Revision: 234145 \$
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#), [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- **Since** 1.0rc1

**HTMLSmartyConverter::\$base\_dir**

*string* = [[line 89](#)]

**target directory passed on the command-line.**

[\\$targetDir](#) is malleable, always adding package/ and package/subpackage/ subdirectories onto it.

**HTMLSmartyConverter::\$class\_dir**

*string* = [line [95](#)]

**output directory for the current class being processed**

**HTMLSmartyConverter::\$current**

*array* = [line [120](#)]

**contains all of the template procedural page element loop data needed for the current template**

**HTMLSmartyConverter::\$currentclass**

*array* = [line [126](#)]

**contains all of the template class element loop data needed for the current template**

**HTMLSmartyConverter::\$juststarted**

*boolean* = false [line [114](#)]

**controls formatting of parser informative output**

Converter prints: "Converting /path/to/file.php... Procedural Page Elements... Classes..." Since HTMLdefaultConverter outputs files while converting, it needs to send a \n to start a new line. However, if there is more than one class, output is messy, with multiple \n's just between class file output. This variable prevents that and is purely cosmetic

**HTMLSmartyConverter::\$leftindex**

*array* = array('classes' => true, 'pages' => true, 'functions' => true, 'defines' => false, 'globals' => false)  
[line [76](#)]

**indexes of elements by package that need to be generated**

**HTMLSmartyConverter::\$name**

*string* = 'Smarty' [line [71](#)]

**HTMLSmartyConverter::\$outputformat**

*string* = 'HTML' [line [69](#)]

**HTMLSmartyConverter::\$package\_pages**

*array* = array() [line [102](#)]

**array of converted package page names.**

Used to link to the package page in the left index

- **Var Format:** array(package => 1)

### **HTMLSmartyConverter::\$page\_dir**

*string = [line [82](#)]*

**output directory for the current procedural page being processed**

### **HTMLSmartyConverter::\$processSpecialRoots**

*boolean = true [line [61](#)]*

**This converter knows about the new root tree processing**

In order to fix PEAR Bug #6389

### **HTMLSmartyConverter::\$ric\_set**

*mixed = array() [line [128](#)]*

### **HTMLSmartyConverter::\$sort\_page\_contents\_by\_type**

*boolean = true [line [67](#)]*

**Smarty Converter wants elements sorted by type as well as alphabetically**

- See [Converter::\\$sort\\_page\\_contents\\_by\\_type](#)

### **HTMLSmartyConverter::\$wrote**

*mixed = false [line [127](#)]*

Constructor **void** function **HTMLSmartyConverter::HTMLSmartyConverter(&\$allp, &\$packp, &\$classes, &\$procpages, \$po, \$pp, \$qm, \$targetDir, \$templateDir, \$title)** [line [148](#)]

**Function Parameters:**

- `&$allp`
- `&$packp`
- `&$classes`
- `&$procpages`
- `$po`
- `$pp`
- `$qm`
- `$targetDir`
- `$templateDir`
- `$title`

*void* function `HTMLSmartyConverter::convertClass(&$element)` [[line 1078](#)]

**Function Parameters:**

- `parserClass &$element`

## Converts class for template output

- See [Converter::getFormattedInheritedMethods\(\)](#), [Converter::getFormattedInheritedVars\(\)](#)
- See [Converter::prepareDocBlock\(\)](#), [Converter::generateChildClassList\(\)](#),  
[HTMLSmartyConverter::generateFormattedClassTree\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function `HTMLSmartyConverter::convertConst(&$element)` [[line 1106](#)]

**Function Parameters:**

- `parserDefine &$element`

## Converts class variables for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function `HTMLSmartyConverter::convertDefine(&$element)` [[line 1147](#)]

**Function Parameters:**

- [\*parserDefine\*](#) &\$element

## Converts defines for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function HTMLSmartyConverter::ConvertErrorLog() [[line 928](#)]

### Create errors.html template file output

This method takes all parsing errors and warnings and spits them out ordered by file and line number.

- Global Variable Used [ErrorTracker](#) \$phpDocumentor\_errors: We'll be using it's output facility

*void* function HTMLSmartyConverter::convertFunction(&\$element) [[line 1126](#)]

### Function Parameters:

- [\*parserFunction\*](#) &\$element

## Converts function for template output

- See [Converter::prepareDocBlock\(\)](#), [parserFunction::getFunctionCall\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function HTMLSmartyConverter::convertGlobal(&\$element) [[line 1157](#)]

### Function Parameters:

- *parserGlobal* &\$element

## Converts global variables for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

void function HTMLSmartyConverter::convertInclude(&\$element) [[line 1137](#)]

### Function Parameters:

- *parserInclude* &\$element

## Converts include elements for template output

- See [Converter::prepareDocBlock\(\)](#)

void function HTMLSmartyConverter::convertMethod(&\$element) [[line 1116](#)]

### Function Parameters:

- *parserDefine* &\$element

## Converts class methods for template output

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

void function HTMLSmartyConverter::convertPackagepage(&\$element) [[line 1004](#)]

**Function Parameters:**

- *parserPackagePage* &\$element

**Converts package page and sets its package as used** [\\$package\\_pages](#)

*void* function `HTMLSmartyConverter::convertPage(&$element)` [*line 1167*]

**Function Parameters:**

- *parserData* &\$element

**converts procedural pages for template output**

- See [Converter::prepareDocBlock\(\)](#), [Converter::getClassesOnPage\(\)](#)

*void* function `HTMLSmartyConverter::ConvertTodoList()` [*line 893*]

*void* function `HTMLSmartyConverter::convertTutorial(&$element)` [*line 1034*]

**Function Parameters:**

- *parserTutorial* &\$element

*void* function `HTMLSmartyConverter::convertVar(&$element)` [*line 1096*]

**Function Parameters:**

- *parserDefine* &\$element

**Converts class variables for template output**

- See [Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function `HTMLSmartyConverter::Convert_RIC($name, $contents)` [[line 872](#)]

**Function Parameters:**

- `README|INSTALL|CHANGELOG $name`
- `string $contents` contents of the file

## Convert README/INSTALL/CHANGELOG file contents to output format

*void* function `HTMLSmartyConverter::copyMediaRecursively($media, $targetdir, [$subdir = "])` [[line 1611](#)]

**Function Parameters:**

- `$media`
- `$targetdir`
- `$subdir`

*void* function `HTMLSmartyConverter::endClass()` [[line 344](#)]

**Writes out the template file [\\$class](#) and unsets the template to save memory**

- See [Converter::endClass\(\)](#)
- See [registerCurrentClass\(\)](#)

*void* function `HTMLSmartyConverter::endPage()` [[line 523](#)]

**Writes out the template file [\\$page](#) and unsets the template to save memory**

- See [Converter::endPage\(\)](#)
- See [registerCurrent\(\)](#)

*void* function `HTMLSmartyConverter::formatIndex()` [[line 642](#)]

**HTMLdefaultConverter uses this function to format template index.html and packages.html**

This function generates the package list from [\\$all packages](#), eliminating any packages that don't have any entries in their package index (no files at all, due to @ignore

or other factors). Then it uses the default package name as the first package index to display. It sets the right pane to be either a blank file with instructions on making package-level docs, or the package-level docs for the default package.

- **Global Variable Used** string \$phpDocumentor\_DefaultPackageName: Used to set the starting package to display

void function HTMLSmartyConverter::formatLeftIndex() [[line 740](#)]

### Generate indexes for li\_package.html and classtree output files

This function generates the li\_package.html files from the template file left.html. It does this by iterating through each of the \$page\_elements, \$class\_elements and \$function\_elements arrays to retrieve the pre-sorted [abstractLink](#) descendants needed for index generation. Conversion of these links to text is done by [returnSee\(\)](#).

Then it uses [generateFormattedClassTrees\(\)](#) to create class trees from the template file classtrees.html. Output filename is classtrees\_packagename.html. This function also unsets [\\$elements](#) and [\\$pkg\\_elements](#) to free up the considerable memory these two class vars use

- See [Converter::\\$page\\_elements](#), [Converter::\\$class\\_elements](#), [Converter::\\$function\\_elements](#)

void function HTMLSmartyConverter::formatPkgIndex() [[line 606](#)]

### HTMLdefaultConverter chooses to format both package indexes and the complete index here

This function formats output for the elementindex.html and pkgelementindex.html template files. It then writes them to the target directory

- See [HTMLSmartyConverter::generateElementIndex\(\)](#),  
[HTMLSmartyConverter::generatePkgElementIndex\(\)](#)

*string* function `HTMLSmartyConverter::formatTutorialTOC($toc)` [line [321](#)]

**Function Parameters:**

- ***array \$toc*** format: array(array('tagname' => section, 'link' => returnsee link, 'id' => anchor name, 'title' => from title tag),...)

**Use the template `tutorial_toc.tpl` to generate a table of contents for HTML**

*void* function `HTMLSmartyConverter::generateElementIndex()` [line [1578](#)]

**Generate alphabetical index of all elements**

- See [Converter::\\$elements](#), [Converter::walk\(\)](#)

*array* function `HTMLSmartyConverter::generateFormattedClassTree($class)` [line [1195](#)]

**Function Parameters:**

- ***parserClass \$class*** class variable

**returns an array containing the class inheritance tree from the root object to the class**

- Uses [parserClass::getParentClassTree\(\)](#)

*void* function `HTMLSmartyConverter::generateFormattedClassTrees($package)` [line [1259](#)]

**Function Parameters:**

- ***string \$package*** package to generate a class tree for

**returns a template-enabled array of class trees**

- See \$roots, `HTMLConverter::getRootTree()`

`void` function `HTMLSmartyConverter::generateFormattedInterfaceTrees($package)` [[line 1313](#)]

**Function Parameters:**

- *string* **\$package** package to generate a class tree for

**returns a template-enabled array of interface inheritance trees**

- See \$roots, `HTMLConverter::getRootTree()`

`void` function `HTMLSmartyConverter::generatePkgElementIndex($package)` [[line 1676](#)]

**Function Parameters:**

- *string* **\$package** name of a package

**Generate alphabetical index of all elements by package and subpackage**

- See [Converter::\\$pkg\\_elements](#), [Converter::walk\(\)](#),  
[HTMLSmartyConverter::generatePkgElementIndexes\(\)](#)

`void` function `HTMLSmartyConverter::generatePkgElementIndexes()` [[line 1726](#)]

- See [HTMLSmartyConverter::generatePkgElementIndex\(\)](#)

*void* function `HTMLSmartyConverter::getCDATA($value)` [[line 990](#)]

**Function Parameters:**

- **\$value**

*void* function `HTMLSmartyConverter::getClassContents()` [[line 466](#)]

*void* function `HTMLSmartyConverter::getClassLeft()` [[line 436](#)]

*mixed* function `HTMLSmartyConverter::getClassLink($expr, $package, [$file = false], [$text = false], [$with_a = true])` [[line 1766](#)]

**Function Parameters:**

- *string* **\$expr** name of class
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$with\_a** return just the URL, or enclose it in an html a tag

- See [Converter::getClassLink\(\)](#)

*mixed* function `HTMLSmartyConverter::getConstLink($expr, $class, $package, [$file = false], [$text = false], 5)` [[line 1881](#)]

**Function Parameters:**

- *boolean* **5** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of class constant
- *string* **\$class** class containing class constant
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getVarLink\(\)](#)

*void* function `HTMLSmartyConverter::getCurrentPageLink()` [[line 287](#)]

*mixed* function `HTMLSmartyConverter::getDefineLink($expr, $package, [$file = false], [$text = false], 4)` [[line](#)

[1798](#)

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of define
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getDefineLink\(\)](#)

*void* function `HTMLSmartyConverter::getExampleLink($path, $title)` [[line 190](#)]

**Function Parameters:**

- **\$path**
- **\$title**

*mixed* function `HTMLSmartyConverter::getFunctionLink($expr, $package, [$file = false], [$text = false], 4)` [[line 1782](#)]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of function
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getFunctionLink\(\)](#)

*mixed* function `HTMLSmartyConverter::getGlobalLink($expr, $package, [$file = false], [$text = false], 4)` [[line 1814](#)]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag

- *string \$expr* name of global variable
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getGlobalLink\(\)](#)

*void* function `HTMLSmartyConverter::getId($element, [$fullpath = true])` [line [814](#)]

**Function Parameters:**

- **\$element**
- **\$fullpath**

*array* function `HTMLSmartyConverter::getIndexInformation($elt)` [line [1460](#)]

**Function Parameters:**

- [\*parserElement\* \\$elt](#) descendant of parserElement

## Generate indexing information for given element

- See [HTMLSmartyConverter::generateElementIndex\(\)](#)

*mixed* function `HTMLSmartyConverter::getMethodLink($expr, $class, $package, [$file = false], [$text = false], 5)` [line [1847](#)]

**Function Parameters:**

- *boolean 5* return just the URL, or enclose it in an html a tag
- *string \$expr* name of method
- *string \$class* class containing method
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getMethodLink\(\)](#)

*void* function **HTMLSmartyConverter::getPageContents()** [*line 482*]

*void* function **HTMLSmartyConverter::getPageLeft()** [*line 499*]

*mixed* function **HTMLSmartyConverter::getPageLink(\$expr, \$package, [\$path = false], [\$text = false], 4)** [*line 1830*]

**Function Parameters:**

- *boolean* **4** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of procedural page
- *string* **\$package** package name
- *string* **\$path** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getPageLink\(\)](#)

*void* function **HTMLSmartyConverter::getPageName(&\$element)** [*line 1181*]

**Function Parameters:**

- **&\$element**

*string* function **HTMLSmartyConverter::getRootTree(\$tree, \$package, [\$noparent = false], \$nounknownparent)** [*line 1370*]

**Function Parameters:**

- *array* **\$tree** output from [getSortedClassTreeFromClass\(\)](#)
- *string* **\$package** package
- *boolean* **\$nounknownparent** if true, an object's parent will not be checked
- **\$noparent**

**return formatted class tree for the Class Trees page**

- See Classes::\$definitechild, [HTMLSmartyConverter::generateFormattedClassTrees\(\)](#)

*string* function HTMLSmartyConverter::getSourceAnchor(\$sourcefile, \$anchor, [\$text = ""], [\$link = false]) [line 211]

**Function Parameters:**

- *string* **\$sourcefile** full path to source file
- *string* **\$anchor** name of anchor
- *string* **\$text** link text, if this is a link
- *boolean* **\$link** returns either a link or a destination based on this parameter

**Retrieve a Converter-specific anchor to a segment of a source code file parsed via a [@filesource](#) tag.**

*void* function HTMLSmartyConverter::getSourceLink(\$path) [line 195]

**Function Parameters:**

- **\$path**

*void* function HTMLSmartyConverter::getTutorialId(\$package, \$subpackage, \$tutorial, \$id) [line 995]

**Function Parameters:**

- **\$package**
- **\$subpackage**
- **\$tutorial**
- **\$id**

*void* function HTMLSmartyConverter::getTutorialList() [line 369]

*void* function HTMLSmartyConverter::getTutorialTree(\$tutorial, [\$k = false]) [line 396]

**Function Parameters:**

- **\$tutorial**
- **\$k**

*mixed* function HTMLSmartyConverter::getVarLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], 5) [line 1864]

**Function Parameters:**

- *boolean* **5** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of var
- *string* **\$class** class containing var
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getVarLink\(\)](#)

*void* function **HTMLSmartyConverter::makeLeft()** [line [554](#)]

*void* function **HTMLSmartyConverter::Output()** [line [1938](#)]

**This function is not used by `HTMLdefaultConverter`, but is required by `Converter`**

*void* function **HTMLSmartyConverter::postProcess(\$text)** [line [307](#)]

**Function Parameters:**

- **\$text**

**Uses `htmlspecialchars()` on the input**

*string* function **HTMLSmartyConverter::ProgramExample(\$example, [\$tutorial = false], [\$inlinesourceparse = null], [\$class = null], [\$linenum = null], [\$filesourcopath = null])** [line [255](#)]

**Function Parameters:**

- *string* **\$example**
- *boolean* **\$tutorial**
- **\$inlinesourceparse**
- **\$class**
- **\$linenum**
- **\$filesourcopath**

**Used to convert the <code> tag in a docblock**

*int* function **HTMLSmartyConverter::rcNatCmp(\$a, \$b)** [line [1895](#)]

**Function Parameters:**

- *mixed \$a*
- *mixed \$b*

**does a nat case sort on the specified second level value of the array**

*int* function `HTMLSmartyConverter::rcNatCmp1($a, $b)` [[line 1911](#)]

**Function Parameters:**

- *mixed \$a*
- *mixed \$b*

**does a nat case sort on the specified second level value of the array.**

this one puts constructors first

*string* function `HTMLSmartyConverter::returnLink($link, $text)` [[line 549](#)]

**Function Parameters:**

- *string \$link*
- *string \$text*

*void* function `HTMLSmartyConverter::returnSee(&$element, [$eltext = false], [$with_a = true], 3)` [[line 784](#)]

**Function Parameters:**

- *boolean 3* determines whether the returned text is enclosed in an &lt;a> tag
- *[abstractLink](#) &\$element* a descendant of abstractlink should be passed, and never text
- *string \$eltext* text to display in the link
- *boolean \$with\_a* this parameter is not used, and is deprecated

**This function takes a[abstractLink](#)descendant and returns an html link**

*void* function `HTMLSmartyConverter::setTargetDir($dir)` [[line 1656](#)]

**Function Parameters:**

- *\$dir*

**calls the converter `setTargetDir`, and then copies any template images and the stylesheet if they haven't been copied**

- See [Converter::setTargetDir\(\)](#)

*void* function `HTMLSmartyConverter::SmartyInit(&$templ)` [[line 328](#)]

**Function Parameters:**

- **&\$templ**

*string* function `HTMLSmartyConverter::sourceLine($linenumber, $line, [$path = false])` [[line 232](#)]

**Function Parameters:**

- *integer* **\$linenumber** line number
- *string* **\$line** highlighted source code line
- *false|string* **\$path** full path to @filesource file this line is a part of,      if this is a single line from a complete file.

**Return a line of highlighted source code with formatted line number**

If the `$path` is a full path, then an anchor to the line number will be added as well

*void* function `HTMLSmartyConverter::TutorialExample($example)` [[line 273](#)]

**Function Parameters:**

- *string* **\$example**

*void* function `HTMLSmartyConverter::unmangle($sourcecode)` [[line 292](#)]

**Function Parameters:**

- **\$sourcecode**

*void* function `HTMLSmartyConverter::writeExample($title, $path, $source)` [[line 172](#)]

**Function Parameters:**

- `$title`
- `$path`
- `$source`

*void* function `HTMLSmartyConverter::writeNewPPage($key)` [[line 706](#)]

**Function Parameters:**

- `$key`

*void* function `HTMLSmartyConverter::writeRIC()` [[line 878](#)]

*void* function `HTMLSmartyConverter::writeSource($path, $value)` [[line 154](#)]

**Function Parameters:**

- `$path`
- `$value`

# class.phpdocpdf.php

**Cezpdf callback class customized for phpDocumentor**  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** PDFdefault
- **Author** Greg Beaver <[cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: class.phpdocpdf.php 212211 2006-04-30 22:18:14Z cellog \$
- **Copyright** 2000-2006 Joshua Eichorn, Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.2
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

include\_once '[phpDocumentor/Converters/PDF/default/ParserPDF.ifile](#)' [42\]](#)

include\_once '[phpDocumentor/Converters/PDF/default/class.ezpdf.php](#)' [41\]](#)

## ezPdf libraries

## ParserPDF.inc

This class handles the XML-based CezPDF markup language created to allow templates for the PDFdefaultConverter  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** PDFdefault
- **Author** Greg Beaver < [cellog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: ParserPDF.inc 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.2
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

PHPDOCUMENTOR\_PDF\_EVENT\_CONTENT = 601 [line [45](#)]  
**used for parsing stuff between <text>**

PHPDOCUMENTOR\_PDF\_EVENT\_FONT = 602 [line [49](#)]  
**when <font> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_EVENT\_NEWPAGE = 603 [line [53](#)]

**when <newpage/> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_EVENT\_PDFFUNCTION = 604 [line [57](#)]

**when <pdffunction> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_EVENT\_TEXT = 600 [line [41](#)]

**when <text> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_STATE\_CONTENT = 701 [line [47](#)]

**used for parsing stuff between <text>**

PHPDOCUMENTOR\_PDF\_STATE\_FONT = 702 [line [51](#)]

**when <font> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_STATE\_NEWPAGE = 703 [line [55](#)]

**when <newpage/> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_STATE\_PDFFUNCTION = 704 [line [59](#)]

**when <pdffunction> is found in an ezText input**

PHPDOCUMENTOR\_PDF\_STATE\_TEXT = 700 [line [43](#)]

**when <text> is found in an ezText input**

# PDFdefaultConverter.inc

Outputs documentation in PDF format

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** PDFdefault
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: PDFdefaultConverter.inc 236747 2007-05-31 02:02:42Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.2
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

include\_once '[phpDocumentor/Converters/PDF/default/class.phpdocpdf.php](#)' [line 42]

## The Cezpdf class library

# Class PDFdefaultConverter

[line [55](#)]

## PDF output converter.

This Converter takes output from the [Parser](#) and converts it to PDF-ready output for use with [Cezpdf](#). This is now beta code

- **Package** Converters
- **Sub-Package** PDFdefault
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: PDFdefaultConverter.inc 236747 2007-05-31 02:02:42Z ashnazg \$
- **TODO** Implement links to conflicts/inheritance
- **Since** 1.1

## PDFdefaultConverter::\$classpackage\_pagenums

*mixed* = array() [line [65](#)]

## PDFdefaultConverter::\$curclasspackage

*mixed* = false [line [71](#)]

## PDFdefaultConverter::\$curpagepackage

*mixed* = false [line [70](#)]

## PDFdefaultConverter::\$leftindex

*mixed* = array('classes' => false, 'pages' => false, 'functions' => false, 'defines' => false, 'globals' => false) [line [63](#)]

## PDFdefaultConverter::\$name

*string* = 'default' [line [69](#)]

- **Var** always default

## PDFdefaultConverter::\$outputformat

*string* = 'PDF' [line [67](#)]

- Var always PDF

### PDFdefaultConverter::\$pagepackage\_pagenums

*mixed = array() [line 64]*

### PDFdefaultConverter::\$pdf

*Cezpdf = false [line 76]*

### PDFdefaultConverter::\$ric\_set

*mixed = array() [line 77]*

### PDFdefaultConverter::\$smarty\_dir

*mixed = [line 72]*

### PDFdefaultConverter::\$sort\_absolutely\_everything

*boolean = true [line 62]*

**default PDF Converter wants elements sorted by type as well as alphabetically**

- See [Converter::\\$sort\\_page\\_contents\\_by\\_type](#)

### PDFdefaultConverter::\$\_sourcecode

*array = [line 84]*

**Source files for appendix C are stored here**

Format:      array(array(package       =>       packagename,       code       =>  
array(highlightedsource code 1, ...)))

Constructor void function PDFdefaultConverter::PDFdefaultConverter(&\$allp, &\$packp, &\$classes,  
&\$procpages, \$po, \$pp, \$qm, \$targetDir, \$templateDir, \$title) [line 88]

**Function Parameters:**

- &\$allp
- &\$packp

- **&\$classes**
- **&\$procpages**
- **\$po**
- **\$pp**
- **\$qm**
- **\$targetDir**
- **\$templateDir**
- **\$title**

- See [Converter::Converter\(\)](#)

*void* function PDFdefaultConverter::convertClass(&\$element) [[line 461](#)]

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::convertConst(&\$element) [[line 439](#)]

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::convertDefine(&\$element) [[line 541](#)]

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::convertDocBlock(&\$element) [[line 249](#)]

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::convertFunction(&\$element) [[line 508](#)]

**Function Parameters:**

- **&\$element**

*void function PDFdefaultConverter::convertGlobal(&\$element) [line [349](#)]*

**Function Parameters:**

- **&\$element**

*void function PDFdefaultConverter::convertInclude(&\$element) [line [494](#)]*

**Function Parameters:**

- **&\$element**

*void function PDFdefaultConverter::convertMethod(&\$element) [line [376](#)]*

**Function Parameters:**

- **&\$element**

*void function PDFdefaultConverter::convertPackagepage(&\$element) [line [638](#)]*

**Function Parameters:**

- **&\$element**

- **Deprecated** html package pages just don't work with PDF, use [phpDocumentor Tutorials](#)

*void function PDFdefaultConverter::convertPage(&\$element) [line [562](#)]*

**Function Parameters:**

- **&\$element**

*void function PDFdefaultConverter::convertParams(&\$element) [line [331](#)]*

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::convertTutorial(&\$element) [/line [645](#)]

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::convertVar(&\$element) [/line [416](#)]

**Function Parameters:**

- **&\$element**

*void* function PDFdefaultConverter::Convert\_RIC(\$name, \$contents) [/line [244](#)]

**Function Parameters:**

- *README|INSTALL|CHANGELOG* **\$name**
- *string* **\$contents** contents of the file

## Convert README/INSTALL/CHANGELOG file contents to output format

*void* function PDFdefaultConverter::generateFormattedClassTrees(\$package) [/line [665](#)]

**Function Parameters:**

- *string* **\$package** package to generate a class tree for

## returns a template-enabled array of class trees

- See \$roots, [PDFdefaultConverter::getRootTree\(\)](#)

*void* function PDFdefaultConverter::getCDATA(\$value) [/line [630](#)]

**Function Parameters:**

- **\$value**

*mixed function PDFdefaultConverter::getClassLink(\$expr, \$package, [\$file = false], [\$text = false]) [line 843]*

**Function Parameters:**

- *string \$expr* name of class
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- **\$text**

- See [Converter::getClassLink\(\)](#)

*mixed function PDFdefaultConverter::getConstLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], 5) [line 953]*

**Function Parameters:**

- *boolean 5* return just the URL, or enclose it in an html a tag
- *string \$expr* name of class constant
- *string \$class* class containing class constant
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getConstLink\(\)](#)

*mixed function PDFdefaultConverter::getDefineLink(\$expr, \$package, [\$file = false], [\$text = false]) [line 873]*

**Function Parameters:**

- *string \$expr* name of define
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated

- See [Converter::getDefineLink\(\)](#)

*void* function PDFdefaultConverter::getExampleLink(\$path, \$title) [line [140](#)]

**Function Parameters:**

- **\$path**
- **\$title**

*void* function PDFdefaultConverter::getFileSourceName(\$path, [\$anchor = "]) [line [152](#)]

**Function Parameters:**

- **\$path**
- **\$anchor**

*mixed* function PDFdefaultConverter::getFunctionLink(\$expr, \$package, [\$file = false], [\$text = false]) [line [858](#)]

**Function Parameters:**

- *string* **\$expr** name of function
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getFunctionLink\(\)](#)

*mixed* function PDFdefaultConverter::getGlobalLink(\$expr, \$package, [\$file = false], [\$text = false]) [line [888](#)]

**Function Parameters:**

- *string* **\$expr** name of global variable
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getGlobalLink\(\)](#)

*void* function PDFdefaultConverter::getGlobalValue(\$value) [*line 371*]

**Function Parameters:**

- **\$value**

*mixed* function PDFdefaultConverter::getMethodLink(\$expr, \$class, \$package, [\$file = false], [\$text = false])

[*line 919*]

**Function Parameters:**

- *string* **\$expr** name of method
- *string* **\$class** class containing method
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getMethodLink\(\)](#)

*mixed* function PDFdefaultConverter::getPageLink(\$expr, \$package, [\$path = false], [\$text = false]) [*line 903*]

**Function Parameters:**

- *string* **\$expr** name of procedural page
- *string* **\$package** package name
- *string* **\$path** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getPageLink\(\)](#)

*void* function PDFdefaultConverter::getPageName(&\$element) [line [619](#)]

**Function Parameters:**

- **&\$element**

*string* function PDFdefaultConverter::getRootTree(\$tree, \$package) [line [684](#)]

**Function Parameters:**

- **array \$tree** output from [getSortedClassTreeFromClass\(\)](#)
- **\$package**

**return formatted class tree for the Class Trees page**

- See Classes::\$definitechild, [PDFdefaultConverter::generateFormattedClassTrees\(\)](#)

*string* function PDFdefaultConverter::getSourceAnchor(\$sourcefile, \$anchor, [\$text = "], [\$link = false]) [line [167](#)]

**Function Parameters:**

- **string \$sourcefile** full path to source file
- **string \$anchor** name of anchor
- **string \$text** link text, if this is a link
- **boolean \$link** returns either a link or a destination based on this parameter

**Retrieve a Converter-specific anchor to a segment of a source code file parsed via a [@filesource](#) tag.**

*void* function PDFdefaultConverter::getSourceLink(\$path) [line [146](#)]

**Function Parameters:**

- **\$path**

*1* function PDFdefaultConverter::getState() [line [967](#)]

*void* function PDFdefaultConverter::getTutorialId(\$package, \$subpackage, \$tutorial, \$id) [line [625](#)]

**Function Parameters:**

- **\$package**
- **\$subpackage**
- **\$tutorial**
- **\$id**

*mixed* function PDFdefaultConverter::getVarLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], 5) [line 936]

**Function Parameters:**

- *boolean* **5** return just the URL, or enclose it in an html a tag
- *string* **\$expr** name of var
- *string* **\$class** class containing var
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated

- See [Converter::getVarLink\(\)](#)

*void* function PDFdefaultConverter::mystrnatcasecmp(\$a, \$b) [line 830]

**Function Parameters:**

- **\$a**
- **\$b**

*void* function PDFdefaultConverter::Output([\$title = 'Generated Documentation']) [line 751]

**Function Parameters:**

- **\$title**

calls [Cezpdf::ezOutput\(\)](#) and writes documentation.pdf to targetDir

*void* function PDFdefaultConverter::postProcess(\$text) [line 121]

**Function Parameters:**

- **\$text**

*string* function PDFdefaultConverter::returnLink(\$link, \$text) [[line 234](#)]

**Function Parameters:**

- *string* **\$link**
- *string* **\$text**

*void* function PDFdefaultConverter::returnSee(&\$element, [\$eltext = false]) [[line 184](#)]

**Function Parameters:**

- *abstractLink* &**\$element** a descendant of abstractlink should be passed, and never text
- *string* **\$eltext** text to display in the link

## Returns a bookmark using Cezpdf 009

*void* function PDFdefaultConverter::setTemplateDir(\$dir) [[line 960](#)]

**Function Parameters:**

- **\$dir**

*string* function PDFdefaultConverter::TranslateTag(\$name, \$attr, \$cdata, \$unconvertedcdata) [[line 602](#)]

**Function Parameters:**

- *string* **\$name** tag name
- *string* **\$attr** any attributes Format: array(name => value)
- *string* **\$cdata** the tag contents, if any
- *string* **\$unconvertedcdata** the tag contents, if any, unpost-processed

## Used to translate an XML DocBook tag from a tutorial by reading the options.ini file for the template.

*void* function PDFdefaultConverter::unmangle(\$notused, \$source) [[line 975](#)]

**Function Parameters:**

- **\$notused**
- **\$source**

- See [Converter::unmangle\(\)](#)

void function PDFdefaultConverter::writeExample(\$title, \$path, \$source) [[line 126](#)]

**Function Parameters:**

- **\$title**
- **\$path**
- **\$source**

void function PDFdefaultConverter::writeSource(\$path, \$value) [[line 106](#)]

**Function Parameters:**

- **\$path**
- **\$value**

## Class PDFParser

[[line 68](#)]

- **Package** Converters
- **Sub-Package** PDFdefault
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net) >
- **Version** \$Id: Parser.inc 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2000-2007 Kellin, Joshua Eichorn
- **Since** 1.2
- **Usedby** [phdopdf::ezText\(\)](#) - extracts all meta-tags and processes text for output

Constructor void function PDFParser::PDFParser() [[line 89](#)]

**Sets up the wordparser for this class**

*void* function PDFParser::getParserEventName(\$value) [line [549](#)]

**Function Parameters:**

- *integer* **\$value**

**Return the name of the parser event**

*bool* function PDFParser::parse(\$parse\_data, \$fontdir, &\$pdf, [\$debug = false]) [line [105](#)]

**Function Parameters:**

- *string* **\$parse\_data** text to parse for PDFParser XML tags
- *string* **\$fontdir** full path to the font directory
- [\*\*phpdocpdf\*\*](#) **&\$pdf**
- *boolean* **\$debug** determines whether output is saved in a variable or directly to the output

added

**Parse text for PDFParser XML tags, and add the text to the PDF file**

- **Static Variable Used** integer \$endrecur: used for recursion limiting if a handler for an event is not found

*void* function PDFParser::setupStates() [line [485](#)]

**setup the parser tokens, and the pushEvent/popEvent arrays**

- See [Publisher::\\$tokens](#), [Publisher::\\$pushEvent](#), [Publisher::\\$popEvent](#)

**Class phpdocpdf**  
[line [50](#)]

- **Package** Converters
- **Sub-Package** PDFdefault
- **Author** Wayne Munro, R&OS Ltd, <http://www.ros.co.nz/pdf>
- **Version** 009 (versioning is linked to class.pdf.php) released under a public domain licence.

**phpdocpdf::\$converter**

*mixed = [line 57]*

**phpdocpdf::\$font\_dir**

*mixed = false [line 55]*

**phpdocpdf::\$indents**

*mixed = array() [line 54]*

**phpdocpdf::\$indexContents**

*mixed = array() [line 53]*

**phpdocpdf::\$listType**

*mixed = 'ordered' [line 59]*

**phpdocpdf::\$reportContents**

*mixed = array() [line 52]*

**phpdocpdf::\$set\_pageNumbering**

*mixed = false [line 56]*

**phpdocpdf::\$\_colorStack**

*mixed = array() [line 60]*

**phpdocpdf::\$\_save**

*mixed = " [line 58]*

Constructor void function `phpdocpdf::phpdocpdf(&$pdfconverter, $fontdir, [$paper = 'a4'], [$orientation = 'portrait'])` [line 62]

**Function Parameters:**

- **&\$pdfconverter**
- **\$fontdir**
- **\$paper**
- **\$orientation**

*void* function `phpdocpdf::addMessage($message)` [[line 263](#)]

**Function Parameters:**

- `$message`

*void* function `phpdocpdf::bullet($Data)` [[line 322](#)]

**Function Parameters:**

- `$Data`

- **Author** Murray Shields

*void* function `phpdocpdf::dots($info)` [[line 175](#)]

**Function Parameters:**

- `$info`

*void* function `phpdocpdf::ezNewPage([$debug = false])` [[line 341](#)]

**Function Parameters:**

- `$debug`

*void* function `phpdocpdf::ezOutput([$debug = false], $template)` [[line 232](#)]

**Function Parameters:**

- `$debug`
- `$template`

*void* function `phpdocpdf::ezProcessText($text)` [[line 270](#)]

**Function Parameters:**

- `$text`

*void* function phpdocpdf::ezText(\$text, [\$size = 0], [\$options = array()], [\$test = 0]) [line [215](#)]

**Function Parameters:**

- **\$text**
- **\$size**
- **\$options**
- **\$test**

- **Uses [PDFParser](#)** - extracts all meta-tags and processes text for output

*void* function phpdocpdf::getColor() [line [72](#)]

**This really should be in the parent class**

*void* function phpdocpdf::getYPlusOffset(\$offset) [line [258](#)]

**Function Parameters:**

- **\$offset**

*void* function phpdocpdf::indent(\$info) [line [300](#)]

**Function Parameters:**

- **\$info**

*void* function phpdocpdf::index(\$info) [line [142](#)]

**Function Parameters:**

- **\$info**

*void* function phpdocpdf::IndexLetter(\$info) [line [152](#)]

**Function Parameters:**

- **\$info**

*void* function phpdocpdf::orderedBullet(\$info) [line [336](#)]

**Function Parameters:**

- **\$info**

*void* function phpdocpdf::rf(\$info) [line [132](#)]

**Function Parameters:**

- **\$info**

*void* function phpdocpdf::setColorArray(\$color) [line [77](#)]

**Function Parameters:**

- **\$color**

*void* function phpdocpdf::setHTMLColor(\$color) [line [105](#)]

**Function Parameters:**

- **\$color**

*void* function phpdocpdf::setupTOC() [line [225](#)]

*void* function phpdocpdf::textcolor(\$info) [line [111](#)]

**Function Parameters:**

- **\$info**

*void* function phpdocpdf::validHTMLColor(\$color) [line [100](#)]

**Function Parameters:**

- **\$color**

*void* function phpdocpdf::\_ezText(\$text, [\$size = 0], [\$options = array()], [\$test = 0]) [line [253](#)]

**Function Parameters:**

- **\$text**
- **\$size**
- **\$options**
- **\$test**

# XMLDocBookpearDoc2Converter.inc

Outputs documentation in XML DocBook format, in the version expected by  
pear.php.net's documentation team  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** Converters
- **Sub-Package** XMLDocBook
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net) >
- **Version** CVS: \$Id: XMLDocBookpearDoc2Converter.inc 234423 2007-04-24 21:32:15Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.2
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

# XMLDocBookConverter.inc

**XML output converter for DocBook.**

This Converter takes output from the [Parser](#) and converts it to DocBook output

- **Package** Converters
- **Sub-Package** XMLDocBook
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: XMLDocBookConverter.inc 234145 2007-04-19 20:20:57Z ashnazg \$
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#), [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- **Since** 1.0

## Class XMLDocBookConverter

*[line 43]*

**XML DocBook converter.**

This Converter takes output from the [Parser](#) and converts it to DocBook output

- **Package** Converters
- **Sub-Package** XMLDocBook
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: XMLDocBookConverter.inc 234145 2007-04-19 20:20:57Z ashnazg \$
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#), [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- **TODO** templates
- **TODO** indexes for other DocBook converters not based on peardoc
- **Since** 1.0

**XMLDocBookConverter:::\$base\_dir**

*string = [line 101]*

**target directory passed on the command-line.**

**\$targetDir** is malleable, always adding package/ and package/subpackage/ subdirectories onto it.

### **XMLDocBookConverter::\$category**

*string = [line 162]*

**peardoc2 Category**

### **XMLDocBookConverter::\$class**

*string = [line 82]*

**name of current class being converted**

### **XMLDocBookConverter::\$class\_data**

*Template = [line 113]*

**template for the class currently being processed**

### **XMLDocBookConverter::\$class\_dir**

*string = [line 107]*

**output directory for the current class being processed**

### **XMLDocBookConverter::\$current**

*array = [line 138]*

**contains all of the template procedural page element loop data needed for the current template**

### **XMLDocBookConverter::\$currentclass**

*array = [line 144]*

**contains all of the template class element loop data needed for the current template**

### **XMLDocBookConverter::\$function\_data**

*mixed = array() [line 155]*

### **XMLDocBookConverter::\$juststarted**

*boolean = false [line 132]*

### **controls formatting of parser informative output**

Converter prints: "Converting /path/to/file.php... Procedural Page Elements... Classes..." Since HTMLdefaultConverter outputs files while converting, it needs to send a \n to start a new line. However, if there is more than one class, output is messy, with multiple \n's just between class file output. This variable prevents that and is purely cosmetic

### **XMDocBookConverter::\$leftindex**

*array = array('classes' => true, 'pages' => true, 'functions' => false, 'defines' => false, 'globals' => false) [line 59]*

### **indexes of elements by package that need to be generated**

### **XMDocBookConverter::\$local**

*boolean = true [line 64]*

**whether a @see is going to be in the \$base\_dir, or in a package/subpackage subdirectory of \$base\_dir**

### **XMDocBookConverter::\$method\_data**

*mixed = array() [line 156]*

### **XMDocBookConverter::\$name**

*string = 'DocBook' [line 54]*

### **XMDocBookConverter::\$outputformat**

*string = 'XML' [line 52]*

### **XMDocBookConverter::\$package\_pages**

*array = array() [line 120]*

#### **array of converted package page names.**

Used to link to the package page in the left index

- **Var Format:** array(package => 1)

**XMLDocBookConverter::\$page**

*string = [line 70]*

**name of current page being converted**

**XMLDocBookConverter::\$page\_data**

*Template = [line 88]*

**template for the procedural page currently being processed**

**XMLDocBookConverter::\$page\_dir**

*string = [line 94]*

**output directory for the current procedural page being processed**

**XMLDocBookConverter::\$path**

*string = [line 76]*

**path of current page being converted**

**XMLDocBookConverter::\$sort\_page\_contents\_by\_type**

*boolean = true [line 50]*

**XMLDocBookConverter wants elements sorted by type as well as alphabetically**

- See [Converter::\\$sort\\_page\\_contents\\_by\\_type](#)

**XMLDocBookConverter::\$sourceloc**

*mixed = " [line 157]*

**XMLDocBookConverter::\$template\_options**

*array = array('usepear' => false) [line 153]*

**template options. Currently only 1 recognized option usepear**

usepear tells the getLink() function to return a package link to PEAR and

PEAR\_ERROR if possible, and to link directly to the fully-delimited link package#class.method or package#file.method in PEAR style, if possible, even if the package is not parsed. This will allow parsing of separate PEAR packages without parsing the entire thing at once!

Constructor void function XMLDocBookConverter::XMLDocBookConverter(&\$allp, &\$packp, &\$classes, &\$procpages, \$po, \$pp, \$qm, \$targetDir, \$templateDir, \$title) [line 167]

**Function Parameters:**

- **&\$allp**
- **&\$packp**
- **&\$classes**
- **&\$procpages**
- **\$po**
- **\$pp**
- **\$qm**
- **\$targetDir**
- **\$templateDir**
- **\$title**

sets [\\$base\\_dir](#) to [\\$targetDir](#)

- See [Converter::Converter\(\)](#)

void function XMLDocBookConverter::convertClass(&\$element) [line 811]

**Function Parameters:**

- [parserClass](#) **&\$element**

**Converts class for template output**

- See [Converter::getFormattedInheritedMethods\(\)](#), [Converter::getFormattedInheritedVars\(\)](#)
- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [Converter::generateChildClassList\(\)](#),  
[XMLDocBookConverter::generateFormattedClassTree\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function XMLDocBookConverter::convertDefine(&\$element) [*line 1021*]

**Function Parameters:**

- *parserDefine* &\$element

### Converts defines for template output

- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void* function XMLDocBookConverter::ConvertErrorLog() [*line 625*]

### Create errors.html template file output

This method takes all parsing errors and warnings and spits them out ordered by file and line number.

- **Global Variable Used** [ErrorTracker](#) \$phpDocumentor\_errors: We'll be using it's output facility

*void* function XMLDocBookConverter::convertFunction(&\$element) [*line 952*]

**Function Parameters:**

- *parserFunction* &\$element

### Converts function for template output

- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [parserFunction::getFunctionCall\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void function XMLDocBookConverter::convertGlobal(&\$element) [line 1040]*

**Function Parameters:**

- [parserGlobal](#) &\$element

### Converts global variables for template output

- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void function XMLDocBookConverter::convertInclude(&\$element) [line 1003]*

**Function Parameters:**

- [parserInclude](#) &\$element

### Converts include elements for template output

- See [XMLDocBookConverter::prepareDocBlock\(\)](#)

*void function XMLDocBookConverter::convertMethod(&\$element) [line 873]*

**Function Parameters:**

- [parserMethod](#) &\$element

### Converts method for template output

- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [parserMethod::getFunctionCall\(\)](#),

[Converter::getFormattedDescMethods\(\)](#), [Converter::getFormattedOverrides\(\)](#)

void function XMLDocBookConverter::convertPackagePage(&\$element) [line 712]

**Function Parameters:**

- [parserPackagePage](#) &\$element

**Converts package page and sets its package as used**

[\\$package\\_pages](#)

void function XMLDocBookConverter::convertPage(&\$element) [line 1062]

**Function Parameters:**

- [parserData](#) &\$element

**converts procedural pages for template output**

- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [Converter::getClassesOnPage\(\)](#)

void function XMLDocBookConverter::convertTutorial(&\$element) [line 736]

**Function Parameters:**

- &\$element

void function XMLDocBookConverter::convertVar(&\$element) [line 774]

**Function Parameters:**

- [parserVar](#) &\$element

**Converts class variables for template output.**

- See [XMLDocBookConverter::prepareDocBlock\(\)](#), [Converter::getFormattedOverrides\(\)](#)

void function XMLDocBookConverter::endClass() [line 217]

**Writes out the template file class data and unsets the template to save memory**

- See [Converter::endClass\(\)](#)
- See [registerCurrentClass\(\)](#)

void function XMLDocBookConverter::endPage() [line 249]

**Writes out the template file page data and unsets the template to save memory**

- See [Converter::endPage\(\)](#)
- See [registerCurrent\(\)](#)

void function XMLDocBookConverter::formatIndex() [line 360]

**HTMLDefaultConverter uses this function to format template index.html and packages.html**

This function generates the package list from [\\$all\\_packages](#), eliminating any packages that don't have any entries in their package index (no files at all, due to @ignore or other factors). Then it uses the default package name as the first package index to display. It sets the right pane to be either a blank file with instructions on making package-level docs, or the package-level docs for the default package.

- **Global Variable Used** string \$phpDocumentor\_DefaultPackageName: Used to set the starting package to display

void function XMLDocBookConverter::formatLeftIndex() [line 438]

**Generate indexes for li\_package.html and classtree output files**

This function generates the li\_package.html files from the template file left.html. It does this by iterating through each of the \$page\_elements, \$class\_elements and \$function\_elements arrays to retrieve the pre-sorted [abstractLink](#) descendants needed for index generation. Conversion of these links to text is done by [returnSee\(\)](#). The \$local parameter is set to false to ensure that paths are correct.

Then it uses [generateFormattedClassTrees\(\)](#) to create class trees from the template file classtrees.html. Output filename is classtrees\_packagename.html. This function also unsets \$elements and \$pkg\_elements to free up the considerable memory these two class vars use

- See [Converter::\\$page\\_elements](#), [Converter::\\$class\\_elements](#), [Converter::\\$function\\_elements](#)

void function XMLDocBookConverter::formatPkgIndex() [line 327]

### **HTMLdefaultConverter chooses to format both package indexes and the complete index here**

This function formats output for the elementindex.html and pkgelementindex.html template files. It then writes them to the target directory

- See [XMLDocBookConverter::generateElementIndex\(\)](#),  
[XMLDocBookConverter::generatePkgElementIndex\(\)](#)

void function XMLDocBookConverter::generateElementIndex() [line 1259]

### **Generate alphabetical index of all elements**

- See [Converter::\\$elements](#), [Converter::walk\(\)](#)

array function XMLDocBookConverter::generateFormattedClassTree(\$class) [line 1122]

#### **Function Parameters:**

- [parserClass \\$class](#) class variable

**returns an array containing the class inheritance tree from the root object to the class**

- [Uses parserClass::getParentClassTree\(\)](#)

*void* function XMLDocBookConverter::generateFormattedClassTrees(\$package) [line 1172]

**Function Parameters:**

- *string* **\$package** package to generate a class tree for

**returns a template-enabled array of class trees**

- [See \\$roots, HTMLConverter::getRootTree\(\)](#)

*void* function XMLDocBookConverter::generatePkgElementIndex(\$package) [line 1447]

**Function Parameters:**

- *string* **\$package** name of a package

**Generate alphabetical index of all elements by package and subpackage**

- [See Converter::\\$pkg\\_elements, Converter::walk\(\),  
XMLDocBookConverter::generatePkgElementIndexes\(\)](#)

*void* function XMLDocBookConverter::generatePkgElementIndexes() [line 1587]

- See [XMLDocBookConverter::generatePkgElementIndex\(\)](#)

*void* function XMLDocBookConverter::getCDATA(\$value) [line 703]

**Function Parameters:**

- **\$value**

*mixed* function XMLDocBookConverter::getClassLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true], [\$with\_a = true]) [line 1627]

**Function Parameters:**

- *string* **\$expr** name of class
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag
- **\$with\_a**

- See [Converter::getClassLink\(\)](#)

*mixed* function XMLDocBookConverter::getDefineLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true]) [line 1659]

**Function Parameters:**

- *string* **\$expr** name of define
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getDefineLink\(\)](#)

*mixed* function XMLDocBookConverter::getFunctionLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true]) [line 1643]

**Function Parameters:**

- *string* **\$expr** name of function
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getFunctionLink\(\)](#)

*mixed* function XMLDocBookConverter::getGlobalLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true]) [line 1675]

**Function Parameters:**

- *string* **\$expr** name of global variable
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getGlobalLink\(\)](#)

*string* function XMLDocBookConverter::getId(&\$el) [line 574]

**Function Parameters:**

- *mixed* **&\$el** descendant of parserElement or parserData/parserPage

## Get the id value needed to allow linking

- See [parserElement](#), [parserData](#), [parserPage](#)

*void* function XMLDocBookConverter::getLink(*\$expr*, [*\$package* = false], [*\$packages* = false]) [line 175]

**Function Parameters:**

- **\$expr**
- **\$package**
- **\$packages**

### do that stuff in **\$template\_options**

*mixed* function XMLDocBookConverter::getMethodLink(*\$expr*, *\$class*, *\$package*, [*\$file* = false], [*\$text* = false], [*\$local* = true]) [line 1708]

**Function Parameters:**

- *string* **\$expr** name of method
- *string* **\$class** class containing method
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getMethodLink\(\)](#)

*mixed* function XMLDocBookConverter::getPageLink(*\$expr*, *\$package*, [*\$path* = false], [*\$text* = false], [*\$local* = true]) [line 1691]

**Function Parameters:**

- *string* **\$expr** name of procedural page
- *string* **\$package** package name
- *string* **\$path** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getPageLink\(\)](#)

*void* function XMLDocBookConverter::getPageName(&\$element) [line 1109]

**Function Parameters:**

- **&\$element**

*string* function XMLDocBookConverter::getRootTree(\$tree, \$package) [line 1191]

**Function Parameters:**

- **array \$tree** output from [getSortedClassTreeFromClass\(\)](#)
- **\$package**

**return formatted class tree for the Class Trees page**

- See Classes::\$definitechild, [XMLDocBookConverter::generateFormattedClassTrees\(\)](#)

*void* function XMLDocBookConverter::getTutorialId(\$package, \$subpackage, \$tutorial, \$id) [line 696]

**Function Parameters:**

- **\$package**
- **\$subpackage**
- **\$tutorial**
- **\$id**

*mixed* function XMLDocBookConverter::getVarLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], [\$local = true]) [line 1725]

**Function Parameters:**

- **string \$expr** name of var
- **string \$class** class containing var
- **string \$package** package name

- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated
- *boolean \$local* return just the URL, or enclose it in an html a tag

- See [Converter::getVarLink\(\)](#)

*void* function XMLDocBookConverter::makeLeft() [line 282]  
*void* function XMLDocBookConverter::Output() [line 1782]

**This function is not used by HTMLdefaultConverter, but is required by Converter**

*void* function XMLDocBookConverter::postProcess(\$text) [line 686]

**Function Parameters:**

- **\$text**

*void* function XMLDocBookConverter::prepareDocBlock(&\$element, [\$nopackage = true]) [line 691]

**Function Parameters:**

- **&\$element**
- **\$nopackage**

*int* function XMLDocBookConverter::rcNatCmp(\$a, \$b) [line 1739]

**Function Parameters:**

- *mixed \$a*
- *mixed \$b*

**does a nat case sort on the specified second level value of the array**

*int* function XMLDocBookConverter::rcNatCmp1(\$a, \$b) [line 1755]

**Function Parameters:**

- *mixed \$a*
- *mixed \$b*

**does a nat case sort on the specified second level value of the array.**  
this one puts constructors first

*string* function XMLDocBookConverter::returnLink(\$link, \$text) [line 277]

**Function Parameters:**

- *string* **\$link**
- *string* **\$text**

*void* function XMLDocBookConverter::returnSee(&\$element, [\$eltext = false], [\$local = true], [\$with\_a = true]) [line 476]

**Function Parameters:**

- **[abstractLink](#)** &\$element a descendant of abstractlink should be passed, and never text
- *string* **\$eltext** text to display in the link
- *boolean* **\$local** this parameter is not used, and is deprecated
- *boolean* **\$with\_a** determines whether the returned text is enclosed in an <a> tag

**This function takes a**[abstractLink](#)**descendant and returns an html link**

*void* function XMLDocBookConverter::setTargetDir(\$dir) [line 1387]

**Function Parameters:**

- **\$dir**

**calls the converter setTargetDir, and then copies any template images and the stylesheet if they haven't been copied**

- See [Converter::setTargetDir\(\)](#)

*void* function XMLDocBookConverter::setTemplateDir(\$dir) [line 1377]

**Function Parameters:**

- **\$dir**

*void* function XMLDocBookConverter::SmartyInit(&\$templ) [*line 205*]

**Function Parameters:**

- **&\$templ**

*void* function XMLDocBookConverter::type\_adjust(\$typename) [*line 186*]

**Function Parameters:**

- **\$typename**

*void* function XMLDocBookConverter::unmangle(\$s, \$sourcecode) [*line 180*]

**Function Parameters:**

- **\$s**
- **\$sourcecode**

*void* function XMLDocBookConverter::writeNewPPage(\$key) [*line 406*]

**Function Parameters:**

- **\$key**

## Class XMLDocBookpeardecor2Converter

[*line 58*]

### XML DocBook converter.

This Converter takes output from the [Parser](#) and converts it to DocBook output for PEAR documentation.

This Converter differs from the parent DocBook Converter in that it does not recognize the possibility of procedural pages or of functions! All functions must be defined as static methods for namespace purposes. In addition, all constants and global variables for a package are grouped together as per peardecor2 requirements. Include statements are

not documented. If you want to document a normal project, don't use the peardoc2 converter, use the DocBook converter.

- **Package** Converters
- **Sub-Package** XMLDocBook
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: XMLDocBookpeardecor2Converter.inc 234423 2007-04-24 21:32:15Z ashnazg \$
- **Since** 1.2

### **XMLDocBookpeardecor2Converter::\$base\_dir**

*string = [line [134](#)]*

**target directory passed on the command-line.**

**\$targetDir** is malleable, always adding package/ and package/subpackage/ subdirectories onto it.

### **XMLDocBookpeardecor2Converter::\$category**

*string = [line [207](#)]*

**peardecor2 Category**

### **XMLDocBookpeardecor2Converter::\$class**

*string = [line [103](#)]*

**name of current class being converted**

### **XMLDocBookpeardecor2Converter::\$class\_data**

*Template = [line [146](#)]*

**template for the class currently being processed**

### **XMLDocBookpeardecor2Converter::\$class\_dir**

*string = [line [140](#)]*

**output directory for the current class being processed**

**XMLDocBookpearDoc2Converter::\$current**

*array = [line 176]*

**contains all of the template procedural page element loop data needed for the current template**

**XMLDocBookpearDoc2Converter::\$currentclass**

*array = [line 182]*

**contains all of the template class element loop data needed for the current template**

**XMLDocBookpearDoc2Converter::\$function\_data**

*mixed = array() [line 198]*

**XMLDocBookpearDoc2Converter::\$juststarted**

*boolean = false [line 170]*

**controls formatting of parser informative output**

Converter prints: "Converting /path/to/file.php... Procedural Page Elements... Classes..." Since HTMLDefaultConverter outputs files while converting, it needs to send a \n to start a new line. However, if there is more than one class, output is messy, with multiple \n's just between class file output. This variable prevents that and is purely cosmetic

**XMLDocBookpearDoc2Converter::\$leftindex**

*array = array('classes' => true, 'pages' => false, 'functions' => false, 'defines' => true, 'globals' => true) [line 80]*

**indexes of elements by package that need to be generated**

**XMLDocBookpearDoc2Converter::\$local**

*boolean = true [line 85]*

**whether a @see is going to be in the \$base\_dir, or in a package/subpackage subdirectory of \$base\_dir**

**XMLDocBookpearDoc2Converter::\$method\_data**

*mixed = array() [line 199]*

**XMLDocBookpearDoc2Converter::\$name**

*string = 'DocBook/peardoc2' [line 75]*

#### **XMLDocBookpeardoc2Converter::\$outputformat**

*string = 'XML' [line 73]*

#### **XMLDocBookpeardoc2Converter::\$packagexml**

*Smarty = [line 159]*

**Contents of the packagename.xml file are stored in this template variable**

#### **XMLDocBookpeardoc2Converter::\$package\_pages**

*array = array() [line 153]*

**array of converted package page names.**

Used to link to the package page in the left index

- **Var Format:** array(package => 1)

#### **XMLDocBookpeardoc2Converter::\$page**

*string = [line 91]*

**name of current page being converted**

#### **XMLDocBookpeardoc2Converter::\$page\_data**

*Template = [line 109]*

**template for the procedural page currently being processed**

#### **XMLDocBookpeardoc2Converter::\$page\_dir**

*string = [line 115]*

**output directory for the current procedural page being processed**

#### **XMLDocBookpeardoc2Converter::\$path**

*string = [line 97]*

## path of current page being converted

**XMDocBookpeardoc2Converter::\$processSpecialRoots**

*boolean = true [line [65](#)]*

### This converter knows about the new root tree processing

In order to fix PEAR Bug #6389

**XMDocBookpeardoc2Converter::\$sort\_absolutely\_everything**

*mixed = true [line [187](#)]*

### Pass elements by package, simplifies generation of package.xml/category.xml

**XMDocBookpeardoc2Converter::\$sort\_page\_contents\_by\_type**

*boolean = true [line [71](#)]*

### XMDocBookConverter wants elements sorted by type as well as alphabetically

- See [Converter::\\$sort\\_page\\_contents\\_by\\_type](#)

**XMDocBookpeardoc2Converter::\$sourceloc**

*mixed = " [line [202](#)]*

**XMDocBookpeardoc2Converter::\$template\_options**

*array = array('usepear' => false) [line [196](#)]*

### template options. Currently only 1 recognized option usepear

usepear tells the getLink() function to return a package link to PEAR and PEAR\_ERROR if possible, and to link directly to the fully-delimited link package#class.method or package#file.method in PEAR style, if possible, even if the package is not parsed. This will allow parsing of separate PEAR packages without parsing the entire thing at once!

**XMDocBookpeardoc2Converter::\$\_peardoc2\_constants**

*array = false* [line [121](#)]

## Constants, used for constants.tpl

### XMLDocBookpeardoc2Converter::\$\_peardoc2\_globals

*array = false* [line [127](#)]

## Global Variables, used for globals.tpl

### XMLDocBookpeardoc2Converter::\$\_write\_constants\_xml

*mixed = array()* [line [200](#)]

### XMLDocBookpeardoc2Converter::\$\_write\_globals\_xml

*mixed = array()* [line [201](#)]

Constructor *void* function XMLDocBookpeardoc2Converter::XMLDocBookpeardoc2Converter(&\$allp, &\$packp, &\$classes, &\$procpages, \$po, \$pp, \$qm, \$targetDir, \$templateDir, \$title) [line [220](#)]

#### Function Parameters:

- **&\$allp**
- **&\$packp**
- **&\$classes**
- **&\$procpages**
- **\$po**
- **\$pp**
- **\$qm**
- **\$targetDir**
- **\$templateDir**
- **\$title**

## sets \$base\_dir to \$targetDir

- See [Converter::Converter\(\)](#)

*void* function XMLDocBookpeardoc2Converter::addSummaryToPackageXml(\$template\_output) [line [345](#)]

#### Function Parameters:

- **\$template\_output**

*void* function XMLDocBookpearDoc2Converter::Br(\$input) [line [681](#)]

**Function Parameters:**

- **\$input**

*void* function XMLDocBookpearDoc2Converter::convertClass(&\$element) [line [793](#)]

**Function Parameters:**

- [\*parserClass\*](#) &\$element

## Converts class for template output

- Uses [\*XMLDocBookpearDoc2Converter::flushPackageXml\(\)\*](#) - creates packagename.xml file when all classes in a package have been converted

*void* function XMLDocBookpearDoc2Converter::convertDefine(&\$element) [line [1009](#)]

**Function Parameters:**

- [\*parserDefine\*](#) &\$element

## Converts defines for template output

- See [\*XMLDocBookpearDoc2Converter::prepareDocBlock\(\)\*](#), [\*Converter::getFormattedConflicts\(\)\*](#)

*void* function XMLDocBookpearDoc2Converter::ConvertErrorLog() [line [565](#)]

### Create errors.html template file output

This method takes all parsing errors and warnings and spits them out ordered by file and line number.

- **Global Variable Used** [ErrorTracker](#) \$phpDocumentor\_errors: We'll be using it's output facility

*void function XMLDocBookpeardoc2Converter::convertFunction(&\$element) [line 937]*

**Function Parameters:**

- [parserFunction](#) &\$element

## Converts function for template output - does nothing in peardoc2!

*void function XMLDocBookpeardoc2Converter::convertGlobal(&\$element) [line 1049]*

**Function Parameters:**

- [parserGlobal](#) &\$element

## Converts global variables for template output

- See [XMLDocBookpeardoc2Converter::prepareDocBlock\(\)](#), [Converter::getFormattedConflicts\(\)](#)

*void function XMLDocBookpeardoc2Converter::convertInclude(&\$element) [line 989]*

**Function Parameters:**

- [parserInclude](#) &\$element

## Converts include elements for template output

Completely ignored by this converter

*void function XMLDocBookpeardoc2Converter::convertMethod(&\$element) [line 861]*

**Function Parameters:**

- [\*parserMethod\*](#) &\$element

## Converts method for template output

- See [XMLDocBookpeardecor2Converter::prepareDocBlock\(\)](#), [parserMethod::getFunctionCall\(\)](#), [Converter::getFormattedDescMethods\(\)](#), [Converter::getFormattedOverrides\(\)](#)

void function XMLDocBookpeardecor2Converter::convertPackagePage(&\$element) [[line 708](#)]

### **Function Parameters:**

- [\*parserPackagePage\*](#) &\$element

## Does nothing - use tutorials for DocBook

void function XMLDocBookpeardecor2Converter::convertPage(&\$element) [[line 1099](#)]

### **Function Parameters:**

- [\*parserData\*](#) &\$element

## converts procedural pages for template output

- See [XMLDocBookpeardecor2Converter::prepareDocBlock\(\)](#), [Converter::getClassesOnPage\(\)](#)

void function XMLDocBookpeardecor2Converter::convertTutorial(&\$element) [[line 716](#)]

### **Function Parameters:**

- [\*parserTutorial\*](#) &\$element

## Convert tutorials for output

void function XMLDocBookpearDoc2Converter::convertVar(&\$element) [/line [755](#)]

**Function Parameters:**

- [parserVar](#) &\$element

### Does nothing in this converter

void function XMLDocBookpearDoc2Converter::endClass() [/line [322](#)]

**Writes out the template file [\\$class](#) data and unsets the template to save memory**

- See registerCurrentClass()
- See [Converter::endClass\(\)](#)
- **TODO** move class summary into an array to be written out at the end of parsing each package

string function XMLDocBookpearDoc2Converter::exampleProgramExample(\$example, [\$tutorial = false], [\$inlinesourceparse = null], [\$class = null], [\$linenum = null], [\$filesourcepath = null]) [/line [276](#)]

**Function Parameters:**

- **string \$example**
- **boolean \$tutorial** true if this is to highlight a tutorial <programlisting>
- **\$inlinesourceparse**
- **\$class**
- **\$linenum**
- **\$filesourcepath**

### Used to convert the {@example} inline tag in a docblock.

By default, this just wraps ProgramExample

- See XMLDocBookpearDoc2Converter::exampleProgramExample

*void* function XMLDocBookpeardoc2Converter::flushPackageXml(*\$element*) [line [353](#)]

**Function Parameters:**

- *parserClass*/*false* **\$element** is false if this is the end of all conversion
  - **Used by** [XMLDocBookpeardoc2Converter::convertClass\(\)](#) - creates packagename.xml file when all classes in a package have been converted

*void* function XMLDocBookpeardoc2Converter::formatIndex() [line [419](#)]

**Does nothing**

*void* function XMLDocBookpeardoc2Converter::formatLeftIndex() [line [440](#)]

**Creates package/lang/categoryname/packagename.xml for each package**

*void* function XMLDocBookpeardoc2Converter::formatPkgIndex() [line [412](#)]

**Does nothing**

*void* function XMLDocBookpeardoc2Converter::generateChildClassList(*\$class*) [line [1165](#)]

**Function Parameters:**

- *parserClass* **\$class** class variable

**returns a list of child classes**

- **Uses** [parserClass::getChildClassList\(\)](#)

*void* function XMLDocBookpeardoc2Converter::generateElementIndex() [line [1378](#)]

**does nothing**

array function XMLDocBookpeardoc2Converter::generateFormattedClassTree(\$class) [line [1124](#)]

**Function Parameters:**

- [\*parserClass \\$class\*](#) class variable

**returns an array containing the class inheritance tree from the root object to the class**

- [\*\*Uses parserClass::getParentClassTree\(\)\*\*](#)

void function XMLDocBookpeardoc2Converter::generateFormattedClassTrees(\$package) [line [1201](#)]

**Function Parameters:**

- [\*string \\$package\*](#) package to generate a class tree for

**returns a template-enabled array of class trees**

- [\*\*See \\$roots, HTMLConverter::getRootTree\(\)\*\*](#)

void function XMLDocBookpeardoc2Converter::generateFormattedInterfaceTrees(\$package) [line [1255](#)]

**Function Parameters:**

- [\*string \\$package\*](#) package to generate a class tree for

**returns a template-enabled array of interface inheritance trees**

- [\*\*See \\$roots, HTMLConverter::getRootTree\(\)\*\*](#)

*void* function XMLDocBookpeardoc2Converter::generatePkgElementIndex(*\$package*) [line [1394](#)]  
**Function Parameters:**

- *string \$package* name of a package

## Generate alphabetical index of all elements by package and subpackage

- See [Converter::\\$pkg\\_elements](#), [Converter::walk\(\)](#),  
[XMLDocBookpeardoc2Converter::generatePkgElementIndexes\(\)](#)

*void* function XMLDocBookpeardoc2Converter::generatePkgElementIndexes() [line [1402](#)]

- See [XMLDocBookpeardoc2Converter::generatePkgElementIndex\(\)](#)

*void* function XMLDocBookpeardoc2Converter::getCDATA(*\$value*) [line [686](#)]

**Function Parameters:**

- **\$value**

*mixed* function XMLDocBookpeardoc2Converter::getClassLink(*\$expr*, *\$package*, [*\$file* = false], [*\$text* = false], [*\$local* = true], [*\$with\_a* = true]) [line [1415](#)]

**Function Parameters:**

- *string \$expr* name of class
- *string \$package* package name
- *string \$file* full path to look in (used in index generation)
- *boolean \$text* deprecated
- *boolean \$local* return just the URL, or enclose it in an html a tag
- *\$with\_a*

- See [Converter::getClassLink\(\)](#)

*mixed* function XMLDocBookpeardecor2Converter::getDefineLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true]) [line [1447](#)]

**Function Parameters:**

- *string* **\$expr** name of define
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getDefineLink\(\)](#)

*void* function XMLDocBookpeardecor2Converter::getExampleLink(\$unused, \$title) [line [290](#)]

**Function Parameters:**

- **\$unused**
- **\$title**

*mixed* function XMLDocBookpeardecor2Converter::getFunctionLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true]) [line [1431](#)]

**Function Parameters:**

- *string* **\$expr** name of function
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getFunctionLink\(\)](#)

*mixed* function XMLDocBookpeardoc2Converter::getGlobalLink(\$expr, \$package, [\$file = false], [\$text = false], [\$local = true]) [line [1463](#)]

**Function Parameters:**

- *string* **\$expr** name of global variable
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getGlobalLink\(\)](#)

*string* function XMLDocBookpeardoc2Converter::getId(&\$el, [\$returnpackage = false]) [line [513](#)]

**Function Parameters:**

- *mixed* **&\$el** descendant of parserElement or parserData/parserPage
- *boolean* **\$returnpackage** true to return the id for the package page

## Get the id value needed to allow linking

- See [parserElement](#), [parserData](#), [parserPage](#)

*void* function XMLDocBookpeardoc2Converter::getLink(\$expr, [\$package = false], [\$packages = false]) [line [237](#)]

**Function Parameters:**

- **\$expr**
- **\$package**
- **\$packages**

## do that stuff in \$template\_options

*mixed* function XMLDocBookpeardoc2Converter::getMethodLink(\$expr, \$class, \$package, [\$file = false], [\$text = false], [\$local = true]) [line [1496](#)]

**Function Parameters:**

- *string* **\$expr** name of method
- *string* **\$class** class containing method
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getMethodLink\(\)](#)

*mixed* function XMLDocBookpeardoc2Converter::getPageLink(\$expr, \$package, [\$path = false], [\$text = false], [\$local = true]) [line [1479](#)]

**Function Parameters:**

- *string* **\$expr** name of procedural page
- *string* **\$package** package name
- *string* **\$path** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getPageLink\(\)](#)

*void* function XMLDocBookpeardoc2Converter::getPageName(&\$element) [line [1111](#)]

**Function Parameters:**

- **&\$element**

*string* function XMLDocBookpeardoc2Converter::getRootTree(\$tree, \$package, [\$noparent = false], [\$nouknownparent]) [line [1312](#)]

**Function Parameters:**

- *array* **\$tree** output from [getSortedClassTreeFromClass\(\)](#)
- *string* **\$package** package
- *boolean* **\$nouknownparent** if true, an object's parent will not be checked
- **\$nparent**

**return formatted class tree for the Class Trees page**

- See Classes::**\$definitechild**,  
[XMLDocBookpeardoc2Converter::generateFormattedClassTrees\(\)](#)

*string* function XMLDocBookpeardoc2Converter::getSourceAnchor(*\$sourcefile*, *\$anchor*, [*\$text* = "], [*\$link* = false]) [line [676](#)]

**Function Parameters:**

- *string* **\$sourcefile** full path to source file
- *string* **\$anchor** name of anchor
- *string* **\$text** link text, if this is a link
- *boolean* **\$link** returns either a link or a destination based on this parameter

**Retrieve a Converter-specific anchor to a segment of a source code file parsed via**

**a [@filesource](#) tag.**

NOTE: unused

*void* function XMLDocBookpeardoc2Converter::getTutorialId(*\$package*, *\$subpackage*, *\$tutorial*, *\$id*, *\$category*) [line [656](#)]

**Function Parameters:**

- **\$package**
- **\$subpackage**
- **\$tutorial**
- **\$id**
- **\$category**

*mixed* function XMLDocBookpeardoc2Converter::getVarLink(*\$expr*, *\$class*, *\$package*, [*\$file* = false], [*\$text* = false], [*\$local* = true]) [line [1513](#)]

**Function Parameters:**

- *string* **\$expr** name of var
- *string* **\$class** class containing var
- *string* **\$package** package name
- *string* **\$file** full path to look in (used in index generation)
- *boolean* **\$text** deprecated
- *boolean* **\$local** return just the URL, or enclose it in an html a tag

- See [Converter::getVarLink\(\)](#)

*void* function XMLDocBookpeardoc2Converter::makeLeft() [*line 405*]

*void* function XMLDocBookpeardoc2Converter::Output() [*line 1575*]

**Generate the constants.xml, packagename.xml, and globals.xml files**

*void* function XMLDocBookpeardoc2Converter::postProcess(\$text) [*line 626*]

**Function Parameters:**

- **\$text**

*void* function XMLDocBookpeardoc2Converter::prepareDocBlock(&\$element, [\$nopackage = true]) [*line 631*]

**Function Parameters:**

- **&\$element**
- **\$nopackage**

*void* function XMLDocBookpeardoc2Converter::ProgramExample(\$listing, [\$tutorial = false],  
[\$inlinesourceparse = null], [\$class = null], [\$linenum = null], [\$filesourcepath = null], [\$origsource = null]) [*line 691*]

**Function Parameters:**

- **\$listing**
- **\$tutorial**
- **\$inlinesourceparse**
- **\$class**
- **\$linenum**
- **\$filesourcepath**
- **\$origsource**

*int* function XMLDocBookpearDoc2Converter::rcNatCmp(\$a, \$b) [/line [1527](#)]

**Function Parameters:**

- *mixed \$a*
- *mixed \$b*

**does a nat case sort on the specified second level value of the array**

*int* function XMLDocBookpearDoc2Converter::rcNatCmp1(\$a, \$b) [/line [1543](#)]

**Function Parameters:**

- *mixed \$a*
- *mixed \$b*

**does a nat case sort on the specified second level value of the array.**

this one puts constructors first

*string* function XMLDocBookpearDoc2Converter::returnLink(\$link, \$text) [/line [400](#)]

**Function Parameters:**

- *string \$link*
- *string \$text*

*void* function XMLDocBookpearDoc2Converter::returnSee(&\$element, [\$eltext = false], [\$local = true], [\$with\_a = true]) [/line [453](#)]

**Function Parameters:**

- *abstractLink &\$element* a descendant of abstractlink should be passed, and never text
- *string \$eltext* text to display in the link
- *boolean \$local* this parameter is not used, and is deprecated
- *boolean \$with\_a* determines whether the returned text is enclosed in an <link> tag

**This function takes a [abstractLink](#) descendant and returns an html link**

*void* function XMLDocBookpearDoc2Converter::setTemplateDir(\$dir) [/line [1382](#)]

**Function Parameters:**

- **\$dir**

*void* function XMLDocBookpeardecor2Converter::type\_adjust(\$typename) [[line 296](#)]

**Function Parameters:**

- **\$typename**

*void* function XMLDocBookpeardecor2Converter::unmangle(\$s, \$sourcecode) [[line 242](#)]

**Function Parameters:**

- **\$s**
- **\$sourcecode**

*void* function XMLDocBookpeardecor2Converter::wordwrap(\$string) [[line 1567](#)]

**Function Parameters:**

- **\$string**

*void* function XMLDocBookpeardecor2Converter::writeExample(\$title, \$path, \$source) [[line 285](#)]

**Function Parameters:**

- **\$title**
- **\$path**
- **\$source**

*void* function XMLDocBookpeardecor2Converter::writeFile(\$file, \$data, [\$binary = false]) [[line 254](#)]

**Function Parameters:**

- *string* **\$file** filename
- *string* **\$data** file contents
- *boolean* **\$binary** true if the data is binary and not text

**Writes a file to target dir, beautify any .xml files first**

*void* function XMLDocBookpeardoc2Converter::writeNewPPage(\$key) [line [426](#)]

**Function Parameters:**

- **\$key**

**Does nothing**

*void* function XMLDocBookpeardoc2Converter::writeSource() [line [433](#)]

**Does nothing**



# Package Cpdf Procedural Elements

## class.ezpdf.php

- **Package Cpdf**

```
include_once 'phpDocumentor/Converters/PDF/default/class.pdf.php[line 8]
```

### Cpdf class

# Package Cpdf Classes

## Class Cezpdf

*[line 23]*

**This class will take the basic interaction facilities of the Cpdf class and make more useful functions so that the user does not have to know all the ins and outs of pdf presentation to produce something pretty.**

**IMPORTANT NOTE** there is no warranty, implied or otherwise with this software.

- **Package** Cpdf
- **Author** Wayne Munro, R&OS Ltd, <http://www.ros.co.nz/pdf>
- **Version** 009 (versioning is linked to class.pdf.php) released under a public domain licence.

### Cezpdf::\$ez

*mixed = array('fontSize'=>10) [line 39]*

### Cezpdf::\$ezPageCount

*mixed = 0 [line 42]*

### Cezpdf::\$ezPages

*mixed = array() [line 41]*

### Cezpdf::\$y

*mixed = [line 40]*

Constructor void function Cezpdf::Cezpdf([\$paper = 'a4'], [\$orientation = 'portrait']) *[line 46]*

**Function Parameters:**

- **\$paper**
- **\$orientation**

*void function Cezpdf::alink(\$info, [\$internal = 0]) [line 1491]*

**Function Parameters:**

- **\$info**
- **\$internal**

*void function Cezpdf::execTemplate(\$id, [\$data = array()], [\$options = array()]) [line 1478]*

**Function Parameters:**

- **\$id**
- **\$data**
- **\$options**

*void function Cezpdf::ezColumnsStart([\$options = array()]) [line 162]*

**Function Parameters:**

- **\$options**

*void function Cezpdf::ezColumnsStop() [line 195]*

*void function Cezpdf::ezGetCurrentPageNumber() [line 284]*

*void function Cezpdf::ezImage(\$image, [\$pad = 5], [\$width = 0], [\$resize = 'full'], [\$just = 'center'], [\$border = '']) [line 1319]*

**Function Parameters:**

- **\$image**
- **\$pad**
- **\$width**
- **\$resize**
- **\$just**
- **\$border**

*void function Cezpdf::ezInsertMode([\$status = 1], [\$pageNum = 1], [\$pos = 'before']) [line 205]*

**Function Parameters:**

- **\$status**
- **\$pageNum**
- **\$pos**

*void function Cezpdf::ezNewPage() [line 222]  
void function Cezpdf::ezOutput([\$options = 0]) [line 479]*

**Function Parameters:**

- **\$options**

*void function Cezpdf::ezProcessText(\$text) [line 1213]*

**Function Parameters:**

- **\$text**

*void function Cezpdf::ezPRVTaddPageNumbers() [line 395]  
void function Cezpdf::ezPRVTcleanUp() [line 466]  
void function Cezpdf::ezPrvtGetTextWidth(\$size, \$text) [line 609]*

**Function Parameters:**

- **\$size**
- **\$text**

*void function Cezpdf::ezPRVTpageNumberSearch(\$lbl, &\$tmp) [line 378]*

**Function Parameters:**

- **\$lbl**
- **&\$tmp**

*void function Cezpdf::ezPrvtTableColumnHeadings(\$cols, \$pos, \$maxWidth, \$height, \$decender, \$gap, \$size, &\$y, [\$optionsAll = array()]) [line 544]*

**Function Parameters:**

- **\$cols**
- **\$pos**
- **\$maxWidth**
- **\$height**
- **\$decender**
- **\$gap**
- **\$size**
- **&\$y**
- **\$optionsAll**

*void function Cezpdf::ezPrvtTableDrawLines(\$pos, \$gap, \$x0, \$x1, \$y0, \$y1, \$y2, \$col, \$inner, \$outer, [\$opt = 1]) [line 515]*

**Function Parameters:**

- **\$pos**
- **\$gap**
- **\$x0**
- **\$x1**
- **\$y0**
- **\$y1**
- **\$y2**
- **\$col**
- **\$inner**
- **\$outer**
- **\$opt**

*void function Cezpdf::ezSetCmMargins(\$top, \$bottom, \$left, \$right) [line 152]*

**Function Parameters:**

- **\$top**
- **\$bottom**
- **\$left**
- **\$right**

*void function Cezpdf::ezSetDy(\$dy, [\$mod = "]) [line 497]*

**Function Parameters:**

- **\$dy**
- **\$mod**

*void function Cezpdf::ezSetMargins(\$top, \$bottom, \$left, \$right) [line 264]*

**Function Parameters:**

- **\$top**
- **\$bottom**
- **\$left**
- **\$right**

*void function Cezpdf::ezSetY(\$y) [line 486]*

**Function Parameters:**

- **\$y**

*void function Cezpdf::ezStartPageNumbers(\$x, \$y, \$size, [\$pos = 'left'], [\$pattern = '{PAGENUM} of {TOTALPAGENUM}'], [\$num = '']) [line 291]*

**Function Parameters:**

- **\$x**
- **\$y**
- **\$size**
- **\$pos**
- **\$pattern**
- **\$num**

*void function Cezpdf::ezStopPageNumbers([\$stopTotal = 0], [\$next = 0], [\$i = 0]) [line 349]*

**Function Parameters:**

- **\$stopTotal**
- **\$next**
- **\$i**

*void function Cezpdf::ezStream([\$options = '']) [line 472]*

**Function Parameters:**

- **\$options**

*void function Cezpdf::ezTable(&\$data, [\$cols = ''], [\$title = ''], [\$options = '']) [line 625]*

**Function Parameters:**

- **&\$data**
- **\$cols**
- **\$title**
- **\$options**

*void function Cezpdf::ezText(\$text, [\$size = 0], [\$options = array()], [\$test = 0]) [line 1223]*

**Function Parameters:**

- **\$text**
- **\$size**
- **\$options**

- **\$test**

*void function Cezpdf::ezWhatPageNumber(\$pageNum, [\$i = 0]) [line 317]*

**Function Parameters:**

- **\$pageNum**
- **\$i**

*void function Cezpdf::ilink(\$info) [line 1487]*

**Function Parameters:**

- **\$info**

*void function Cezpdf::loadTemplate(\$templateFile) [line 1441]*

**Function Parameters:**

- **\$templateFile**

*void function Cezpdf::uline(\$info) [line 1534]*

**Function Parameters:**

- **\$info**

## Class Cpdf

*[line 26]*

### Cpdf

A PHP class to provide the basic functionality to create a pdf document without any requirement for additional modules.

Note that they companion class CezPdf can be used to extend this class and dramatically simplify the creation of documents.

**IMPORTANT NOTE** there is no warranty, implied or otherwise with this software.

- **Package** Cpdf
- **Author** Wayne Munro < [pdf@ros.co.nz](mailto:pdf@ros.co.nz)>
- **Version** 009
- **Link** <http://www.ros.co.nz/pdf>

#### Cpdf::\$addLooseObjects

*mixed = array() [line 114]*

**array contains infomation about how the loose objects are to be added to the document**

#### Cpdf::\$arc4

*mixed = " [line 159]*

**the encryption array for the document encryption is stored here**

#### Cpdf::\$arc4\_objnum

*mixed = 0 [line 163]*

**the object Id of the encryption information**

#### Cpdf::\$callback

*mixed = array() [line 179]*

**array which forms a stack to keep track of nested callback functions**

#### Cpdf::\$catalogId

*mixed = [line 39]*

**the objectId (number within the objects array) of the document catalog**

#### Cpdf::\$checkpoint

*mixed = " [line 194]*

**store the stack for the transaction commands, each item in here is a record of the values of all the variables within the class, so that the user can rollback at will (from each 'start' command) note that this includes the objects array, so these can be large.**

**Cpdf::\$currentBaseFont**

*mixed = " [line 52]*

**the current base font**

**Cpdf::\$currentColour**

*mixed = array('r'=>-1,'g'=>-1,'b'=>-1) [line 76]*

**current colour for fill operations, defaults to inactive value, all three components should be between 0 and 1 inclusive when active**

**Cpdf::\$currentContents**

*mixed = [line 68]*

**object number of the currently active contents block**

**Cpdf::\$currentFont**

*mixed = " [line 48]*

**a record of the current font**

**Cpdf::\$currentFontNum**

*mixed = 0 [line 56]*

**the number of the current font within the font array**

**Cpdf::\$currentLineStyle**

*mixed = " [line 84]*

**current style that lines are drawn in**

**Cpdf::\$currentNode**

*mixed = [line 60]*

**Cpdf::\$currentPage**

*mixed = [line 64]*

### **object number of the current page**

#### **Cpdf::\$currentStrokeColour**

*mixed = array('r'=>-1,'g'=>-1,'b'=>-1) [line 80]*

### **current colour for stroke operations (lines etc.)**

#### **Cpdf::\$currentTextState**

*mixed = " [line 151]*

### **track if the current font is bolded or italicised**

#### **Cpdf::\$destinations**

*mixed = array() [line 188]*

### **store label->id pairs for named destinations, these will be used to replace internal links**

done this way so that destinations can be defined after the location that links to them

#### **Cpdf::\$encrypted**

*mixed = 0 [line 171]*

### **a flag to say if a document is to be encrypted or not**

#### **Cpdf::\$encryptionKey**

*mixed = " [line 175]*

### **the encryption key for the encryption of all the document content (structure is not encrypted)**

#### **Cpdf::\$fileIdentifier**

*mixed = " [line 167]*

### **the file identifier, used to uniquely identify a pdf document**

#### **Cpdf::\$firstPageId**

*mixed = [line 132]*

**the objectId of the first page of the document**

#### Cpdf::\$fontFamilies

*mixed = array() [line 147]*

**store the information about the relationship between font families this used so that the code knows which font is the bold version of another font, etc.**  
the value of this array is initialised in the constuctor function.

#### Cpdf::\$fonts

*mixed = array() [line 44]*

**array carrying information about the fonts that the system currently knows about**  
used to ensure that a font is not loaded twice, among other things

#### Cpdf::\$infoObject

*mixed = 0 [line 119]*

**the objectId of the information object for the document this contains authorship, title etc.**

#### Cpdf::\$looseObjects

*mixed = array() [line 110]*

**an array which contains information about the objects which are not firmly attached to pages**  
these have been added with the addObject function

#### Cpdf::\$messages

*mixed = " [line 155]*

**messages are stored here during processing, these can be selected afterwards to give some useful debug information**

#### Cpdf::\$nCallback

*mixed = 0 [line 183]*

**the number of callback functions in the callback array**

**Cpdf::\$nStack**

*mixed = 0 [line 105]*

**number of elements within the object Id storage stack**

**Cpdf::\$nStateStack**

*mixed = 0 [line 93]*

**number of elements within the state stack**

**Cpdf::\$numFonts**

*mixed = 0 [line 72]*

**number of fonts within the system**

**Cpdf::\$numImages**

*mixed = 0 [line 123]*

**number of images being tracked within the document**

**Cpdf::\$numObj**

*mixed = 0 [line 31]*

**the current number of pdf objects in the document**

**Cpdf::\$numPages**

*mixed = 0 [line 97]*

**number of page objects within the document**

**Cpdf::\$objects**

*mixed = array() [line 35]*

**this array contains all of the pdf objects, ready for final assembly**

**Cpdf::\$options**

*mixed = array('compression'=>1) [line 128]*

### **an array containing options about the document**

it defaults to turning on the compression of the objects

### **Cpdf::\$procsetObjectId**

*mixed = [line 141]*

### **the object Id of the procset object**

### **Cpdf::\$stack**

*mixed = array() [line 101]*

### **object Id storage stack**

### **Cpdf::\$stateStack**

*mixed = array() [line 89]*

### **an array which is used to save the state of the document, mainly the colours and styles**

it is used to temporarily change to another state, the change back to what it was before

### **Cpdf::\$wordSpaceAdjust**

*mixed = 0 [line 137]*

**used to track the last used value of the inter-word spacing, this is so that it is known when the spacing is changed.**

Constructor void function Cpdf::Cpdf([*\$pageSize = array(0,0,612,792)*]) [line 200]

#### **Function Parameters:**

- **\$pageSize**

### **class constructor**

this will start a new document

- **Var** array of 4 numbers, defining the bottom left and upper right corner of the page. first two are normally zero.

*void function Cpdf::addDestination(\$label, \$style, [\$a = 0], [\$b = 0], [\$c = 0]) [line 2972]*

**Function Parameters:**

- **\$label**
- **\$style**
- **\$a**
- **\$b**
- **\$c**

### **create a labelled destination within the document**

*void function Cpdf::addImage(&\$img, \$x, \$y, [\$w = 0], [\$h = 0], [\$quality = 75]) [line 2873]*

**Function Parameters:**

- **&\$img**
- **\$x**
- **\$y**
- **\$w**
- **\$h**
- **\$quality**

### **add an image into the document, from a GD object**

this function is not all that reliable, and I would probably encourage people to use the file based functions

*void function Cpdf::addInfo(\$label, [\$value = 0]) [line 2593]*

**Function Parameters:**

- **\$label**
- **\$value**

### **add content to the documents info object**

*void function Cpdf::addInternalLink(\$label, \$x0, \$y0, \$x1, \$y1) [line 1176]*

***Function Parameters:***

- **\$label**
- **\$x0**
- **\$y0**
- **\$x1**
- **\$y1**

**add a link in the document to an internal destination (ie. within the document)**

*void function Cpdf::addJpegFromFile(\$img, \$x, \$y, [\$w = 0], [\$h = 0]) [line 2828]*

***Function Parameters:***

- **\$img**
- **\$x**
- **\$y**
- **\$w**
- **\$h**

**add a JPEG image into the document, from a file**

*void function Cpdf::addLink(\$url, \$x0, \$y0, \$x1, \$y1) [line 1167]*

***Function Parameters:***

- **\$url**
- **\$x0**
- **\$y0**
- **\$x1**
- **\$y1**

**add a link in the document to an external URL**

*void function Cpdf::addMessage(\$message) [line 3025]*

***Function Parameters:***

- **\$message**

## **used to add messages for use in debugging**

*void function Cpdf::addObject(\$id, [\$options = 'add']) [line 2547]*

**Function Parameters:**

- **\$id**
- **\$options**

**after an object has been created, it wil only show if it has been added, using this function.**

*void function Cpdf::addPngFromFile(\$file, \$x, \$y, [\$w = 0], [\$h = 0]) [line 2640]*

**Function Parameters:**

- **\$file**
- **\$x**
- **\$y**
- **\$w**
- **\$h**

**add a PNG image into the document, from a file**

this should work with remote files

*void function Cpdf::addText(\$x, \$y, \$size, \$text, [\$angle = 0], [\$wordSpaceAdjust = 0]) [line 2159]*

**Function Parameters:**

- **\$x**
- **\$y**
- **\$size**
- **\$text**
- **\$angle**
- **\$wordSpaceAdjust**

**add text to the document, at a specified location, size and angle on the page**

*void function Cpdf::addTextWrap(\$x, \$y, \$width, \$size, \$text, [\$justification = 'left'], [\$angle = 0], [\$test = 0]) [line 2334]*

**Function Parameters:**

- **\$x**
- **\$y**
- **\$width**
- **\$size**
- **\$text**
- **\$justification**
- **\$angle**
- **\$test**

**add text to the page, but ensure that it fits within a certain width if it does not fit then put in as much as possible, splitting at word boundaries and return the remainder.**

justification and angle can also be specified for the text

*void function Cpdf::ARC4(\$text) [line 1141]*

**Function Parameters:**

- **\$text**

### **ARC4 encrypt a text string**

*void function Cpdf::ARC4\_init([\$key = "]) [line 1115]*

**Function Parameters:**

- **\$key**

### **initialize the ARC4 encryption**

*void function Cpdf::checkAllHere() [line 1217]*

**should be used for internal checks, not implemented as yet**

*void function Cpdf::closeObject() [line 2521]*

**close an object**

*void function Cpdf::curve(\$x0, \$y0, \$x1, \$y1, \$x2, \$y2, \$x3, \$y3) [line 1685]*

**Function Parameters:**

- \$x0
- \$y0
- \$x1
- \$y1
- \$x2
- \$y2
- \$x3
- \$y3

## draw a bezier curve based on 4 control points

*void function Cpdf::ellipse(\$x0, \$y0, \$r1, [\$r2 = 0], [\$angle = 0], [\$nSeg = 8], [\$astart = 0], [\$afinish = 360], [\$close = 1], [\$fill = 0]) [line 1716]*

**Function Parameters:**

- \$x0
- \$y0
- \$r1
- \$r2
- \$angle
- \$nSeg
- \$astart
- \$afinish
- \$close
- \$fill

## draw an ellipse note that the part and filled ellipse are just special cases of this function

draws an ellipse in the current line style centered at \$x0,\$y0, radii \$r1,\$r2 if \$r2 is not set, then a circle is drawn nSeg is not allowed to be less than 2, as this will simply draw a line (and will even draw a pretty crappy shape at 2, as we are approximating with bezier curves.

*void function Cpdf::encryptInit(\$id) [line 1101]*

**Function Parameters:**

- \$id

## initialize the encryption for processing a particular object

*void function Cpdf::filledEllipse(\$x0, \$y0, \$r1, [\$r2 = 0], [\$angle = 0], [\$nSeg = 8], [\$astart = 0], [\$afinish = 360]) [line 1702]*

**Function Parameters:**

- \$x0
- \$y0
- \$r1
- \$r2
- \$angle
- \$nSeg
- \$astart
- \$afinish

### **draw a filled ellipse**

*void function Cpdf::filledRectangle(\$x1, \$y1, \$width, \$height) [line 1838]*

**Function Parameters:**

- \$x1
- \$y1
- \$width
- \$height

**a filled rectangle, note that it is the width and height of the rectangle which are the secondary parameters, not  
the coordinates of the upper-right corner**

*void function Cpdf::getFirstPageId() [line 1642]*

**function for the user to find out what the ID is of the first page that was created  
during startup - useful if they wish to add something to it later.**

*void function Cpdf::getFontDecender(\$size) [line 1946]*

**Function Parameters:**

- \$size

**return the font decender, this will normally return a negative number**

**if you add this number to the baseline, you get the level of the bottom of the font it is  
in the pdf user units**

*void* function Cpdf::getFontHeight(\$size) [line 1932]

**Function Parameters:**

- \$size

**return the height in units of the current font in the given size**

*void* function Cpdf::getTextWidth(\$size, \$text) [line 2251]

**Function Parameters:**

- \$size
- \$text

**calculate how wide a given text string will be on a page, at a given size.**

this can be called externally, but is also used by the other class functions

*void* function Cpdf::line(\$x1, \$y1, \$x2, \$y2) [line 1678]

**Function Parameters:**

- \$x1
- \$y1
- \$x2
- \$y2

**draw a line from one set of coordinates to another**

*void* function Cpdf::md5\_16(\$string) [line 1089]

**Function Parameters:**

- \$string

**calculate the 16 byte version of the 128 bit md5 digest of the string**

*void* function Cpdf::newPage([\$insert = 0], [\$id = 0], [\$pos = 'after']) [line 1854]

**Function Parameters:**

- **\$insert**
- **\$id**
- **\$pos**

**add a new page to the document**

this also makes the new page the current active object

*void function Cpdf::openHere(\$style, [\$a = 0], [\$b = 0], [\$c = 0]) [line 2952]*

**Function Parameters:**

- **\$style**
- **\$a**
- **\$b**
- **\$c**

**specify where the document should open when it first starts**

*void function Cpdf::openObject() [line 2493]*

**make a loose object, the output will go into this object, until it is closed, then will revert to the current one.**

this object will not appear until it is included within a page. the function will return the object number

*void function Cpdf::output([\$debug = 0]) [line 1223]*

**Function Parameters:**

- **\$debug**

**return the pdf stream as a string returned from the function**

*void function Cpdf::o\_action(\$id, \$action, [\$options = "]) [line 729]*

**Function Parameters:**

- **\$id**
- **\$action**

- **\$options**

**an action object, used to link to URLs initially**

*void function Cpdf::o\_annotation(\$id, \$action, [\$options = ""]) [line 772]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**an annotation object, this will add an annotation to the current page.**  
initially will support just link annotations

*void function Cpdf::o\_catalog(\$id, \$action, [\$options = ""]) [line 300]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**define the document catalog, the overall controller for the document**

*void function Cpdf::o\_contents(\$id, \$action, [\$options = ""]) [line 893]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**the contents objects hold all of the content which appears on pages**

*void function Cpdf::o\_destination(\$id, \$action, [\$options = ""]) [line 227]*

**Function Parameters:**

- **\$id**

- **\$action**
- **\$options**

**destination object, used to specify the location for the user to jump to, presently on opening**

*void function Cpdf::o\_encryption(\$id, \$action, [\$options = "]) [line 1018]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**encryption object.**

*void function Cpdf::o\_font(\$id, \$action, [\$options = "]) [line 478]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**an object to hold the font description**

*void function Cpdf::o\_fontDescriptor(\$id, \$action, [\$options = "]) [line 562]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**a font descriptor, needed for including additional fonts**

*void function Cpdf::o\_fontEncoding(\$id, \$action, [\$options = "]) [line 613]*

**Function Parameters:**

- **\$id**

- **\$action**
- **\$options**

## the font encoding

*void function Cpdf::o\_image(\$id, \$action, [\$options = ""]) [line 943]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

## an image object, will be an XObject in the document, includes description and data

*void function Cpdf::o\_info(\$id, \$action, [\$options = ""]) [line 685]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

## define the document information

*void function Cpdf::o\_outlines(\$id, \$action, [\$options = ""]) [line 448]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

## define the outlines in the doc, empty for now

*void function Cpdf::o\_page(\$id, \$action, [\$options = ""]) [line 824]*

**Function Parameters:**

- **\$id**
- **\$action**

- **\$options**

**a page object, it also creates a contents object to hold its contents**

*void function Cpdf::o\_pages(\$id, \$action, [\$options = "]) [line 350]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**object which is a parent to the pages in the document**

*void function Cpdf::o\_procset(\$id, \$action, [\$options = "]) [line 650]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**the document procset, solves some problems with printing to old PS printers**

*void function Cpdf::o\_viewerPreferences(\$id, \$action, [\$options = "]) [line 262]*

**Function Parameters:**

- **\$id**
- **\$action**
- **\$options**

**set the viewer preferences**

*void function Cpdf::partEllipse(\$x0, \$y0, \$astart, \$afinish, \$r1, [\$r2 = 0], [\$angle = 0], [\$nSeg = 8]) [line 1695]*

**Function Parameters:**

- **\$x0**
- **\$y0**
- **\$astart**

- **\$afinish**
- **\$r1**
- **\$r2**
- **\$angle**
- **\$nSeg**

### **draw a part of an ellipse**

*void function Cpdf::polygon(\$p, \$np, [\$f = 0]) [line 1821]*

**Function Parameters:**

- **\$p**
- **\$np**
- **\$f**

### **draw a polygon, the syntax for this is similar to the GD polygon command**

*void function Cpdf::rectangle(\$x1, \$y1, \$width, \$height) [line 1846]*

**Function Parameters:**

- **\$x1**
- **\$y1**
- **\$width**
- **\$height**

### **draw a rectangle, note that it is the width and height of the rectangle which are the secondary parameters, not the coordinates of the upper-right corner**

*void function Cpdf::reopenObject(\$id) [line 2508]*

**Function Parameters:**

- **\$id**

### **open an existing object for editing**

*void function Cpdf::restoreState([\$pageEnd = 0]) [line 2474]*

**Function Parameters:**

- **\$pageEnd**

**restore a previously saved state**

*void function Cpdf::saveState([\$pageEnd = 0]) [line 2450]*

**Function Parameters:**

- **\$pageEnd**

**this will be called at a new page to return the state to what it was on the**

end of the previous page, before the stack was closed down This is to get around  
not being able to have open 'q' across pages

*void function Cpdf::selectFont(\$fontName, [\$encoding = ""], [\$set = 1]) [line 1420]*

**Function Parameters:**

- **\$fontName**
- **\$encoding**
- **\$set**

**if the font is not loaded then load it and make the required object**

else just make it the current font the encoding array can contain 'encoding'=>  
'none','WinAnsiEncoding','MacRomanEncoding' or 'MacExpertEncoding' note that  
encoding='none' will need to be used for symbolic fonts and 'differences' => an array  
of mappings between numbers 0->255 and character names.

*void function Cpdf::setColor(\$r, \$g, \$b, [\$force = 0]) [line 1658]*

**Function Parameters:**

- **\$r**
- **\$g**
- **\$b**
- **\$force**

## **sets the colour for fill operations**

*void function Cpdf::setEncryption([UserPass = "], [OwnerPass = "], [\$pc = array()]) [line 1187]*

### **Function Parameters:**

- **\$userPass**
- **\$ownerPass**
- **\$pc**

**set the encryption of the document can be used to turn it on and/or set the passwords which it will have.**

also the functions that the user will have are set here, such as print, modify, add

*void function Cpdf::setFontFamily(\$family, [\$options = "]) [line 2988]*

### **Function Parameters:**

- **\$family**
- **\$options**

**define font families, this is used to initialize the font families for the default fonts and for the user to add new ones for their fonts. The default bahavious can be overridden should that be desired.**

*void function Cpdf::setLineStyle([\$width = 1], [\$cap = "], [\$join = "], [\$dash = "], [\$phase = 0]) [line 1792]*

### **Function Parameters:**

- **\$width**
- **\$cap**
- **\$join**
- **\$dash**
- **\$phase**

**this sets the line drawing style.**

width, is the thickness of the line in user units cap is the type of cap to put on the line, values can be 'butt','round','square' where the difffERENCE between 'square' and 'butt' is that 'square' projects a flat end past the end of the line. join can be 'miter', 'round', 'bevel' dash is an array which sets the dash pattern, is a series of length values, which are the lengths of the on and off dashes. (2) represents 2 on, 2 off, 2 on , 2 off ... (2,1) is 2 on, 1 off, 2 on, 1 off.. etc phase is a modifier on the dash pattern which is

used to shift the point at which the pattern starts.

*void function Cpdf::setPreferences(\$label, [\$value = 0]) [line 2610]*

**Function Parameters:**

- **\$label**
- **\$value**

**set the viewer preferences of the document, it is up to the browser to obey these.**

*void function Cpdf::setStrokeColor(\$r, \$g, \$b, [\$force = 0]) [line 1668]*

**Function Parameters:**

- **\$r**
- **\$g**
- **\$b**
- **\$force**

**sets the colour for stroke operations**

*void function Cpdf::stopObject(\$id) [line 2536]*

**Function Parameters:**

- **\$id**

**stop an object from appearing on pages from this point on**

*void function Cpdf::stream([\$options = ']) [line 1902]*

**Function Parameters:**

- **\$options**

**output the pdf code, streaming it to the browser**

the relevant headers are set so that hopefully the browser will recognise it

*void function Cpdf::transaction(\$action) [line 3032]*

**Function Parameters:**

- **\$action**

**a few functions which should allow the document to be treated transactionally.**



# Package HTML\_TreeMenu Procedural Elements

## file\_dialog.php

**phpDocumentor :: docBuilder Web Interface**

Advanced Web Interface to phpDocumentor

PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** [HTML\\_TreeMenu](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Author** Andrew Eddie
- **Version** CVS: \$Id: file\_dialog.php 231860 2007-03-14 12:48:37Z ashnazg \$
- **Copyright** 2003-2006 Andrew Eddie, Greg Beaver
- **See** [phpdoc.php](#)
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

include\_once **PHPDOC\_WEBROOT\_DIR**."/docbuilder/includes/utilities.php" [line [72](#)]

include\_once **PHPDOC\_WEBROOT\_DIR**."/HTML\_TreeMenu-1.1.2/TreeMenu.php" [line [71](#)]

include\_once **PHPDOC\_WEBROOT\_DIR**."/phpDocumentor/common.inc.php" [line [70](#)]

### common file information

**PHPDOC\_WEBROOT\_DIR** = 'PhpDocumentor' [line [52](#)]

require\_once 'PhpDocumentor/HTML\_TreeMenu-1.1.2/TreeMenu.php' [line [49](#)]

require\_once '@WEB-DIR@'.PATH\_DELIMITER.'PhpDocumentor/docbuilder/includes/utilities.php' [line [50](#)]

require\_once 'PhpDocumentor/phpDocumentor/common.inc.php' [line [48](#)]

### common file information

# Package HTML\_TreeMenu Classes

## Class DirNode

*[line 130]*

### Directory Node

- **Package** HTML\_TreeMenu
- **Author** Richard Heyes < [richard@php.net](mailto:richard@php.net)>
- **Author** Harald Radi < [harald.radi@nme.at](mailto:harald.radi@nme.at)>

#### DirNode::\$path

*string = [line 136]*

**full path to this node**

Constructor **void** function DirNode::DirNode([**\$text** = false], [**\$link** = false], [**\$icon** = false], **\$path**, [**\$events** = array()]) *[line 138]*

**Function Parameters:**

- **\$text**
- **\$link**
- **\$icon**
- **\$path**
- **\$events**

# Class HTML\_TreeMenu

[line 67]

## HTML\_TreeMenu Class

A simple couple of PHP classes and some not so simple Jabbasctipt which produces a tree menu. In IE this menu is dynamic, with branches being collapsable. In IE5+ the status of the collapsed/open branches persists across page refreshes. In any other browser the tree is static. Code is based on work of Harald Radi.

Usage.

After installing the package, copy the example php script to your servers document root. Also place the TreeMenu.js and the images folder in the same place. Running the script should then produce the tree.

Thanks go to Chip Chapin (<http://www.chipchapin.com>) for many excellent ideas and improvements.

- **Package** HTML\_TreeMenu
- **Author** Harald Radi < [harald.radi@nme.at](mailto:harald.radi@nme.at)>
- **Author** Richard Heyes < [richard@php.net](mailto:richard@php.net)>
- **Access** public

## HTML\_TreeMenu::\$items

array = [line 73]

### Indexed array of subnodes

Constructor void function HTML\_TreeMenu::HTML\_TreeMenu() [line 80]

#### Constructor

- **Access** public

object Returns function HTML\_TreeMenu::addItem(&\$node, \$node) [line 94]

#### Function Parameters:

- **object \$node** The node to add. This object should be a `HTML_TreeNode` object.
- **&\$node**

**This function adds an item to the tree.**

- **Access** public

## Class `HTML_TreeMenu_DHTML` *[line 329]*

### **`HTML_TreeMenu_DHTML` class**

This class is a presentation class for the tree structure created using the `TreeMenu/TreeNode`. It presents the traditional tree, static for browsers that can't handle the DHTML.

- **Package** `HTML_TreeMenu`

### **`HTML_TreeMenu_DHTML::$defaultClass`**

*mixed = [line 359]*

### **The default CSS class for the nodes**

### **`HTML_TreeMenu_DHTML::$images`**

*string = [line 342]*

### **Path to the images**

## **HTML\_TreeMenu\_DHTML::\$isDynamic**

*mixed = [line 336]*

**Dynamic status of the treemenu. If true (default) this has no effect. If false it will override all dynamic status vars and set the menu to be fully expanded an non-dynamic.**

## **HTML\_TreeMenu\_DHTML::\$linkTarget**

*string = [line 348]*

**Target for the links generated**

## **HTML\_TreeMenu\_DHTML::\$noTopLevelImages**

*bool = [line 365]*

**Whether to skip first level branch images**

## **HTML\_TreeMenu\_DHTML::\$userPersistence**

*bool = [line 354]*

**Whether to use clientside persistence or not**

Constructor void function `HTML_TreeMenu_DHTML::HTML_TreeMenu_DHTML(&$structure, [$options = array()], [$isDynamic = true], $structure)` [line 386]

**Function Parameters:**

- *object \$structure* The menu structure
- *array \$options* Array of options
- *bool \$isDynamic* Whether the tree is dynamic or not
- *&\$structure*

**Constructor, takes the tree structure as**

an argument and an array of options which can consist of:

- *images* - The path to the images folder. Defaults to "images"
- *linkTarget* - The target for the link. Defaults to "\_self"
- *defaultClass* - The default CSS class to apply to a node. Default is none.
- *usePersistence* - Whether to use clientside persistence. This persistence is achieved using cookies. Default is true.
- *noTopLevelImages* - Whether to skip displaying the first level of images if there is multiple top level branches.

And also a boolean for whether the entire tree is dynamic or not. This overrides any perNode dynamic settings.

*string* function **HTML\_TreeMenu\_DHTML::toHTML()** [*line 411*]

**Returns the HTML for the menu. This method can be used instead of printMenu() to use the menu system with a template system.**

- **Access** public

## Class **HTML\_TreeMenu\_Listbox** [*line 493*]

### **HTML\_TreeMenu\_Listbox class**

This class presents the menu as a listbox

- **Package** **HTML\_TreeMenu**

**HTML\_TreeMenu\_Listbox::\$indentChar**

*string* = [*line 505*]

**The character used for indentation**

**HTML\_TreeMenu\_Listbox::\$indentNum**

*integer* = [*line 512*]

**How many of the indent chars to use**  
per indentation level

## **HTML\_TreeMenu\_Listbox::\$linkTarget**

*string = [line 518]*

**Target for the links generated**

## **HTML\_TreeMenu\_Listbox::\$promoText**

*string = [line 499]*

**The text that is displayed in the first option**

Constructor *void* function **HTML\_TreeMenu\_Listbox::HTML\_TreeMenu\_Listbox(\$structure, [\$options = array()])** [line 535]

**Function Parameters:**

- *object \$structure* The menu structure
- *array \$options* Options which affect the display of the listbox. These can consist of:
  - *promoText* The text that appears at the top of the listbox  
Defaults to "Select..."
  - *indentChar* The character to use for indenting the nodes  
Defaults to "&nbsp;"
  - *indentNum* How many of the indentChars to use per indentation level  
Defaults to 2
  - *linkTarget* Target for the links. Defaults to "\_self"
  - *submitText* Text for the submit button. Defaults to "Go"

## **Constructor**

*void* function **HTML\_TreeMenu\_Listbox::toHTML()** [line 552]

**Returns the HTML generated**

# **Class HTML\_TreeMenu\_Presentation**

[line 282]

## **HTML\_TreeMenu\_Presentation class**

Base class for other presentation classes to inherit from.

- **Package** `HTML_TreeMenu`

### `HTML_TreeMenu_Presentation::$menu`

*object = [line 288]*

#### **The TreeMenu structure**

Constructor `void function HTML_TreeMenu_Presentation::HTML_TreeMenu_Presentation(&$structure, $structure) [line 295]`

**Function Parameters:**

- *object* **\$structure** The menu structure
- **&\$structure**

#### **Base constructor simply sets the menu object**

`void function HTML_TreeMenu_Presentation::printMenu([&$options = array()]) [line 310]`

**Function Parameters:**

- *array* **\$options** Options to set. Any options taken by specified here. the presentation class can be

#### **Prints the HTML generated by the `toHTML()` method.**

`toHTML()` must therefore be defined by the derived class.

- **Access** public

# Class `HTML_TreeNode`

*[line 113]*

## **HTML\_TreeNode class**

This class is supplementary to the above and provides a way to add nodes to the tree. A node can have other nodes added to it.

- **Package** `HTML_TreeMenu`
- **Author** Harald Radi < [harald.radi@nme.at](mailto:harald.radi@nme.at)>
- **Author** Richard Heyes < [richard@php.net](mailto:richard@php.net)>
- **Access** public

### **HTML\_TreeNode::\$cssClass**

*string = [line 137]*

#### **The css class for this node**

### **HTML\_TreeNode::\$ensureVisible**

*bool = [line 161]*

#### **Should this node be made visible?**

### **HTML\_TreeNode::\$events**

*array = [line 173]*

#### **Javascript event handlers;**

### **HTML\_TreeNode::\$expanded**

*bool = [line 149]*

#### **Whether this node is expanded or not**

### **HTML\_TreeNode::\$icon**

*string = [line 131]*

#### **The icon for this node.**

### **HTML\_TreeNode::\$isDynamic**

*bool = [line 155]*

**Whether this node is dynamic or not**

### **HTML\_TreeNode::\$items**

*array = [line 143]*

**Indexed array of subnodes**

### **HTML\_TreeNode::\$link**

*string = [line 125]*

**The link for this node.**

### **HTML\_TreeNode::\$parent**

*object = [line 167]*

**The parent node. Null if top level**

### **HTML\_TreeNode::\$text**

*string = [line 119]*

**The text for this node.**

Constructor **void** function **HTML\_TreeNode::HTML\_TreeNode([\$options = array()], [\$events = array()])** [line 197]

#### **Function Parameters:**

- **array \$options** An array of options which you can pass to change this node looks/acts. This can consist of:
  - **text** The title of the node, defaults to blank
  - **link** The link for the node, defaults to blank
  - **icon** The icon for the node, defaults to blank
  - **class** The CSS class for this node, defaults to blank
  - **expanded** The default expanded status of this node, defaults to false
    - This doesn't affect non dynamic presentation types
  - **isDynamic** If this node is dynamic or not. Only affects certain presentation types.
  - **ensureVisible** If true this node will be made visible despite the expanded settings, and client side persistence. Will not affect

some presentation styles, such as Listbox. Default is false

- **array \$events** An array of javascript events and the corresponding event handlers.  
Additionally to the standard javascript events you can specify handlers for the 'onexpand', 'oncollapse' and 'ontoggle' events which will be fired whenever a node is collapsed and/or expanded.

## Constructor

- **Access** public

*void* function HTML\_TreeNode::addItem(&\$node, \$node) [line 242]

**Function Parameters:**

- *object \$node* The new node
- **&\$node**

**Adds a new subnode to this node.**

- **Access** public

*void* function HTML\_TreeNode::setOption(\$option, \$value) [line 231]

**Function Parameters:**

- *string \$option* Option to set
- *string \$value* Value to set the option to

**Allows setting of various parameters after the initial constructor call.** Possible options you can set are:

- text
- link
- icon

- cssClass
- expanded
- isDynamic
- ensureVisible
  - ie The same options as in the constructor

- **Access** public



# Package org-phpdoc Procedural Elements

bug-904820.php

This is a test of a package with a . in its name

- **Package** org-phpdoc

element = true [line 12]

I'm an element



# Package Smarty Procedural Elements

## Config\_File.class.php

### **Config\_File class.**

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

You may contact the author of Config\_File by e-mail at: [andrei@php.net](mailto:andrei@php.net)

The latest version of Config\_File can be obtained from: <http://smarty.php.net/>

- **Package** Smarty
- **Author** Andrei Zmievski < [andrei@php.net](mailto:andrei@php.net)>
- **Version** 2.6.0
- **Copyright** Copyright: 2001-2003 ispi of Lincoln, Inc.
- **Link** <http://smarty.php.net/>
- **Access** public

# Smarty.class.php

**Project: Smarty: the PHP compiling template engine File: Smarty.class.php**

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For questions, help, comments, discussion, etc., please join the Smarty mailing list. Send a blank e-mail to [smarty-general-subscribe@lists.php.net](mailto:smarty-general-subscribe@lists.php.net)

You may contact the authors of Smarty by e-mail at: [monte@ispi.net](mailto:monte@ispi.net) [andrei@php.net](mailto:andrei@php.net)

Or, write to: Monte Ohrt Director of Technology, ispi 237 S. 70th suite 220 Lincoln, NE 68510

The latest version of Smarty can be obtained from: <http://smarty.php.net/>

- **Package** Smarty
- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Author** Andrei Zmievski < [andrei@php.net](mailto:andrei@php.net)>
- **Version** 2.6.0
- **Copyright** 2001-2003 ispi of Lincoln, Inc.
- **Link** <http://smarty.php.net/>

DIR\_SEP = DIRECTORY\_SEPARATOR [line 52]

**DIR\_SEP isn't used anymore, but third party apps might**

SMARTY\_DIR = dirname(\_\_FILE\_\_).DIRECTORY\_SEPARATOR [line 62]

**set SMARTY\_DIR to absolute path to Smarty library files.**

if not defined, include\_path will be used. Sets SMARTY\_DIR only if user application has not already defined it.

SMARTY\_PHP\_ALLOW = 3 [line 68]  
SMARTY\_PHP\_PASSTHRU = 0 [line 65]  
SMARTY\_PHP\_QUOTE = 1 [line 66]  
SMARTY\_PHP\_REMOVE = 2 [line 67]

## Smarty\_Compiler.class.php

Project: Smarty: the PHP compiling template engine File:

### Smarty\_Compiler.class.php

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

You may contact the authors of Smarty by e-mail at: [monte@ispi.net](mailto:monte@ispi.net) [andrei@php.net](mailto:andrei@php.net)

Or, write to: Monte Ohrt Director of Technology, ispi 237 S. 70th suite 220 Lincoln, NE 68510

The latest version of Smarty can be obtained from: <http://smarty.php.net/>

- **Package** Smarty
- **Author** Andrei Zmievski < [andrei@php.net](mailto:andrei@php.net)>
- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 2.6.0
- **Copyright** 2001-2003 ispi of Lincoln, Inc.
- **Link** <http://smarty.php.net/>

# Package Smarty Classes

## Class Config\_File [line 39]

**Config file reading class**

- **Package** Smarty

**Config\_File::\$booleanize**

*boolean = true [line 53]*

**Controls whether config values of on/true/yes and off/false/no get converted to boolean values automatically.**

Options

**Config\_File::\$fix\_newlines**

*boolean = true [line 64]*

**Controls whether or not to fix mac or dos formatted newlines.**

Options If set to true, \r or \r\n will be changed to \n.

**Config\_File::\$overwrite**

*boolean = true [line 47]*

**Controls whether variables with the same name overwrite each other.**

## Options

### **Config\_File::\$read\_hidden**

*boolean = true [line 58]*

**Controls whether hidden config sections/vars are read from the file.**

## Options

### **Config\_File::\$\_config\_data**

*mixed = array() [line 69]*

Constructor *void function Config\_File::Config\_File([&\$config\_path = NULL]) [line 77]*

**Function Parameters:**

- *string \$config\_path* (optional) path to the config files

**Constructs a new config file class.**

*void function Config\_File::clear([&\$file\_name = NULL]) [line 218]*

**Function Parameters:**

- *string \$file\_name* file to clear config data for

**Clear loaded config data for a certain file or all files.**

*string|array function Config\_File::get(&\$file\_name, [&\$section\_name = NULL], [&\$var\_name = NULL]) [line 113]*

**Function Parameters:**

- *string \$file\_name* config file to get info for
- *string \$section\_name* (optional) section to get info for
- *string \$var\_name* (optional) variable to get info for

**Retrieves config info based on the file, section, and variable name.**

- **Used by** [Config\\_File::get\\_key\(\)](#) - retrieves information from config file and returns it

array function Config\_File::get\_file\_names() [line 165]

### Get all loaded config file names.

string/array function Config\_File::get\_key(\$config\_key) [line 153]

#### Function Parameters:

- **\$file\_name \$config\_key** string config key (filename/section/var)

### Retrieves config info based on the key.

- **Uses** [Config\\_File::get\(\)](#) - retrieves information from config file and returns it

array function Config\_File::get\_section\_names(\$file\_name) [line 177]

#### Function Parameters:

- **string \$file\_name** config file to get section names from

### Get all section names from a loaded file.

array function Config\_File::get\_var\_names(\$file\_name, [\$section = NULL], \$section\_name) [line 196]

#### Function Parameters:

- **string \$file\_name** config file to get info for
- **string \$section\_name** (optional) section to get info for
- **\$section**

### Get all global or section variable names.

void function Config\_File::load\_file(\$file\_name, [\$prepend\_path = true]) [line 234]

#### Function Parameters:

- *string \$file\_name* file name to load
- *boolean \$prepend\_path* whether current config path should be prepended to the filename

### Load a configuration file manually.

*void function Config\_File::set\_path(\$config\_path) [line 89]*

#### **Function Parameters:**

- *string \$config\_path* path to the config files

### Set the path where configuration files can be found.

## Class Smarty

*[line 73]*

- **Package** Smarty

**Smarty::\$autoload\_filters**

*array = array() [line 357]*

### This indicates which filters are automatically loaded into Smarty.

Smarty Configuration Section

- **Var** array of filter names

**Smarty::\$cache\_dir**

*string = 'cache' [line 169]*

**The name of the directory for cache files.**

Smarty Configuration Section

**Smarty::\$cache\_handler\_func**

*null|string = null [line 333]*

**The function used for cache file handling. If not set, built-in caching is used.**

Smarty Configuration Section

- **Var** function name

**Smarty::\$cache\_lifetime**

*integer = 3600 [line 180]*

**This is the number of seconds cached content will persist.**

Smarty Configuration Section

- 0 = always regenerate cache
- -1 = never expires

**Smarty::\$cache\_modified\_check**

*boolean = false [line 190]*

**Only used when \$caching is enabled. If true, then If-Modified-Since headers are respected with cached content, and appropriate HTTP headers are sent.**

Smarty Configuration Section This way repeated hits to a cached page do not send the entire page to the client every time.

**Smarty::\$caching**

*integer = 0 [line 162]*

### This enables template caching.

Smarty Configuration Section

- 0 = no caching
- 1 = use class cache\_lifetime value
- 2 = use cache\_lifetime in cache file

### Smarty::\$compiler\_class

*string = 'Smarty\_Compiler' [line 411]*

### The class used for compiling templates.

### Smarty::\$compiler\_file

*string = 'Smarty\_Compiler.class.php' [line 404]*

### The file that contains the compiler class. This can a full pathname, or relative to the php\_include path.

### Smarty::\$compile\_check

*boolean = true [line 143]*

### This tells Smarty whether to check for recompiling or not. Recompiling does not need to happen unless a template or config file is changed.

Smarty Configuration Section Typically you enable this during development, and disable for production.

### Smarty::\$compile\_dir

*string = 'templates\_c' [line 91]*

### The directory where compiled templates are located.

Smarty Configuration Section

### Smarty::\$compile\_id

*string = null [line 293]*

### Set this if you want different sets of compiled files for the same templates. This is useful for things like different languages.

Smarty Configuration Section Instead of creating separate sets of templates per language, you set different compile\_ids like 'en' and 'de'.

### **Smarty::\$config\_booleanize**

*boolean = true [line 373]*

**This tells whether or not to automatically booleanize config file variables.**

If enabled, then the strings "on", "true", and "yes" are treated as boolean true, and "off", "false" and "no" are treated as boolean false.

### **Smarty::\$config\_class**

*string = 'Config\_File' [line 418]*

**The class used to load config vars.**

### **Smarty::\$config\_dir**

*string = 'configs' [line 98]*

**The directory where config files are located.**

Smarty Configuration Section

### **Smarty::\$config\_fix\_newlines**

*boolean = true [line 387]*

**This tells whether or not automatically fix newlines in config files.**

It basically converts \r (mac) or \r\n (dos) to \n

### **Smarty::\$config\_overwrite**

*boolean = true [line 366]*

**This tells if config file vars of the same name overwrite each other or not.**

if disabled, same name variables are accumulated in an array.

### **Smarty::\$config\_read\_hidden**

*boolean = false [line 381]*

**This tells whether hidden sections [.foobar] are readable from the templates or**

**not. Normally you would never allow this since that is the point behind hidden sections: the application can access them, but the templates cannot.**

### **Smarty::\$debugging**

*boolean = false [line 114]*

**If debugging is enabled, a debug console window will display when the page loads  
(make sure your browser allows unrequested popup windows)**  
Smarty Configuration Section

### **Smarty::\$debugging\_ctrl**

*string = 'NONE' [line 133]*

**This determines if debugging is enable-able from the browser.**

Smarty Configuration Section

- **NONE => no debugging control allowed**
- **URL => enable debugging when SMARTY\_DEBUG is found in the URL.**

- **Link [http://www.foo.dom/index.php?SMARTY\\_DEBUG](http://www.foo.dom/index.php?SMARTY_DEBUG)**

### **Smarty::\$debug\_tpl**

*string = " [line 122]*

**This is the path to the debug console template. If not set, the default one will be used.**

Smarty Configuration Section

### **Smarty::\$default\_modifiers**

*array = array() [line 312]*

**This is a list of the modifiers to apply to all template variables.**

Smarty Configuration Section Put each modifier in a separate array element in the order you want them applied. example:

```
1   array('escape:htmlall' );
```

### **Smarty::\$default\_resource\_type**

*array = 'file' [line 326]*

#### **This is the resource type to be used when not specified**

Smarty Configuration Section at the beginning of the resource path. examples:  
\$smarty->display('file:index.tpl'); \$smarty->display('db:index.tpl'); \$smarty->display('index.tpl'); // will use default resource type {include file="file:index.tpl"} {include file="db:index.tpl"} {include file="index.tpl"} {\* will use default resource type \*}

### **Smarty::\$default\_template\_handler\_func**

*string = " [line 396]*

#### **If a template cannot be found, this PHP function will be executed.**

Useful for creating templates on-the-fly or other special action.

- Var function name

### **Smarty::\$force\_compile**

*boolean = false [line 151]*

#### **This forces templates to compile every time. Useful for development or debugging.**

Smarty Configuration Section

### **Smarty::\$global\_assign**

*array = array('HTTP\_SERVER\_VARS' => array('SCRIPT\_NAME')) [line 343]*

#### **These are the variables from the globals array that are assigned to all templates automatically. This isn't really necessary any more, you can use the \$smarty var to access them directly.**

Smarty Configuration Section

- **Used by** [Smarty::Smarty\(\)](#) - uses [assign\(\)](#) to assign each corresponding value from \$GLOBALS to the template vars

### **Smarty::\$left\_delimiter**

*string = '{' [line 257]*

#### **The left delimiter used for the template tags.**

Smarty Configuration Section

### **Smarty::\$php\_handling**

*integer = SMARTY\_PHP\_PASSTHRU [line 204]*

#### **This determines how Smarty handles "<?php ... ?>" tags in templates.**

Smarty Configuration Section possible values:

- SMARTY\_PHP\_PASSTHRU -> print tags as plain text
- SMARTY\_PHP\_QUOTE -> escape tags as entities
- SMARTY\_PHP\_REMOVE -> remove php tags
- SMARTY\_PHP\_ALLOW -> execute php tags

### **Smarty::\$plugins\_dir**

*array = array('plugins') [line 105]*

#### **An array of directories searched for plugins.**

Smarty Configuration Section

### **Smarty::\$request\_use\_auto\_globals**

*boolean = false [line 283]*

#### **Indicates whether \$HTTP\_\*\_VARS[] (request\_use\_auto\_globals=false)**

Smarty Configuration Section are used as request-vars or \$\_\*[]-vars. note: if request\_use\_auto\_globals is true, then \$request\_vars\_order has no effect, but the phpini-value "gpc\_order"

### **Smarty::\$request\_vars\_order**

*string = "EGPCS" [line 273]*

**The order in which request variables are registered, similar to `variables_order` in `php.ini` E = Environment, G = GET, P = POST, C = Cookies, S = Server**  
Smarty Configuration Section

#### **Smarty::\$right\_delimiter**

*string = '}' [line 264]*

**The right delimiter used for the template tags.**  
Smarty Configuration Section

#### **Smarty::\$secure\_dir**

*array = array() [line 223]*

**This is the list of template directories that are considered secure. This is used only if `$security` is enabled. One directory per array element. `$template_dir` is in this list implicitly.**

Smarty Configuration Section

#### **Smarty::\$security**

*boolean = false [line 214]*

**This enables template security. When enabled, many things are restricted**  
Smarty Configuration Section in the templates that normally would go unchecked.  
This is useful when untrusted parties are editing templates and you want a reasonable level of security. (no direct execution of PHP in templates for example)

#### **Smarty::\$security\_settings**

*array = array(  
 'PHP\_HANDLING' => false,  
 'IF\_FUNCS' => array('array', 'list',  
 'isset', 'empty',  
 'count', 'sizeof',  
 'in\_array', 'is\_array',  
  
 'true', 'false'), 'INCLUDE\_ANY'=>false, 'PHP\_TAGS'=>false, 'MODIFIER\_FUNCS'=>array('count'), 'ALLOW\_CONSTANTS'=>false) [line 231]*

**These are the security settings for Smarty. They are used only when `$security` is**

**enabled.**

Smarty Configuration Section

**Smarty::\$template\_dir**

*string = 'templates' [line 84]*

**The name of the directory where templates are located.**

Smarty Configuration Section

**Smarty::\$trusted\_dir**

*array = array() [line 250]*

**This is an array of directories where trusted php scripts reside.**

Smarty Configuration Section [\\$security](#) is disabled during their inclusion/execution.

**Smarty::\$undefined**

*null = null [line 350]*

**The value of "undefined". Leave it alone :-)**

Smarty Configuration Section

**Smarty::\$use\_sub\_dirs**

*boolean = true [line 303]*

**This tells Smarty whether or not to use sub dirs in the cache/ and templates\_c/ directories. sub directories better organized, but may not work well with PHP safe mode enabled.**

Smarty Configuration Section

Constructor void function Smarty::Smarty() [line 597]

**The class constructor.**

- **Uses [Smarty::\\$global\\_assign](#)** - uses [assign\(\)](#) to assign each corresponding value from \$GLOBALS to the template vars

*void* function Smarty::append(\$tpl\_var, [\$value = null], [\$merge = false]) [line 657]  
**Function Parameters:**

- *array/string \$tpl\_var* the template variable name(s)
- *mixed \$value* the value to append
- *\$merge*

### appends values to template variables

*void* function Smarty::append\_by\_ref(\$tpl\_var, &\$value, [\$merge = false], \$value) [line 697]  
**Function Parameters:**

- *string \$tpl\_var* the template variable name
- *mixed \$value* the referenced value to append
- *&\$value*
- *\$merge*

### appends values to template variables by reference

*void* function Smarty::assign(\$tpl\_var, [\$value = null]) [line 625]  
**Function Parameters:**

- *array/string \$tpl\_var* the template variable name(s)
- *mixed \$value* the value to assign

### assigns values to template variables

*void* function Smarty::assign\_by\_ref(\$tpl\_var, &\$value, \$value) [line 645]  
**Function Parameters:**

- *string \$tpl\_var* the template variable name
- *mixed \$value* the referenced value to assign
- *&\$value*

### assigns values to template variables by reference

*void* function Smarty::clear\_all\_assign() [*line 1062*]  
**clear all the assigned template variables.**

*boolean* function Smarty::clear\_all\_cache([*\$exp\_time = null*]) [*line 1015*]  
**Function Parameters:**

- *string \$exp\_time* expire time

### **clear the entire contents of cache (all templates)**

*void* function Smarty::clear\_assign(*\$tpl\_var*) [*line 719*]  
**Function Parameters:**

- *string \$tpl\_var* the template variable to clear

### **clear the given assigned template variable.**

*boolean* function Smarty::clear\_cache([*\$tpl\_file = null*], [*\$cache\_id = null*], [*\$compile\_id = null*], [*\$exp\_time = null*]) [*line 983*]

**Function Parameters:**

- *string \$tpl\_file* name of template file
- *string \$cache\_id* name of cache\_id
- *string \$compile\_id* name of compile\_id
- *string \$exp\_time* expiration time

### **clear cached content for the given template and cache id**

*boolean* function Smarty::clear\_compiled\_tpl([*\$tpl\_file = null*], [*\$compile\_id = null*], [*\$exp\_time = null*]) [*line 1077*]

**Function Parameters:**

- *string \$tpl\_file*
- *string \$compile\_id*
- *string \$exp\_time*

### **clears compiled version of specified template resource, or all compiled template**

## **files if one is not specified.**

This function is for advanced use only, not normally needed.

*void* function Smarty::clear\_config([*\$var* = null]) [line 1381]

### **Function Parameters:**

- *string* **\$var**

## **clear configuration values**

*void* function Smarty::config\_load(*\$file*, [*\$section* = null], [*\$scope* = 'global']) [line 1354]

### **Function Parameters:**

- *string* **\$file**
- *string* **\$section**
- *string* **\$scope**

## **load configuration values**

*void* function Smarty::display(*\$resource\_name*, [*\$cache\_id* = null], [*\$compile\_id* = null]) [line 1155]

### **Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$cache\_id**
- *string* **\$compile\_id**

## **executes & displays the template results**

*void* function Smarty::fetch(*\$resource\_name*, [*\$cache\_id* = null], [*\$compile\_id* = null], [*\$display* = false]) [line 1168]

### **Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$cache\_id**
- *string* **\$compile\_id**
- *boolean* **\$display**

## **executes & returns or displays the template results**

*array* function Smarty::get\_config\_vars([*\$name* = null], *\$type*) [line 1127]

### **Function Parameters:**

- *string* **\$name**
- *string* **\$type**

## **Returns an array containing config variables**

*object* function Smarty::get\_registered\_object(*\$name*) [line 1366]

### **Function Parameters:**

- *string* **\$name**

## **return a reference to a registered object**

*array* function Smarty::get\_template\_vars([*\$name* = null], *\$type*) [line 1110]

### **Function Parameters:**

- *string* **\$name**
- *string* **\$type**

## **Returns an array containing template variables**

*string/false* function Smarty::is\_cached(*\$tpl\_file*, [*\$cache\_id* = null], [*\$compile\_id* = null]) [line 1040]

### **Function Parameters:**

- *string* **\$tpl\_file** name of template file
- *string* **\$cache\_id**
- *string* **\$compile\_id**

## **test to see if valid cache exists for this template**

*void* function Smarty::load\_filter(*\$type*, *\$name*) [line 957]

**Function Parameters:**

- *string* **\$type** filter type
- *string* **\$name** filter name

## load a filter of specified type and name

*void* function Smarty::register\_block(\$block, \$block\_impl, [\$cacheable = true], [\$cache\_attrs = null]) [line 786]

**Function Parameters:**

- *string* **\$block** name of template block
- *string* **\$block\_impl** PHP function to register
- **\$cacheable**
- **\$cache\_attrs**

## Registers block function to be used in templates

*void* function Smarty::register\_compiler\_function(\$function, \$function\_impl, [\$cacheable = true]) [line 808]

**Function Parameters:**

- *string* **\$function** name of template function
- *string* **\$function\_impl** name of PHP function to register
- **\$cacheable**

## Registers compiler function

*void* function Smarty::register\_function(\$function, \$function\_impl, [\$cacheable = true], [\$cache\_attrs = null]) [line 735]

**Function Parameters:**

- *string* **\$function** the name of the template function
- *string* **\$function\_impl** the name of the PHP function to register
- **\$cacheable**
- **\$cache\_attrs**

## Registers custom function to be used in templates

*void function Smarty::register\_modifier(\$modifier, \$modifier\_impl) [line 830]*

**Function Parameters:**

- *string \$modifier* name of template modifier
- *string \$modifier\_impl* name of PHP function to register

## Registers modifier to be used in templates

*void function Smarty::register\_object(\$object, &\$object\_impl, [\$allowed = array()], [\$smarty\_args = true], [\$block\_methods = array()], \$block\_functs) [line 761]*

**Function Parameters:**

- *string \$object* name of template object
- *object &\$object\_impl* the referenced PHP object to register
- *null|array \$allowed* list of allowed methods (empty = all)
- *boolean \$smarty\_args* smarty argument format, else traditional
- *null|array \$block\_functs* list of methods that are block format
- *\$block\_methods*

## Registers object to be used in templates

*void function Smarty::register\_outputfilter(\$function) [line 934]*

**Function Parameters:**

- *string \$function* name of PHP function

## Registers an output filter function to apply to a template output

*void function Smarty::register\_postfilter(\$function) [line 911]*

**Function Parameters:**

- *string \$function* name of PHP function to register

## Registers a postfilter function to apply to a compiled template after compilation

*void function Smarty::register\_prefilter(\$function) [line 888]*

**Function Parameters:**

- *string* **\$function** name of PHP function to register

## Registers a prefilter function to apply to a template before compiling

*void* function Smarty::register\_resource(**\$type**, **\$functions**) [line 852]

### Function Parameters:

- *string* **\$type** name of resource
- *array* **\$functions** array of functions to handle resource

## Registers a resource to fetch a template

*boolean* function Smarty::template\_exists(**\$tpl\_file**) [line 1097]

### Function Parameters:

- *string* **\$tpl\_file**

## Checks whether requested template exists.

*void* function Smarty::trigger\_error(**\$error\_msg**, [**\$error\_type** = E\_USER\_WARNING]) [line 1142]

### Function Parameters:

- *string* **\$error\_msg**
- *integer* **\$error\_type**

## trigger Smarty error

*void* function Smarty::unregister\_block(**\$block**) [line 797]

### Function Parameters:

- *string* **\$block** name of template function

## Unregisters block function

*void* function Smarty::unregister\_compiler\_function(**\$function**) [line 819]

**Function Parameters:**

- *string* **\$function** name of template function

## Unregisters compiler function

*void* function Smarty::unregister\_function(**\$function**) [line 747]

**Function Parameters:**

- *string* **\$function** name of template function

## Unregisters custom function

*void* function Smarty::unregister\_modifier(**\$modifier**) [line 841]

**Function Parameters:**

- *string* **\$modifier** name of template modifier

## Unregisters modifier

*void* function Smarty::unregister\_object(**\$object**) [line 774]

**Function Parameters:**

- *string* **\$object** name of template object

## Unregisters object

*void* function Smarty::unregister\_outputfilter(**\$function**) [line 946]

**Function Parameters:**

- *string* **\$function** name of PHP function

## Unregisters an outputfilter function

*void* function Smarty::unregister\_postfilter(\$function) [line 923]

**Function Parameters:**

- *string* **\$function** name of PHP function

## Unregisters a postfilter function

*void* function Smarty::unregister\_prefilter(\$function) [line 900]

**Function Parameters:**

- *string* **\$function** name of PHP function

## Unregisters a prefilter function

*void* function Smarty::unregister\_resource(\$type) [line 877]

**Function Parameters:**

- *string* **\$type** name of resource

## Unregisters a resource

*boolean* function Smarty::\_compile\_resource(\$resource\_name, \$compile\_path) [line 1446]

**Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$compile\_path**

## compile the template

*boolean* function Smarty::\_compile\_source(\$resource\_name, &\$source\_content, &\$compiled\_content, [\$cache\_include\_path = null], \$source\_content, \$compiled\_content) [line 1485]

**Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$source\_content**

- *string* **\$compiled\_content**
- **&\$source\_content**
- **&\$compiled\_content**
- **\$cache\_include\_path**

### compile the given source

*string* function Smarty::\_dequote(\$string) [line 1732]

**Function Parameters:**

- *string* **\$string**

### Remove starting and ending quotes from the string

*mixed* function Smarty::\_eval(\$code, [\$params = null]) [line 2000]

**Function Parameters:**

- **\$code**
- **\$params**

### wrapper for eval() retaining \$this

*boolean* function Smarty::\_fetch\_resource\_info(&\$params, \$resource\_name, \$source\_content, \$resource\_timestamp, \$get\_source, \$quiet) [line 1565]

**Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$source\_content**
- *integer* **\$resource\_timestamp**
- *boolean* **\$get\_source**
- *boolean* **\$quiet**
- **&\$params**

### fetch the template info. Gets timestamp, and source if get\_source is true

sets \$source\_content to the source of the template, and \$resource\_timestamp to its time stamp

*string* function Smarty::\_\_get\_auto\_filename(\$auto\_base, [\$auto\_source = null], [\$auto\_id = null]) [line 1796]

**Function Parameters:**

- *string* \$auto\_base
- *string* \$auto\_source
- *string* \$auto\_id

**get a concrete filename for automagically created content**

- **Static Variable Used** string|null \$cache\_info: 1
- **Static Variable Used** string|null \$\_cache\_info:

*string|null* function Smarty::\_\_get\_auto\_id([\$cache\_id = null], [\$compile\_id = null]) [line 1854]

**Function Parameters:**

- *string* \$cache\_id
- *string* \$compile\_id

**returns an auto\_id for auto-file-functions**

*string* function Smarty::\_\_get\_compile\_path(\$resource\_name) [line 1545]

**Function Parameters:**

- *string* \$resource\_name

**Get the compile path for this resource**

*string|false* function Smarty::\_\_get\_plugin\_filepath(\$type, \$name) [line 1399]

**Function Parameters:**

- *string* \$type
- *string* \$name

## get filepath of requested plugin

*mixed* function Smarty::\_include(\$filename, [\$once = false], [\$params = null]) [line 1986]

**Function Parameters:**

- **\$filename**
- **\$once**
- **\$params**

## wrapper for include() retaining \$this

*boolean* function Smarty::\_is\_compiled(\$resource\_name, \$compile\_path) [line 1413]

**Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$compile\_path**

## test if resource needs compiling

*boolean* function Smarty::\_parse\_resource\_name(&\$params, \$resource\_base\_path, \$resource\_name, \$resource\_type) [line 1647]

**Function Parameters:**

- *string* **\$resource\_base\_path**
- *string* **\$resource\_name**
- *string* **\$resource\_type**
- **&\$params**

## parse out the type and name from the resource

*string* function Smarty::\_process\_compiled\_include\_callback(\$match) [line 1894]

**Function Parameters:**

- **\$match**

## callback function for preg\_replace, to call a non-cacheable block

*string* function Smarty::\_read\_file(\$filename, [\$start = null], [\$lines = null]) [line 1751]

**Function Parameters:**

- *string* **\$filename**
- *integer* **\$start**
- *integer* **\$lines**

**read in a file from line \$start for \$lines.**

read the entire file if \$start and \$lines are null.

*string* function Smarty::\_run\_mod\_handler(\$modifier\_name, \$map\_array) [line 1711]

**Function Parameters:**

- *string|null* **\$modifier\_name**
- *array|null* **\$map\_array**

## Handle modifiers

*array* function Smarty::\_smarty\_cache\_attrs(\$cache\_serial, \$count) [line 1963]

**Function Parameters:**

- **\$cache\_serial**
- **\$count**

**get or set an array of cached attributes for function that is  
not cacheable**

*void* function Smarty::\_smarty\_include(\$params, \$\_smarty\_include\_tpl\_file, \$\_smarty\_include\_vars) [line 1913]

**Function Parameters:**

- *string* **\$\_smarty\_include\_tpl\_file**
- *string* **\$\_smarty\_include\_vars**
- **\$params**

## called for included templates

*void function Smarty::\_trigger\_fatal\_error(\$error\_msg, [\$tpl\_file = null], [\$tpl\_line = null], [\$file = null], [\$line = null], [\$error\_type = E\_USER\_ERROR]) [line 1873]*

**Function Parameters:**

- *string \$error\_msg*
- *string \$tpl\_file*
- *integer \$tpl\_line*
- *string \$file*
- *integer \$line*
- *integer \$error\_type*

### **trigger Smarty plugin error**

*void function Smarty::\_unlink(\$resource, [\$exp\_time = null]) [line 1836]*

**Function Parameters:**

- *string \$resource*
- *integer \$exp\_time*

### **unlink a file, possibly using expiration time**

## **Class Smarty\_Compiler**

*[line 48]*

### **Template compiling class**

- **Package** Smarty

Constructor *void function Smarty\_Compiler::Smarty\_Compiler() [line 95]*

### **The class constructor.**

*void* function Smarty\_Compiler::\_add\_plugin(\$type, \$name, [\$delayed\_loading = null]) [line 1856]

**Function Parameters:**

- *string* **\$type**
- *string* **\$name**
- *boolean?* **\$delayed\_loading**

## add plugin

*void* function Smarty\_Compiler::\_compile\_arg\_list(\$type, \$name, \$attrs, &\$cache\_code) [line 1306]

**Function Parameters:**

- **\$type**
- **\$name**
- **\$attrs**
- **&\$cache\_code**

*boolean* function Smarty\_Compiler::\_compile\_block\_tag(\$tag\_command, \$tag\_args, \$tag\_modifier, &\$output, \$output) [line 621]

**Function Parameters:**

- *string* **\$tag\_command**
- *string* **\$tag\_args**
- *string* **\$tag\_modifier**
- *string* **\$output**
- **&\$output**

## compile block function tag

sets \$output to compiled block function tag

*string* function Smarty\_Compiler::\_compile\_capture\_tag(\$start, [\$tag\_args = ""]) [line 1116]

**Function Parameters:**

- *boolean* **\$start** true if this is the {capture} tag
- *string* **\$tag\_args**

## Compile {capture} ..{/capture} tags

*boolean* function Smarty\_Compiler::\_compile\_compiler\_tag(\$tag\_command, \$tag\_args, &\$output, \$output) [line 552]

**Function Parameters:**

- *string* **\$tag\_command**
- *string* **\$tag\_args**
- *string* **\$output**
- **&\$output**

### **compile the custom compiler tag**

sets \$output to the compiled custom compiler tag

*string* function Smarty\_Compiler::\_compile\_custom\_tag(\$tag\_command, \$tag\_args, \$tag\_modifier) [line 706]

**Function Parameters:**

- *string* **\$tag\_command**
- *string* **\$tag\_args**
- *string* **\$tag\_modifier**

### **compile custom function tag**

*true* function Smarty\_Compiler::\_compile\_file(\$resource\_name, \$source\_content, &\$compiled\_content, \$compiled\_content) [line 227]

**Function Parameters:**

- *string* **\$resource\_name**
- *string* **\$source\_content**
- *string* **\$compiled\_content**
- **&\$compiled\_content**

### **compile a resource**

sets \$compiled\_content to the compiled source

*string* function Smarty\_Compiler::\_compile\_foreach\_start(\$tag\_args) [line 1050]

**Function Parameters:**

- *string* **\$tag\_args**

## **Compile {foreach ...} tag.**

*string* function Smarty\_Compiler::\_compile\_if\_tag(\$tag\_args, [\$elseif = false]) [line 1151]

**Function Parameters:**

- *string* **\$tag\_args**
- *boolean* **\$elseif** if true, uses elseif instead of if

## **Compile {if ...} tag**

*string* function Smarty\_Compiler::\_compile\_include\_php\_tag(\$tag\_args) [line 914]

**Function Parameters:**

- *string* **\$tag\_args**

## **Compile {include ...} tag**

*string* function Smarty\_Compiler::\_compile\_include\_tag(\$tag\_args) [line 861]

**Function Parameters:**

- *string* **\$tag\_args**

## **Compile {include ...} tag**

*string* function Smarty\_Compiler::\_compile\_insert\_tag(\$tag\_args) [line 827]

**Function Parameters:**

- *string* **\$tag\_args**

## **Compile {insert ...} tag**

*string* function Smarty\_Compiler::\_compile\_plugin\_call(\$type, \$name) [line 1986]

**Function Parameters:**

- *string* **\$type**
- *string* **\$name**

### **compiles call to plugin of type \$type with name \$name**

returns a string containing the function-name or method call without the parameter-list that would have followed to make the call valid php-syntax

*string* function Smarty\_Compiler::\_compile\_registered\_object\_tag(\$tag\_command, \$attrs, \$tag\_modifier) [line 735]

#### **Function Parameters:**

- *string* **\$tag\_command**
- *array* **\$attrs**
- *string* **\$tag\_modifier**

### **compile a registered object tag**

*string* function Smarty\_Compiler::\_compile\_section\_start(\$tag\_args) [line 945]

#### **Function Parameters:**

- *string* **\$tag\_args**

### **Compile {section ...} tag**

*string* function Smarty\_Compiler::\_compile\_smarty\_ref(&\$indexes, \$indexes) [line 1875]

#### **Function Parameters:**

- *string* **\$indexes**
- **&\$indexes**

### **Compiles references of type \$smarty.foo**

*string* function Smarty\_Compiler::\_compile\_tag(\$template\_tag) [line 406]

#### **Function Parameters:**

- *string \$template\_tag*

## Compile a template tag

*string function Smarty\_Compiler::\_expand\_quoted\_text(\$var\_expr) [line 1556]*

**Function Parameters:**

- *string \$var\_expr*

## expand quoted text with embedded variables

*void function Smarty\_Compiler::\_load\_filters() [line 2010]*

**load pre- and post-filters**

*array function Smarty\_Compiler::\_parse\_attrs(\$tag\_args) [line 1407]*

**Function Parameters:**

- *string \$tag\_args*

## Parse attribute string

*void function Smarty\_Compiler::\_parse\_conf\_var(\$conf\_var\_expr) [line 1741]*

**Function Parameters:**

- *string \$conf\_var\_expr*

## parse configuration variable expression into PHP code

*array function Smarty\_Compiler::\_parse\_is\_expr(\$is\_arg, \$tokens) [line 1346]*

**Function Parameters:**

- *string \$is\_arg*
- *array \$tokens*

## **Parse is expression**

*void* function Smarty\_Compiler::\_parse\_modifiers(&\$output, \$modifier\_string, \$output) [line 1787]  
**Function Parameters:**

- *string* **\$output**
- *string* **\$modifier\_string**
- **&\$output**

### **parse modifier chain into PHP code**

sets \$output to parsed modified chain

*string* function Smarty\_Compiler::\_parse\_parenth\_args(\$parenth\_args) [line 1725]

**Function Parameters:**

- *string* **\$parenth\_args**

### **parse arguments in function call parenthesis**

*string* function Smarty\_Compiler::\_parse\_section\_prop(\$section\_prop\_expr) [line 1762]

**Function Parameters:**

- *string* **\$section\_prop\_expr**

### **parse section property expression into PHP code**

*string* function Smarty\_Compiler::\_parse\_var(\$var\_expr, \$output) [line 1582]

**Function Parameters:**

- *string* **\$var\_expr**
- *string* **\$output**

### **parse variable expression into PHP code**

*void* function Smarty\_Compiler::\_parse\_vars\_props(&\$tokens, \$tokens) [line 1491]

**Function Parameters:**

- *array* **\$tokens**
- *&\$tokens*

**compile multiple variables and section properties tokens into PHP code**

*string* function Smarty\_Compiler::\_parse\_var\_props(\$val, \$tag\_attrs) [line 1506]

**Function Parameters:**

- *string* **\$val**
- *string* **\$tag\_attrs**

**compile single variable and section properties token into PHP code**

*string* function Smarty\_Compiler::\_pop\_cacheable\_state(\$type, \$name) [line 2090]

**Function Parameters:**

- **\$type**
- **\$name**

**check if the compilation changes from non-cacheable to cacheable state with the end of the current plugin return php-code to reflect the transition.**

*string* function Smarty\_Compiler::\_push\_cacheable\_state(\$type, \$name) [line 2072]

**Function Parameters:**

- **\$type**
- **\$name**

**check if the compilation changes from cacheable to non-cacheable state with the beginning of the current plugin. return php-code to reflect the transition.**

*string* function Smarty\_Compiler::\_quote\_replace(\$string) [line 2041]

**Function Parameters:**

- *string \$string*

## Quote subpattern references

*void function Smarty\_Compiler::\_\_syntax\_error(\$error\_msg, [\$error\_type = E\_USER\_ERROR], [\$file = null], [\$line = null]) [line 2054]*

### **Function Parameters:**

- *string \$error\_msg*
- *integer \$error\_type*
- *string \$file*
- *integer \$line*

## display Smarty syntax error

## core.assemble\_plugin\_filepath.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string/false* function smarty\_core\_assemble\_plugin\_filepath(\$params, &\$smarty, \$type, \$name) [*line 15*]

#### **Function Parameters:**

- *string* **\$type**
- *string* **\$name**
- **\$params**
- **&\$smarty**

### assemble filepath of requested plugin

## core.assign\_smarty\_interface.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_core\_assign\_smarty\_interface(\$params, &\$smarty) [line 17]*

#### **Function Parameters:**

- **array \$params** Format: null
- **Smarty &\$smarty**

### Smarty assign\_smarty\_interface core plugin

Type: core  
Name: assign\_smarty\_interface  
Purpose: assign the \$smarty interface variable

## core.create\_dir\_structure.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void* function smarty\_core\_create\_dir\_structure(*\$params*, &*\$smarty*, *\$dir*) *[line 16]*

#### **Function Parameters:**

- *string* *\$dir*
- *\$params*
- &*\$smarty*

### create full directory structure

## core.display\_debug\_console.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_core\_display\_debug\_console(\$params, &\$smarty) [line 17]*

#### **Function Parameters:**

- **array \$params** Format: null
- **Smarty &\$smarty**

### Smarty debug\_console function plugin

Type: core  
Name: display\_debug\_console  
Purpose: display the javascript debug console window

## Smarty plugin

### core.get\_include\_path.php

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_get\_include\_path(&\$params, &\$smarty, \$file\_path, \$new\_file\_path) [line 19]

#### **Function Parameters:**

- *string* **\$file\_path**
- *string* **\$new\_file\_path**
- **&\$params**
- **&\$smarty**

## Get path to file from include\_path

- **Static Variable Used** array|null \$\_path\_array:

## core.get\_microtime.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*double* function smarty\_core\_get\_microtime(*\$params*, &*\$smarty*) [*line 12*]

**Function Parameters:**

- **\$params**
- **&\$smarty**

### Get seconds and microseconds

## Smarty plugin

### core.get\_php\_resource.php

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_get\_php\_resource(&\$params, &\$smarty, \$resource, \$resource\_type, 2) [line 18]

#### **Function Parameters:**

- *string* **\$resource**
- *string* **\$resource\_type**
- **\$php\_resource** 2
- **&\$params**
- **&\$smarty**

#### **Retrieves PHP script resource**

sets \$php\_resource to the returned resource

## core.is\_secure.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_is\_secure(\$params, &\$smarty, \$resource\_type, \$resource\_name) [line 18]

#### **Function Parameters:**

- *string* **\$resource\_type**
- *string* **\$resource\_name**
- **\$params**
- **&\$smarty**

**determines if a resource is secure or not.**

## core.is\_trusted.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_is\_trusted(\$params, &\$smarty, \$resource\_type, \$resource\_name) [line 21]

#### **Function Parameters:**

- *string* **\$resource\_type**
- *string* **\$resource\_name**
- **\$params**
- **&\$smarty**

**determines if a resource is trusted or not**

## core.load\_plugins.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_core\_load\_plugins(\$params, &\$smarty, \$plugins) [line 16]*

#### **Function Parameters:**

- *array \$plugins*
- *\$params*
- *&\$smarty*

### Load requested plugins

## core.load\_resource\_plugin.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_core\_load\_resource\_plugin(\$params, &\$smarty, \$type) [line 16]*

#### **Function Parameters:**

- *string* **\$type**
- **\$params**
- **&\$smarty**

### load a resource plugin

## core.process\_cached\_inserts.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_core\_process\_cached\_inserts(\$params, &\$smarty, \$results) [*line 14*]

#### **Function Parameters:**

- *string* **\$results**
- **\$params**
- **&\$smarty**

### Replace cached inserts with the actual results

## Smarty plugin

### core.process\_compiled\_include.php

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_core\_process\_compiled\_include(*\$params*, &&*\$smarty*, *\$compiled\_tpl*, *\$cached\_source*)  
[line 17]

#### **Function Parameters:**

- *string* **\$compiled\_tpl**
- *string* **\$cached\_source**
- **\$params**
- **&\$smarty**

**Replace nocache-tags by results of the corresponding non-cacheable functions and return it**

## core.read\_cache\_file.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_read\_cache\_file(&\$params, &\$smarty, \$tpl\_file, \$cache\_id, \$compile\_id, \$results) [line 21]

#### **Function Parameters:**

- *string* **\$tpl\_file**
- *string* **\$cache\_id**
- *string* **\$compile\_id**
- *string* **\$results**
- **&\$params**
- **&\$smarty**

**read a cache file, determine if it needs to be regenerated or not**

## core.rmdir.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_rmdir(\$params, &\$smarty, \$dirname, \$level, \$exp\_time) [*line 20*]

#### **Function Parameters:**

- *string* **\$dirname**
- *integer* **\$level**
- *integer* **\$exp\_time**
- **\$params**
- **&\$smarty**

**delete a dir recursively (level=0 -> keep root) WARNING: no tests, it will try to remove what you tell it!**

## core.rm\_auto.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_rm\_auto(\$params, &\$smarty, \$auto\_base, \$auto\_source, \$auto\_id, \$exp\_time)  
[line 20]

#### **Function Parameters:**

- *string* **\$auto\_base**
- *string* **\$auto\_source**
- *string* **\$auto\_id**
- *integer* **\$exp\_time**
- **\$params**
- **&\$smarty**

**delete an automagically created file by name and id**

## core.run\_insert\_handler.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_core\_run\_insert\_handler(*\$params*, &*\$smarty*, *\$args*) [line 14]

**Function Parameters:**

- *array \$args*
- *\$params*
- *&\$smarty*

### Handle insert tags

## core.smarty\_include\_php.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_core\_smarty\_include\_php(\$params, &\$smarty, \$smarty\_file, \$smarty\_assign, \$smarty\_once, \$smarty\_include\_vars) [line 21]*

#### **Function Parameters:**

- **string \$smarty\_file**
- **string \$smarty\_assign** variable to assign the included template's output into
- **boolean \$smarty\_once** uses include\_once if this is true
- **array \$smarty\_include\_vars** associative array of vars from {include file="blah" var=\$var}
- **\$params**
- **&\$smarty**

### called for included php files within templates

## core.write\_cache\_file.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*true|**null* function smarty\_core\_write\_cache\_file(\$params, &\$smarty, \$tpl\_file, \$cache\_id, \$compile\_id, \$results) [line 21]

#### **Function Parameters:**

- *string* **\$tpl\_file**
- *string* **\$cache\_id**
- *string* **\$compile\_id**
- *string* **\$results**
- **\$params**
- **&\$smarty**

**Prepend the cache information to the cache file and write it**

## Smarty plugin

### core.write\_compiled\_include.php

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_write\_compiled\_include(\$params, &\$smarty, \$compile\_path, \$template\_compiled, \$template\_timestamp) [line 17]

#### **Function Parameters:**

- *string* **\$compile\_path**
- *string* **\$template\_compiled**
- *integer* **\$template\_timestamp**
- **\$params**
- **&\$smarty**

**Extract non-cacheable parts out of compiled template and write it**

## core.write\_compiled\_resource.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

true function smarty\_core\_write\_compiled\_resource(\$params, &\$smarty, \$compile\_path, \$compiled\_content, \$resource\_timestamp) [line 16]

#### **Function Parameters:**

- *string* **\$compile\_path**
- *string* **\$compiled\_content**
- *integer* **\$resource\_timestamp**
- **\$params**
- **&\$smarty**

### write the compiled resource

## core.write\_file.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*boolean* function smarty\_core\_write\_file(\$params, &\$smarty, \$filename, \$contents, \$create\_dirs) [line 16]

#### **Function Parameters:**

- *string* **\$filename**
- *string* **\$contents**
- *boolean* **\$create\_dirs**
- **\$params**
- **&\$smarty**

### write out a file to disk

# block.strip.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_block\_strip(\$params, \$content, &\$this) [line 21]

### **Function Parameters:**

- *array* **\$params** unused, no parameters for this block
- *string* **\$content** content of {strip}{/strip} tags
- [Smarty](#) **&\$this** clever method emulation

## Smarty {strip}{/strip} block plugin

Type: block function

Name: strip

Purpose: strip unwanted white space from text

- [Link {strip}](#)  
[\(Smarty online manual\)](#)

# block.textformat.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_block\_textformat(\$params, \$content, &\$smarty) [line 30]

### **Function Parameters:**

- **array \$params** Params: style: string (email)  
indent: integer (0)  
wrap: integer (80)  
wrap\_char string ("\n")  
indent\_char: string (" ")  
wrap\_boundary: boolean (true)
- **string \$content** contents of the block
- **Smarty &\$smarty** clever simulation of a method

## Smarty {textformat}{/textformat} block plugin

Type: block function

Name: textformat

Purpose: format text a certain way with preset styles  
settings

or custom wrap/indent

- **Link {textformat}**  
**(Smarty online manual)**

# function.assign.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_function\_assign(\$params, &\$smarty) [line 19]*

### **Function Parameters:**

- **array \$params** Format: array('var' => variable name, 'value' => value to assign)
- **Smarty &\$smarty**

## Smarty {assign} function plugin

Type: function  
Name: assign  
Purpose: assign a value to a template variable

- **Link {assign}**  
(Smarty online manual)

## function.assign\_debug\_info.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_function\_assign\_debug\_info(\$params, &\$smarty) [line 18]*

#### **Function Parameters:**

- *array \$params* unused in this plugin, this plugin uses Smarty::\$\_config, Smarty::\$\_tpl\_vars and Smarty::\$\_smarty\_debug\_info
- *Smarty &\$smarty*

### Smarty {assign\_debug\_info} function plugin

Type: function

Name: assign\_debug\_info

Purpose: assign debug info to the template

## function.config\_load.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_function\_config\_load(\$params, &\$smarty) [line 25]*

#### **Function Parameters:**

- **array \$params** Format: array('file' => required config file name,  
'section' => optional config file section to load  
'scope' => local/parent/global  
'global' => overrides scope, setting to parent if true)
- [Smarty](#) &\$smarty

### Smarty {config\_load} function plugin

Type: function  
Name: config\_load  
Purpose: load config file vars

- [Link {config\\_load}](#)  
[\(Smarty online manual\)](#)

# function.counter.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string|null* function smarty\_function\_counter(\$params, &\$smarty) [line 21]

### **Function Parameters:**

- **array \$params** parameters
- **[Smarty](#) &\$smarty**

## Smarty {counter} function plugin

Type: function

Name: counter

Purpose: print out a counter value

- [Link {counter}](#)  
[\(Smarty online manual\)](#)

# function.cycle.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string|null* function smarty\_function\_cycle(\$params, &\$smarty) [line 44]

### **Function Parameters:**

- **array \$params**
- **[Smarty](#) &\$smarty**

## Smarty {cycle} function plugin

Type: function

Name: cycle

Date: May 3, 2002

Purpose: cycle through given values

Input:

- name = name of cycle (optional)
- values = comma separated list of values to cycle,  
or an array of values to cycle  
(this can be left out for subsequent calls)
- reset = boolean - resets given var to true
- print = boolean - print var or not. default is true
- advance = boolean - whether or not to advance the cycle
- delimiter = the value delimiter, default is ","
- assign = boolean, assigns to template var instead of  
printed.

Examples:

```
{cycle values="#eeeeee,#d0d0d0d"}  
{cycle name=row values="one,two,three" reset=true}  
{cycle name=row}
```

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Author** credit to Mark Priatel < [mpriatel@rogers.com](mailto:mpriatel@rogers.com)>
- **Author** credit to Gerard < [gerard@interfold.com](mailto:gerard@interfold.com)>

- **Author** credit to Jason Sweat < [jsweat\\_php@yahoo.com](mailto:jsweat_php@yahoo.com)>
- **Version** 1.3
- **Link {cycle}**  
[\(Smarty online manual\)](#)

# function.debug.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_debug(*\$params*, &&*\$smarty*) [line 24]

### **Function Parameters:**

- *array* **\$params**
- *Smarty* &**\$smarty**

## Smarty {debug} function plugin

Type: function  
Name: debug  
Date: July 1, 2002  
Purpose: popup debug window

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.0
- **Link** [{debug}](#)  
[\(Smarty online manual\)](#)

# function.eval.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_function\_eval(\$params, &\$smarty) [line 20]*

### **Function Parameters:**

- **array \$params**
- **[Smarty](#) &\$smarty**

## Smarty {eval} function plugin

Type: function

Name: eval

Purpose: evaluate a template variable as a template

- [Link {eval}](#)  
[\(Smarty online manual\)](#)

# function.fetch.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string|null* function smarty\_function\_fetch(\$params, &\$smarty) [line 22]

**Function Parameters:**

- **array \$params**
- **[Smarty](#) &\$smarty**

## Smarty {fetch} plugin

Type: function

Name: fetch

Purpose: fetch file, web or ftp data and display results

- [Link {fetch}](#)  
[\(Smarty online manual\)](#)

# function.html\_checkboxes.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_checkboxes(*\$params*, &*\$smarty*) [line 40]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &\$smarty*

## Smarty {html\_checkboxes} function plugin

File: function.html\_checkboxes.php

Type: function

Name: html\_checkboxes

Date: 24.Feb.2003

Purpose: Prints out a list of checkbox input types

Input:

- name (optional) - string default "checkbox"
  - values (required) - array
  - options (optional) - associative array
  - checked (optional) - array default not set
  - separator (optional) - ie  
or &nbsp;
  - output (optional) - without this one the buttons don't have names
- Examples: {html\_checkboxes values=\$ids output=\$names}  
{html\_checkboxes values=\$ids name='box' separator='<br>' output=\$names}  
{html\_checkboxes values=\$ids checked=\$checked separator='<br>' output=\$names}

- **Author** credits to Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Author** Christopher Kvarme < [christopher.kvarme@flashjab.com](mailto:christopher.kvarme@flashjab.com)>
- **Version** 1.0
- **Link** [{html\\_checkboxes}](#)  
[\(Smarty online manual\)](#)

- **Uses** [smarty\\_function\\_escape\\_special\\_chars\(\)](#)

*void function smarty\_function\_html\_checkboxes\_output(\$name, \$value, \$output, \$selected, \$extra, \$separator, \$labels) [line 118]*

**Function Parameters:**

- **\$name**
- **\$value**
- **\$output**
- **\$selected**
- **\$extra**
- **\$separator**
- **\$labels**

# function.html\_image.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_image(*\$params*, &&*\$smarty*) [line 37]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &&\$smarty*

## Smarty {html\_image} function plugin

Type: function

Name: html\_image

Date: Feb 24, 2003

Purpose: format HTML tags for the image

Input:

- file = file (and path) of image (required)
- border = border width (optional, default 0)
- height = image height (optional, default actual height)
- image =image width (optional, default actual width)
- basedir = base directory for absolute paths, default  
is environment variable DOCUMENT\_ROOT

Examples: {html\_image file="images/masthead.gif"} Output: 

- **Author** credits to Duda < [duda@big.hu](mailto:duda@big.hu)> - wrote first image function in repository, helped with lots of functionality
- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.0
- **Link** [{html\\_image}](#)  
[\(Smarty online manual\)](#)
- **Uses** [smarty\\_function\\_escape\\_special\\_chars\(\)](#)



# function.html\_options.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_options(*\$params*, &*\$smarty*) [line 29]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &\$smarty*

## Smarty {html\_options} function plugin

Type: function  
Name: html\_options  
Input:

- name (optional) - string default "select"
- values (required if no options supplied) - array
- options (required if no values supplied) - associative array
- selected (optional) - string default not set
- output (required if not options supplied) - array

Purpose: Prints the list of <option> tags generated from parameters

the passed

- [Link {html\\_image}](#)  
[\(Smarty online manual\)](#)
- [Uses smarty\\_function\\_escape\\_special\\_chars\(\)](#)

*void* function smarty\_function\_html\_options\_optgroup(*\$key*, *\$values*, *\$selected*) [line 107]

### **Function Parameters:**

- *\$key*

- **\$values**
- **\$selected**

*void function smarty\_function\_html\_options\_optoutput(\$key, \$value, \$selected) [line 94]*

**Function Parameters:**

- **\$key**
- **\$value**
- **\$selected**

# function.html\_radios.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_radios(*\$params*, &*\$smarty*) [line 40]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &\$smarty*

## Smarty {html\_radios} function plugin

File: function.html\_radios.php

Type: function

Name: html\_radios

Date: 24.Feb.2003

Purpose: Prints out a list of radio input types

Input:

- name (optional) - string default "radio"
- values (required) - array
- options (optional) - associative array
- checked (optional) - array default not set
- separator (optional) - ie  
or &nbsp;
- output (optional) - without this one the buttons don't have names

Examples: {html\_radios values=\$ids output=\$names}

{html\_radios values=\$ids name='box' separator='<br>' output=\$names}

{html\_radios values=\$ids checked=\$checked separator='<br>' output=\$names}

- **Author** credits to Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Author** Christopher Kvarme < [christopher.kvarme@flashjab.com](mailto:christopher.kvarme@flashjab.com)>
- **Version** 1.0
- **Link** [{html\\_radios}](#)  
[\(Smarty online manual\)](#)

- **Uses** [smarty\\_function\\_escape\\_special\\_chars\(\)](#)

*void function smarty\_function\_html\_radios\_output(\$name, \$value, \$output, \$selected, \$extra, \$separator, \$labels) [line 121]*

**Function Parameters:**

- **\$name**
- **\$value**
- **\$output**
- **\$selected**
- **\$extra**
- **\$separator**
- **\$labels**

# function.html\_select\_date.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_select\_date(*\$params*, &*\$\$smarty*) [*line 33*]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &\$\$smarty*

## Smarty {html\_select\_date} plugin

Type: function  
Name: html\_select\_date  
Purpose: Prints the dropdowns for date selection.

### ChangeLog:

- 1.0 initial release
- 1.1 added support for +/- N syntax for begin and end year values. (Monte)
- 1.2 added support for yyyy-mm-dd syntax for time value. (Jan Rosier)
- 1.3 added support for choosing format for month values (Gary Loescher)
- 1.3.1 added support for choosing format for day values (Marcus Bointon)

- **Author** Andrei Zmievski
- **Version** 1.3
- **Link** [{html\\_select\\_date}](#)  
[\(Smarty online manual\)](#)



## function.html\_select\_time.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_select\_time(*\$params*, &*\$\$smarty*) [line 22]

#### **Function Parameters:**

- *array* **\$params**
- [Smarty](#) &*\$\$smarty*

### Smarty {html\_select\_time} function plugin

Type: function  
Name: html\_select\_time  
Purpose: Prints the dropdowns for time selection

- [Link {html\\_select\\_time}](#)  
[\(Smarty online manual\)](#)
- [Uses smarty\\_make\\_timestamp\(\)](#)

# function.html\_table.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_html\_table(*\$params*, &*\$smarty*) [line 44]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &\$smarty*

## Smarty {html\_table} function plugin

Type: function

Name: html\_table

Date: Feb 17, 2003

Purpose: make an html table from an array of data

Input:

- loop = array to loop through
- cols = number of columns
- rows = number of rows
- table\_attr = table attributes
- tr\_attr = table row attributes (arrays are cycled)
- td\_attr = table cell attributes (arrays are cycled)
- trailpad = value to pad trailing cells with
- vdir = vertical direction (default: "down", means top-to-bottom)
- hdir = horizontal direction (default: "right", means left-to-right)
- inner = inner loop (default "cols": print \$loop line by line,  
\$loop will be printed column by column otherwise)

Examples: {table loop=\$data}  
{table loop=\$data cols=4 tr\_attr="bgcolor=red"}  
{table loop=\$data cols=4 tr\_attr=\$colors}

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.0
- **Link** [{html\\_table}](#)  
[\(Smarty online manual\)](#)

*void function smarty\_function\_html\_table\_cycle(\$name, \$var, \$no) [line 100]*

**Function Parameters:**

- **\$name**
- **\$var**
- **\$no**

# function.mailto.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_mailto(*\$params*, &*\$smarty*) [line 49]

### **Function Parameters:**

- *array \$params*
- *[Smarty](#) &\$smarty*

## Smarty {mailto} function plugin

Type: function

Name: mailto

Date: May 21, 2002 Purpose: automate mailto address link creation, and optionally encode them.

Input:

- address = e-mail address
- text = (optional) text to display, default is address
- encode = (optional) can be one of:
  - \* none : no encoding (default)
  - \* javascript : encode with javascript
  - \* hex : encode with hexadecimal (no javascript)
- cc = (optional) address(es) to carbon copy
- bcc = (optional) address(es) to blind carbon copy
- subject = (optional) e-mail subject
- newsgroups = (optional) newsgroup(s) to post to
- followupto = (optional) address(es) to follow up to
- extra = (optional) extra tags for the href link

Examples: {mailto address="me@domain.com"}  
{mailto address="me@domain.com" encode="javascript"}  
{mailto address="me@domain.com" encode="hex"}  
{mailto address="me@domain.com" subject="Hello to you!"}  
{mailto address="me@domain.com" cc="you@domain.com,they@domain.com"}  
{mailto address="me@domain.com" extra='class="mailto"'}

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Author** credits to Jason Sweat (added cc, bcc and subject functionality)
- **Version** 1.2
- **Link {mailto}**  
[\(Smarty online manual\)](#)

# function.math.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_math(*\$params*, &*\$smarty*) [*line 21*]

### **Function Parameters:**

- *array* **\$params**
- *Smarty* &**\$smarty**

## Smarty {math} function plugin

Type: function  
Name: math  
Purpose: handle math computations in template

- [Link {math}](#)  
[\(Smarty online manual\)](#)

# function.popup.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_popup(\$params, &\$smarty) [line 21]

**Function Parameters:**

- *array* \$params
- [Smarty](#) &\$smarty

## Smarty {popup} function plugin

Type: function

Name: popup

Purpose: make text pop up in windows via overlib

- [Link {popup}](#)  
[\(Smarty online manual\)](#)

## function.popup\_init.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_popup\_init(*\$params*, &*\$smarty*) [line 21]

**Function Parameters:**

- *array* **\$params**
- [Smarty](#) &*\$smarty*

### Smarty {popup\_init} function plugin

Type: function

Name: popup\_init

Purpose: initialize overlib

- [Link {popup\\_init}](#)  
[\(Smarty online manual\)](#)

## function.var\_dump.php

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_function\_var\_dump(\$params, &\$smarty) [line 13]*

**Function Parameters:**

- **\$params**
- **&\$smarty**

### Smarty plugin

----- Type: function Name: assign  
Purpose: assign a value to a template variable -----  
-----

# modifier.capitalize.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_capitalize(\$string) [line 20]

**Function Parameters:**

- *string* **\$string**

## Smarty capitalize modifier plugin

Type: modifier

Name: capitalize

Purpose: capitalize words in the string

- [Link capitalize \(Smarty online manual\)](#)

# modifier.cat.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_cat(\$string, \$cat) [*line 26*]

### **Function Parameters:**

- *string* **\$string**
- *string* **\$cat**

## Smarty cat modifier plugin

Type: modifier

Name: cat

Date: Feb 24, 2003 Purpose: catenate a value to a variable Input: string to  
catenate Example: {\$var|cat:"foo"}

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.0
- **Link** [cat](#)  
[\(Smarty online manual\)](#)

## modifier.count\_characters.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*integer* function smarty\_modifier\_count\_characters(\$string, [\$include\_spaces = false]) [line 21]

#### **Function Parameters:**

- *string* **\$string**
- *boolean* **\$include\_spaces** include whitespace in the character count

### Smarty count\_characters modifier plugin

Type: modifier  
Name: count\_characters  
Purpose: count the number of characters in a text

- [Link count\\_characters \(Smarty online manual\)](#)

## modifier.count\_paragraphs.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*integer* function smarty\_modifier\_count\_paragraphs(\$string) [line 20]

**Function Parameters:**

- *string* \$string

### Smarty count\_paragraphs modifier plugin

Type: modifier

Name: count\_paragraphs

Purpose: count the number of paragraphs in a text

- [Link count\\_paragraphs \(Smarty online manual\)](#)

## modifier.count\_sentences.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*integer* function smarty\_modifier\_count\_sentences(\$string) [line 20]

**Function Parameters:**

- *string* \$string

### Smarty count\_sentences modifier plugin

Type: modifier  
Name: count\_sentences Purpose: count the number of sentences in a text

- [Link count\\_sentences \(Smarty online manual\)](#)

## modifier.count\_words.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*integer* function smarty\_modifier\_count\_words(\$string) [line 20]

**Function Parameters:**

- *string* \$string

### Smarty count\_words modifier plugin

Type: modifier  
Name: count\_words  
Purpose: count the number of words in a text

- [Link count\\_words \(Smarty online manual\)](#)

# modifier.date\_format.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

```
require_once $smarty->_get_plugin_filepath('shared','make_timestamp') [line 11]
```

### Include the [shared.make\\_timestamp.php](#) plugin

*string/void* function smarty\_modifier\_date\_format(\$string, [\$format = "%b %e, %Y"], [\$default\_date = null]) [line 30]

#### Function Parameters:

- *string* **\$string**
- *string* **\$format**
- *string* **\$default\_date**

## Smarty date\_format modifier plugin

Type: modifier  
Name: date\_format  
Purpose: format datestamps via strftime  
Input:

- string: input date string
- format: strftime format for output
- default\_date: default date if \$string is empty

- [Link date\\_format \(Smarty online manual\)](#)
- [Uses smarty\\_make\\_timestamp\(\)](#)

## modifier.debug\_print\_var.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_debug\_print\_var(\$var, [\$depth = 0], [\$length = 40]) [line 22]

#### **Function Parameters:**

- *array/object* \$var
- *integer* \$depth
- *integer* \$length

### Smarty debug\_print\_var modifier plugin

Type: modifier

Name: debug\_print\_var

Purpose: formats variable contents for display in the console

- [Link debug\\_print\\_var \(Smarty online manual\)](#)

# modifier.default.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_default(*\$string*, [*\$default* = "]) [line 21]

**Function Parameters:**

- *string* **\$string**
- *string* **\$default**

## Smarty default modifier plugin

Type: modifier

Name: default

Purpose: designate default value for empty variables

- [Link default \(Smarty online manual\)](#)

# modifier.escape.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_escape(\$string, [\$esc\_type = 'html']) [line 21]

### **Function Parameters:**

- *string* **\$string**
- *html|htmlall|url|quotes|hex|hexentity|javascript* **\$esc\_type**

## Smarty escape modifier plugin

Type: modifier  
Name: escape  
Purpose: Escape the string according to escapement type

- [Link escape \(Smarty online manual\)](#)

# modifier.htmlentities.php

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_modifier\_htmlentities(\$string) [line 13]*

**Function Parameters:**

- **\$string**

## Smarty plugin

Purpose: convert string to uppercase ----- Type: modifier Name: upper

# modifier.indent.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_indent(\$string, [\$chars = 4], [\$char = " "]) *[line 22]*

### **Function Parameters:**

- *string* **\$string**
- *integer* **\$chars**
- *string* **\$char**

## Smarty indent modifier plugin

Type: modifier

Name: indent

Purpose: indent lines of text

- [Link indent \(Smarty online manual\)](#)

# modifier.lower.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_lower(\$string) [line 20]

**Function Parameters:**

- *string* **\$string**

## Smarty lower modifier plugin

Type: modifier

Name: lower

Purpose: convert string to lowercase

- [Link lower \(Smarty online manual\)](#)

# modifier.nl2br.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_nl2br(\$string) [line 28]

**Function Parameters:**

- *string* **\$string**

## Smarty plugin

Type: modifier

Name: nl2br

Date: Feb 26, 2003 Purpose: convert \r\n, \r or \n to <br> Input:

- contents = contents to replace
- preceed\_test = if true, includes preceding break tags in replacement

Example: {\$text|nl2br}

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.0
- **Link** [nl2br \(Smarty online manual\)](#)

# modifier.rawurlencode.php

- **Package** Smarty
- **Sub-Package** plugins

*void function smarty\_modifier\_rawurlencode(\$string) [line 13]*

**Function Parameters:**

- **\$string**

## Smarty plugin

----- Type: modifier Name:  
rawurlencode Purpose: encode string for use in PDFdefaultConverter TOC -----  
-----

# modifier.regex\_replace.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_regex\_replace(\$string, \$search, \$replace) [*line 22*]

**Function Parameters:**

- *string* **\$string**
- *string|array* **\$search**
- *string|array* **\$replace**

## Smarty regex\_replace modifier plugin

Type: modifier

Name: regex\_replace

Purpose: regular expression search/replace

- [Link regex\\_replace \(Smarty online manual\)](#)

# modifier.replace.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_replace(\$string, \$search, \$replace) [line 22]

**Function Parameters:**

- *string* **\$string**
- *string* **\$search**
- *string* **\$replace**

## Smarty replace modifier plugin

Type: modifier

Name: replace

Purpose: simple search/replace

- [Link replace \(Smarty online manual\)](#)

# modifier.spacify.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_spacify(\$string, [\$spacify\_char = ' ']) [line 21]

### **Function Parameters:**

- *string* **\$string**
- *string* **\$spacify\_char**

## Smarty spacify modifier plugin

Type: modifier

Name: spacify

Purpose: add spaces between characters in a string

- [Link spacify \(Smarty online manual\)](#)

# modifier.string\_format.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_string\_format(\$string, \$format) [*line 21*]

**Function Parameters:**

- *string* **\$string**
- *string* **\$format**

## Smarty string\_format modifier plugin

Type: modifier  
Name: string\_format  
Purpose: format strings via sprintf

- [Link string\\_format \(Smarty online manual\)](#)

# modifier.strip.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_strip(\$text, [\$replace = ' ']) [line 26]

### **Function Parameters:**

- *string* **\$text**
- *string* **\$replace**

## Smarty strip modifier plugin

Type: modifier

Name: strip

Purpose: Replace all repeated spaces, newlines, tabs with a single space or supplied replacement string.

Example: {\$var|strip} {\$var|strip:" "} Date: September 25th, 2002

- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.0
- **[Link strip \(Smarty online manual\)](#)**

## modifier.strip\_tags.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_strip\_tags(\$string, [\$replace\_with\_space = true]) [line 21]

#### **Function Parameters:**

- *string* **\$string**
- *boolean* **\$replace\_with\_space**

### Smarty strip\_tags modifier plugin

Type: modifier  
Name: strip\_tags  
Purpose: strip html tags from text

- [Link strip\\_tags \(Smarty online manual\)](#)

# modifier.truncate.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_truncate(\$string, [\$length = 80], [\$etc = '...'], [\$break\_words = false]) [line 25]

### **Function Parameters:**

- *string* **\$string**
- *integer* **\$length**
- *string* **\$etc**
- *boolean* **\$break\_words**

## Smarty truncate modifier plugin

Type: modifier

Name: truncate

Purpose: Truncate a string to a certain length if necessary, optionally splitting in the middle of a word, and appending the \$etc string.

- [Link truncate \(Smarty online manual\)](#)

# modifier.upper.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_upper(*\$string*) [*line 20*]

**Function Parameters:**

- *string* **\$string**

## Smarty upper modifier plugin

Type: modifier

Name: upper

Purpose: convert string to uppercase

- [Link upper \(Smarty online manual\)](#)

## modifier.wordwrap.php

### Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_modifier\_wordwrap(\$string, [\$length = 80], [\$break = "\n"], [\$cut = false])  
[line 23]

#### **Function Parameters:**

- *string* **\$string**
- *integer* **\$length**
- *string* **\$break**
- *boolean* **\$cut**

### Smarty wordwrap modifier plugin

Type: modifier  
Name: wordwrap  
Purpose: wrap a string of text at a given length

- [Link wordwrap \(Smarty online manual\)](#)

# outputfilter.trimwhitespace.php

## Smarty plugin

- **Package** Smarty
- **Sub-Package** plugins

void function smarty\_outputfilter\_trimwhitespace(\$source, &\$smarty) [line 28]

### Function Parameters:

- *string* **\$source**
- *Smarty* &**\$smarty**

## Smarty trimwhitespace outputfilter plugin

File: outputfilter.trimwhitespace.php

Type: outputfilter

Name: trimwhitespace

Date: Jan 25, 2003

Purpose: trim leading white space and blank lines from template source after it gets interpreted, cleaning up code and saving bandwidth. Does not affect <<PRE>></PRE> and <SCRIPT></SCRIPT> blocks.

Install: Drop into the plugin directory, call

1    \$smarty-> load\_filter('output','trimwhitespace');  
      from application.

- **Author** Contributions from Lars Noschinski < [lars@usenet.noschinski.de](mailto:lars@usenet.noschinski.de)>
- **Author** Monte Ohrt < [monte@ispi.net](mailto:monte@ispi.net)>
- **Version** 1.3

void function smarty\_outputfilter\_trimwhitespace\_replace(\$search\_str, \$replace, &\$subject) [line 64]

### Function Parameters:

- **\$search\_str**
- **\$replace**
- **&\$subject**



# shared.escape\_special\_chars.php

## Smarty shared plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_function\_escape\_special\_chars(\$string) [line 18]

**Function Parameters:**

- *string* **\$string**

### escape\_special\_chars common function

Function: smarty\_function\_escape\_special\_chars

Purpose: used by other smarty functions to escape already escaped ones

special chars except for

- **Usedby** [smarty\\_function\\_html\\_checkboxes\(\)](#)
- **Usedby** [smarty\\_function\\_html\\_image\(\)](#)
- **Usedby** [smarty\\_function\\_html\\_options\(\)](#)
- **Usedby** [smarty\\_function\\_html\\_radios\(\)](#)

# shared.make\_timestamp.php

## Smarty shared plugin

- **Package** Smarty
- **Sub-Package** plugins

*string* function smarty\_make\_timestamp(\$string) [line 16]

**Function Parameters:**

- *string* \$string

### Function: smarty\_make\_timestamp

**Purpose:** used by other smarty functions to make a timestamp

from a

string.

- **Usedby** [smarty\\_modifier\\_date\\_format\(\)](#)
- **Usedby** [smarty\\_function\\_html\\_select\\_time\(\)](#)



# Package tests Procedural Elements

## bug-441275.php

- **Package** tests

```
void function test2_441275([&#03$foo = array("item1","item2",'item3' =>
"item4",array("item5","item6"),'item7'=>array('item8','item9','array('it
em9','item10')))], $foobar) [/line 13]
```

**Function Parameters:**

- **\$foo**
- **\$foobar**

**This tests some more advanced cases to make sure to handle them**

```
void function test_441275([&#03$foo = array()]) [/line 6]
```

**Function Parameters:**

- **\$foo**

**This makes sure the basic element of bug 44127 is fixed**

# bug-441278.php

- **Package** tests

array function test\_441278() [line 14]

## **Test description.**

anything said here

# bug-441287.php

- **Package** tests

```
void function test_441287() [line 12]
```

```
sdesc
1234567890
Test
Test2
```

```
void function test_4412872() [line 25]
```

```
Check to see if we are stripping whitespace before the *
1234567890
Test
Test2
```

# bug-441289.php

- Package tests

*void function test6() [line 58]*

**This desc**

tries to fool the non period short desc systems

junk should only show the first line since we dont' look that far down for the short desc line break

*void function test7() [line 69]*

**This desc tries to fool the non period short desc systems**

junk junk

*void function test8() [line 90]*

**This is a test case where i think things break**

extended desc many lines of desc more extended desc that ends with a period. without a line limiter everything til period. would be part of the short desc allowing this many lines to be part of the short desc seemed like a problem to me so now using both the period and line break as a separator you can only have a 4 line extended desc. the first line is usually blank since its the /\*\* line

*void function test\_441289() [line 9]*

**This is a long comment which needs more than one line.**

This is the detailed documentation. This is another line of detailed docs

*void function test\_4412892() [line 17]*

**I started my short desc on the first line.**

This is the detailed documentation.

*void function test\_4412893() [line 25]*

**I am using just the first line as my short desc**

since i didn't use a period anywhere in this desc

`void function test_4412894() [line 35]`

**I am using a blank line to end my short desc**

I think this looks the nicest when writing your code and makes the code more readable

`void function test_4412895() [line 44]`

**This is a test to see if the desc is fooled by 2.4 the short desc ends here.**

This is the detailed documentation.

`void function test_bug_567455() [line 100]`

**This example is really tricky**

The short desc should end on the blank line even though this line ends with a period.  
Here is the long desc

`void function test_bug_567455_2() [line 111]`

**This example is also really tricky**

The tries to break the parser test. Here is the long desc

# bug-441433.php

- **Package** tests

```
void function SendEMail([$sendto = "\"Your Name\""  
<your@email.here>"]) [line 6]
```

**Function Parameters:**

- **\$sendto**

## This is a Test

```
void function test_441433([$test = "Some stuff in a quote"]) [line 16]
```

**Function Parameters:**

- **\$test**

## a second test.

Make sure that we are clearing out quote data after using it since how we set that values has changed

# bug-443153.php

- **Package** tests

*void function test\_443153() [line 8]*

## **test function**

- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Test Person < [tperson@test.net](mailto:tperson@test.net)>

# bug-445298.php

- **Package** tests

```
void function test_445298() [line 9]
test function
```

- **See** link2
- **See** link1

# bug-445305.php

- **Package** tests

*void* function test\_445305(*\$foo*) [line 8]

**Function Parameters:**

- *String \$foo* A string to use in a function, if you really want to.

# bug-445820.php

- **Package** tests

*void function test\_445820([**\$val** = array("1","2",'3' => 4)], **\$string**, **\$my\_array**) [line 5]*

**Function Parameters:**

- **\$val**
- **\$string**
- **\$my\_array**

## test function

# bug-540368.php

- **Package** tests

*void function blah() [line 20]*

# bug-542586.php

- **Package** tests

```
void function func() [line 11]
testie = 1 [line 4]
testie
```

# bug-550489.php

- **Package** tests

thisisdumbbutworks = 'I'.parse this' [*line 8*]

!

# bug-551120.php

- **Package** tests

include ('test'.'me').'.php' [*line 6*]

**include**

# bug-553137.php

- **Package** tests

*void* function func1() [*line 8*]

**func 1**

*void* function func2() [*line 12*]

*void* function func3() [*line 19*]

- See [func1\(\)](#), [func2\(\)](#)

# bug-554712.php

- **Package** tests

*void function passbyref(&\$varname) [line 15]*

**Function Parameters:**

- **&\$varname**

**passes a variable by reference**

*void function returnsme() [line 8]*

**returns a reference**

## bug-557861.php

**bug is fixed. to test, remove this page-level docblock**

- **Package** tests

# bug-558031.php

- **Package** tests

*void function test\_558031() [line 10]*

- **Access** public

# bug-558051.php

- **Package** tests

*void function one() [line 11]*

- **Access** public

# bug-559467.php

- **Package** tests

CONSTANT1 = 1 [*line 8*]

**blah blah short description**  
blah blah long description

- See [CONSTANT2](#)

CONSTANT2 = 2 [*line 15*]

**blah blah short description**  
blah blah long description

- See [CONSTANT1](#)

# **bug-559494.php**

**tests variable names with the word 'array' in them**

- **Package** tests

# bug-559668.php

- **Package** tests

include **PEAR.'test'.'me'** [line 8]

include '**file.ext**' [line 7]

include '**file.ext**' [line 6]

require **PEAR.'test'.'me'** [line 5]

## **include** tests

# Config

## Config file for MLib

File includes all of the static variables that controls the default behavior of MLib.

- **Package** tests
- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- **Version** 0.1
- **See** MLib
- **Access** public
- **Name** Config

### **\$\_MLIB\_GLOBAL\_DSN**

```
array = array('db_type'=> 'mysql',
              'username' => 'nobody',
              'password' => '',
              'database' => 'miester',
              'server' => 'localhost') [line 99]
```

## Global DSN to be used by classes

The DSN to be used. Please see the PEAR documentation information.

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- **See** TemplateDB
- **Link** <http://pear.php.net>

### **MLIB\_GLOBAL\_DEBUG = false [line 85]**

## Global debugging

By turning on global debugging you enable debugging in ALL classes derived from MLib on ALL pages. BE CAREFUL SETTING THIS TO TRUE!!

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- **See** MLib, MLIB::\$debug, MLib::MLib()

MLIB\_INCLUDE\_PATH = '/home/jstump/public\_html/v3/includes' [line 24]

### MLib include path

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)> The absolute path to the directory where MLib resides.

MLIB\_LOG\_FILE = '/tmp/mlib.log' [line 36]

### Default log file

The log file where you wish all class errors to be Must be writable by the webserver.

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- See [MLIB\\_USE\\_SYSLOG](#)

MLIB\_SYSLOG\_PRIORITY = LOG\_WARNING [line 60]

### Syslog priority

The PHP function syslog takes a priority as a parameter. do not know what this means do NOT change this variable.

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- See [MLIB\\_USE\\_SYSLOG](#)
- Link <http://www.php.net/manual/en/function.syslog.php>

MLIB\_TEMPLATE\_PATH = MLIB\_INCLUDE\_PATH.'templates' [line 73]

### Template path

MLib comes with a template class that lets you separate your code from your HTML. This is the path where the template reside.

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- **See** Template, [MLIB\\_INCLUDE\\_PATH](#)

MLIB\_USE\_SYSLOG = false [line 47]

### Use syslog

If set to true MLib will send errors to syslog instead of the file defined in `MLIB_LOG_FILE`

- **Author** Joe Stump < [joe@joestump.net](mailto:joe@joestump.net)>
- **See** [MLIB\\_LOG\\_FILE](#)

require\_once `MLIB_INCLUDE_PATH.'/Template/TemplateVar.php'` [line 151]

### TemplateVar include file

- **See** `TemplateVar`

require\_once `MLIB_INCLUDE_PATH.'/DSN.php'` [line 121]

### DSN include file

- **See** `DSN`

require\_once `MLIB_INCLUDE_PATH.'/Debug.php'` [line 115]

## **Debug include file**

- **See** Debug

require\_once **MLIB\_INCLUDE\_PATH**.'/Table.php' [*line 127*]

## **Table include file**

- **See** Table

require\_once **MLIB\_INCLUDE\_PATH**.'/Template.php' [*line 133*]

## **Template include file**

- **See** Template

require\_once **MLIB\_INCLUDE\_PATH**.'/Template/TemplateDB.php' [*line 145*]

## **TemplateDB include file**

- **See** TemplateDB

require\_once **MLIB\_INCLUDE\_PATH**.'/Template/TemplateFile.php' [*line 139*]

## **TemplateFile include file**

- **See** TemplateFile

require\_once **MLIB\_INCLUDE\_PATH**.'/MLib.php' [line 109]

## **MLib include file**

- **See** MLib

# bug-560578.php

## page-level stuff

- **Package** tests
- **Author** test @

require\_once **MLIB\_INCLUDE\_PATH**.'/Debug.php' [line 12]

## Debug include file

- **Author** test @
- **See** Debug

# bug-560595.php

- **Package** tests

```
one = 'two' [line 10]
```

## bug-562997.php

This page returns a class with name "%s"."\\n", and  
shouldn't find class at all

- **Package** tests

PDERROR\_MULTIPLE\_PARENT = 1 [line 9]

**warning triggered when inheritance could be from more than one class**

# bug-566200.php

- **Package** tests

*void function ezStartPageNumbers(\$x, \$y, \$size, [\$pos = 'left'], [\$pattern = '{PAGENUM} of {TOTALPAGENUM}'], [\$num = '']) [line 8]*

**Function Parameters:**

- **\$x**
- **\$y**
- **\$size**
- **\$pos**
- **\$pattern**
- **\$num**

## tests function param bug

# bug-566600.php

- **Package** tests

*void function a() [line 20]*

**test links to methods and vars**

- See a::d()
- See [a::\\$a](#)

# bug-567059.php

- **Package** tests

*void function aa\_567059(&\$test, \$two) [line 21]*

**Function Parameters:**

- *integer &\$test* testing @ param integer testing
- *\$two*

## test links to methods and vars

- See a::a()
- See [a::\\$a](#)

# bug-645588.php

## Global functions

- **Package** tests
- **Author** Mark van Renswoude
- **Version** 1.0
- **Since** 26-03-2002 15:56

*string* function globalGetVar(\$name) [*line 40*]

**Function Parameters:**

- *string* **\$name** the name of the variable to return

### Get a variable's value

globalGetVar returns the value of an internal variable. This variable must be previously assigned using either globalSetVar, or an indirect setvar-tag in a loaded template.

- See [globalSetVar\(\)](#)

*string* function globalSetVar(\$name, \$value) [*line 22*]

**Function Parameters:**

- *string* **\$name** the name of the variable to set
- *string* **\$value** new value

### Set a variable, used by the template engine but available to all scripts

globalSetVar sets an internal variable. This variable may later be retrieved using globalGetVar, and is automagically available to templates using the getvar-tag.

- See [globalGetVar\(\)](#)

## bug-698356.php

This is a test of bug 698356. Must be parsed with -pp on to test  
should not throw error.

- Package tests

void function bug698356\_Output() [/line 26]

### Create the phptoc.hhp, contents.hhc files needed by MS HTML Help Compiler to create a CHM file

The output function generates the table of contents (contents.hhc) and file list (phptoc.hhp) files used to create a .CHM by the free MS HTML Help compiler. {@, a list of all separate .html files is created in CHM format, and written to phptoc.hhp. This list was generated by writefile.

Next, a call to the table of contents: finishes things off}}

- Link <http://www.microsoft.com/downloads/release.asp?releaseid=33071>
- Uses generateTOC() - assigns to the toc template variable

# bug-authoremail.php

- **Package** tests

*void function test\_authoremail() [line 8]*

## **test function**

- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Link** <http://www.phpdoc.org>

# bug-defineparse.php

- **Package** tests

```
SMART_PATH_DELIMITER = (substr(PHP_OS,0,3)=='WIN')?'\\':/ [line 6]  
does not parse correctly because of function in definition
```

# bug-eofquotes.php

- **Package** tests

*void function test\_eofquotes() [line 7]*

**tests**

# bug-escaping.php

- **Package** tests

*void function test\_escape() [line 5]*

**tests**

*void function test\_escape2() [line 11]*

## bug-loseprocedural.php

- **Package** tests

```
void function test() [line 4]
test2 was lost, isn't any more
```

```
void function test2() [line 15]
```

## bug-pageleveldocsblocks.php

I'm a page level docblock

- **Package** tests

*void* function dummy() [line 12]

**I'm a dummy function currently you need at least one function in your procedural page for this to work**

# bug-quote\_new\_parser.php

- **Package** tests

**\$bqnpTester**

*string = "testing this testing this \$parser(\$me thingo)" [line 15]*

**The tokenizer splits up strings that have inline variables**

and will fool the GLOBAL\_VALUE handler here

bqnpTester = "testing this testing this \$parser(\$me thingo)" [line 9]

**The tokenizer splits up strings that have inline variables**

and will fool the DEFINE\_PARAMS\_PARENTHESIS handler here

*void function bqnpTestie() [line 22]*

**The tokenizer splits up strings that have inline variables**

and will fool the STATICVAR handler here

- **Static Variable Used** string \$test:

# bug-shortdesc.php

- Package tests

testContantBlah = "hi" [line 11]

**Short Desc Test. This is the Extended Comment**  
and so is this

testContantBlah2 = "hi" [line 18]

**Short Desc 2.0 Test. This is the Extended Comment**  
and so is this

testContantBlah3 = "hi" [line 24]

**Short Desc 2.0 Test. This is the Extended Comment**  
and so is this

testContantBlah4 = "hi" [line 30]

**Short Desc 2.0 Test This is still the short desc Comment**  
and this is the extended

testContantBlah5 = "hi" [line 38]

**Short Desc 2.0 Test This is still the short desc Comment and so is this**  
Now were to the extended

# Package tests Classes

## Class a [line 8]

**tests linking to methods and vars**

- **Package** tests

a::\$a

*mixed = [line 9]*

a::\$c

*mixed = [line 11]*

*void function a::b() [line 10]*

## Class b553607\_Parser [line 9]

- **Package** tests
- See [b553607 Parser](#)

## Class baby

[line 8]

- **Package** tests

**baby::\$oopsieindexing**

*mixed = [line 10]*

## Class brokenlinkstovars

[line 8]

- **Package** tests

**brokenlinkstovars::\$broken**

*mixed = [line 10]*

Constructor *void* function **brokenlinkstovars::brokenlinkstovars()** [line 14]

- See [brokenlinkstovars::\\$broken](#)

## Class bug540341

[line 4]

- **Package** tests

`void function bug540341::get_header2($atate) [line 6]`

**Function Parameters:**

- **\$atate**

## Class bug557861

*[line 9]*

**this page will not be shown in the package list and should be**

- **Package** tests

## Class bug\_489398

*[line 4]*

- **Package** tests

**bug\_489398::\$test\_01**

`mixed = '$Id: bug-489398.php 198623 2005-10-17 18:37:50Z jeichorn $' [line 9]`

**Checking the single quote var case**

**bug\_489398::\$test\_02**

*mixed = "Double quoted value" [line 14]*

### **checking the double quote var case**

**bug\_489398::\$test\_03**

*mixed = false [line 19]*

### **Checking the no quote cause**

**bug\_489398::\$test\_04**

*mixed = array() [line 24]*

### **Checking the empty array case**

**bug\_489398::\$test\_05**

*mixed = array("test1","test2" => "value") [line 29]*

### **Checking the array with data case**

**Class childofpriv**  
*[line 58]*

**what happens here? testing bug 564135**

- **Package** tests

# Class ClubBase

[line 103]

- **Package** tests

## ClubBase::\$\_bPrintFlush

*mixed = false* [line 146]

Constructor void function ClubBase::ClubBase() [line 158]

### ClubBase() Der Konstruktor

Macht nicht viel mehr, als fuer alle abgeleiteten Klassen eine Debug-Meldung abzusetzen und PEAR-Funktionen einzubinden

- **Access** public

array function ClubBase::getAllProperties([\${simplify} = false], [\${exclude} = array()]) [line 216]

#### Function Parameters:

- **array \$exclude** diese Eigenschaften werden ignoriert
- **\$simplify**

### getAllProperties() Gibt die Wert aller Eigenschaften einer Klasse zurueck

Es werden nur 'oeffentliche' Eigenschaften (ohne '\_' davor) zurueckgegeben/bearbeitet. Wird ein optionales 'Target' angegeben wird versucht den Werte eines Hashes auszulesen (z.B. \$class->\$target[\$property]) bzw. zu schreiben

- See [ClubBase::getProperty\(\)](#), [ClubBase::getAllProperties\(\)](#), [ClubBase::setProperty\(\)](#), [ClubBase::getPropType\(\)](#)
- **Access** public

*mixed function ClubBase::getProperty(\$property, [\$target = ""], [\$index = 0], [\$simplify = false]) [line 176]*

**Function Parameters:**

- **string \$property** Name der Eigenschaft/Schlüssel
- **string \$target** Name des Targets
- **\$index**
- **\$simplify**

**getProperty() Gibt den Wert einer Eigenschaft zurück**

Es werden nur 'öffentliche' Eigenschaften (ohne '\_' davor) zurückgegeben/bearbeitet. Wird ein optionales 'Target' angegeben wird versucht den Werte eines Hashes auszulesen (z.B. \$class->\$target[\$property]) bzw. zu schreiben

- See [ClubBase::getProperty\(\)](#), [ClubBase::getAllProperties\(\)](#), [ClubBase::setProperty\(\)](#), [ClubBase::getPropType\(\)](#)
- **Access** public

*string function ClubBase::getPropType(\$property, [\$target = null], [\$index = 0]) [line 271]*

**Function Parameters:**

- **\$property**
- **\$target**
- **\$index**

**getPropType gibt den Datentyp eines Objekt-Eigenschaft zurück.**

- **Access** public

*void function ClubBase::loadClass(\$classname, [\$path = './'], [\$ext = '.class.inc'], [\$require = false]) [line 333]*

**Function Parameters:**

- **\$classname**
- **\$path**

- **\$ext**
- **\$require**

*void function ClubBase::printFlush() [line 319]  
void function ClubBase::printVar(\$var, [\$info = "]) [line 326]*

**Function Parameters:**

- **\$var**
- **\$info**

*void function ClubBase::setDebug(\$level, [\$hide = true], [\$file = "], [\$print = true], [\$flush = false]) [line 298]*

**Function Parameters:**

- *integer \$level* Level, bis zu dem Debugausgaben angezeigt werden
- *boolean \$hide* Sollen Debugausgaben im HTML in Kommentare gepackt werden?
- *string \$file* Soll in eine Datei geschrieben werden (null = nein)
- *boolean \$print* Sollen Kommentare in HTML geschrieben werden, auch wenn schon in Datei geloggt wird?
- **\$flush**

### **set\_debug() Debug-Level fuer die Klasse setzen**

Wird diese Methode statisch (ClubBase::set\_debug()) aufgerufen, dann wird die globale Variable `$_ClubDebugOptions` gesetzt. Diese gilt fuer ALLE KLASSEN, die keine eigenen Werte mit `$obj->set_debug()` gestzt haben.

- **Global Variable Used** array `$_ClubDebugOptions`: Globale Debugging-Einstellungen
- **See** `$_ClubDebugOptions`, ClubBase::set\_debug(),  
[ClubBase:: PHPDOCUMENTOR\\_DEBUG\(\)](#)
- **See** `$_bDebugLevel`, `$_bHideDebug`, `$_sLogFile`, `$_bPrintDebug`
- **Static**
- **Access** public

*mixed function ClubBase::setProperty(\$property, \$value, [\$target = "], [\$index = 0], [\$force = false]) [line 240]*

**Function Parameters:**

- *mixed \$value* Wert der der Eigenschaft zugewiesen werden soll
- **\$property**
- **\$target**

- **\$index**
- **\$force**

## **setProperty()** Setzt den Wert einer Eigenschaft

- **Access** public

*void function ClubBase::\_\_ERROR([\${message} = 'unknown error'], [\${code} = PHPDOCUMENATOR\_DEBUG\_ERROR], [\${mode} = null], [\${options} = null], [\${userinfo} = null], [\${error\_class} = null]) [line 423]*

### **Function Parameters:**

- **\$message**
- **\$code**
- **\$mode**
- **\$options**
- **\$userinfo**
- **\$error\_class**

## **\_ERROR()** Fehler registrieren und ggf. zur Debug-Ausgabe weiterleiten

Dies muss erst noch implementiert werden ;-))

- **Static**
- **Final**
- **Access** public

*void function ClubBase::\_\_PHPDOCUMENTOR\_DEBUG(\${message}, [\${level} = PHPDOCUMENTOR\_DEBUG\_INFO]) [line 375]*

### **Function Parameters:**

- *string* **\$message** Meldungstext
- *integer* **\$level** das Debuglevel der Meldung

## **\_PHPDOCUTOR\_DEBUG()** Debugmeldungen ausgeben

Mit dieser Funktion koennen alle Klassen einheitlich Debug-Meldungen ausgeben.

- **Global Variable Used** array \$\_ClubDebugOptions: Globale Debugging-Einstellungen
- See \$\_ClubDebugOptions, ClubBase::set\_debug(),  
[ClubBase:: PHPDOCUTOR\\_DEBUG\(\)](#)
- See \$\_bDebugLevel, \$\_bHideDebug, \$\_sLogFile, \$\_bPrintDebug
- **Static**
- **Access** public

## Class **ctest**

*[line 23]*

- **Package** tests

**ctest::\$t1**

*mixed = [line 25]*

**ctest::\$t3**

*mixed = [line 28]*

Constructor **void** function ctest::ctest() *[line 29]*

**void** function ctest::btest() *[line 38]*

## Class **few**

*[line 8]*

- **Package** tests

```
few::$pfh
mixed = [line 13]

****==really fancy==****
my * is ignored
```

## Class functionincomment [line 8]

- **Package** tests

```
void function functionincomment::process($trxtpe, [$tender = 'C']) [line 14]
```

**Function Parameters:**

- **\$trxtpe**
- **\$tender**

## Class iConverter [line 34]

**Base class for all output converters.**

A Converter takes output from the [Parser](#) and converts it to template-friendly output. A converter for the standard phpDocumentor template, HTMLConverter, is provided with this release. Future releases will have support for other formats and templates, including DocBook, XML, and possibly other HTML templates. The converter takes output directly from NewRender and using [walk\(\)](#) it "walks" over an array of phpDocumentor elements, as represented by descendants of [parserElement](#)

a converter must define the abstract function Convert (an example is `HTMLConverter::Convert()`), a function that takes a passed object and uses it to generate structures for an output template, or directly generates output. Since all elements have their DocBlocks linked directly, this allows for more logical parsing than earlier versions of `phpDocumentor`.

A Converter is passed several data structures in its constructor, all of which are optional in output and may be handled in any way the converter would like. These structures are arrays:

- array of methods by class (see `NewRender::$methods`)
- array of variables by class (see `NewRender::$vars`)
- array of links to documented elements (see `NewRender::$links`)
- array of class parents by name (see `NewRender::$classtree`)
- array of class packages by classname (see `NewRender::$classpackages`)
- array of packages to document (see `NewRender::$packageoutput`)
- array of extended classes by parent classname (see `NewRender::$class_children`)
- array of all documented elements by name (see `NewRender::$elements`)
- array of all documented elements by name, split by package (see `NewRender::$pkg_elements`)
- boolean option, set to true to parse all elements marked `@access private` (see `NewRender::$parsePrivate`)
- boolean option, set to true to stop informative output while parsing (good for cron jobs) (see `NewRender::$quietMode`)

- **Package** tests
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#), [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- **Abstract Element**

`void` function `iConverter::walk()` [*line 37*]

## Class iHTMLConverter

[*line 49*]

### Base class for all output converters.

A Converter takes output from the [Parser](#) and converts it to template-friendly output. A converter for the standard `phpDocumentor` template, `HTMLConverter`, is provided with this release. Future releases will have support for other formats and templates, including

DocBook, XML, and possibly other HTML templates. The converter takes output directly from NewRender and using [walk\(\)](#) it "walks" over an array of phpDocumentor elements, as represented by descendants of [parserElement](#)

a converter must define the abstract function Convert (an example is `HTMLConverter::Convert()`), a function that takes a passed object and uses it to generate structures for an output template, or directly generates output. Since all elements have their DocBlocks linked directly, this allows for more logical parsing than earlier versions of phpDocumentor.

A Converter is passed several data structures in its constructor, all of which are optional in output and may be handled in any way the converter would like. These structures are arrays:

- array of methods by class (see `NewRender::$methods`)
- array of variables by class (see `NewRender::$vars`)
- array of links to documented elements (see `NewRender::$links`)
- array of class parents by name (see `NewRender::$classtree`)
- array of class packages by classname (see `NewRender::$classpackages`)
- array of packages to document (see `NewRender::$packageoutput`)
- array of extended classes by parent classname (see `NewRender::$class_children`)
- array of all documented elements by name (see `NewRender::$elements`)
- array of all documented elements by name, split by package (see `NewRender::$pkg_elements`)
- boolean option, set to true to parse all elements marked @access private (see `NewRender::$parsePrivate`)
- boolean option, set to true to stop informative output while parsing (good for cron jobs) (see `NewRender::$quietMode`)

- **Package** tests

`void` function `iHTMLConverter::Convert()` [[line 51](#)]

## Class `iiparserBase` [[line 8](#)]

- **Package** tests

**iiparserBase::\$type**

*string = 'base' [line 14]*

**always base**

**iiparserBase::\$value**

*mixed = false [line 20]*

**set to different things by its descendants**

- **Abstract Element**

*string function iiparserBase::getType() [line 25]*

*mixed function iiparserBase::getValue() [line 41]*

*void function iiparserBase::setValue(\$value) [line 33]*

**Function Parameters:**

- *mixed \$value* set the value of this element

## Class iNewRender

*[line 66]*

- **Package** tests

**iNewRender::\$classpackage**

*string = 'default' [line 233]*

**set to the name of the package of the current class being parsed**

#### iNewRender::\$classpackages

*array = array() [line 153]*

**used in Converter::getClassPackage() to inherit package from parent classes.**

format: array(classname => array(array(package,subpackage),  
array(package1,subpackage1),.... If a name conflict exists between two packages,  
automatic inheritance will not work, and the packages will need to be documented  
separately.

#### iNewRender::\$classsubpackage

*string = " [line 239]*

**set to the name of the subpackage of the current class being parsed**

#### iNewRender::\$classtree

*array = array() [line 140]*

**a tree of class inheritance by name.**

format: array(childname => parentname, childname1 => parentname1,  
rootname => 0, ... )

- See Converter::generateClassTreeFromClass()

#### iNewRender::\$class\_children

*array = array() [line 183]*

**An array of extended classes by package and parent class Format:**

array(packagename => array(parentclass => array(childclassname1,  
childclassname2,...  
)) ) ) )

#### iNewRender::\$data

*parserData* = [line 227]

## **data contains parsed structures for the current page being parsed**

- See [parserData](#)

### **iNewRender::\$elements**

*mixed* = array() [line 185]

### **iNewRender::\$event\_handlers**

```
mixed = array(  
    'docblock' => 'handleDocBlock',  
    'page' => 'handlePage',  
    'class' => 'handleClass',  
    'define' => 'handleDefine',  
    'function' => 'handleFunction',  
    'method' => 'handleMethod',  
    'var' => 'handleVar',  
    'packagepage' => 'handlePackagePage',  
    'include' => 'handleInclude',  
) [line 210]
```

## **the functions which handle output from [Parser](#)**

- See [handleEvent\(\)](#), [handleDocBlock\(\)](#), [handlePage\(\)](#), [handleClass\(\)](#), [handleDefine\(\)](#), [handleFunction\(\)](#), [handleMethod\(\)](#), [handleVar\(\)](#), [handlePackagePage\(\)](#), [handleInclude\(\)](#)

### **iNewRender::\$links**

*array* = array() [line 128]

## **the workhorse of linking.**

This array is an array of link objects of format:  
[package][subpackage][eltype][elname] = descendant of [abstractLink](#) eltype can be  
page|include|function|define|class|method|var if eltype is method or var, the array format  
is: [package][subpackage][eltype][class][elname]

- See [functionLink](#), [pageLink](#), [classLink](#), [defineLink](#), [methodLink](#), [varLink](#)

### iNewRender::\$methods

*array = array() [line 83]*

#### **array of methods by package, subpackage and class**

format: array(packagename => array(subpackagename =>  
array(classname => array(methodname1 => [parserMethod](#) class,  
methodname2 => [parserMethod](#) class,...) ) )  
) )

- See [Converter](#)

### iNewRender::\$packageoutput

*mixed = false [line 203]*

#### **array of packages to parser and output documentation for, if not all packages should be documented**

Format: array(package1,package2,...) or false if not set Use this option to limit output similar to ignoring files. If you have some temporary files that you don't want to specify by name but don't want included in output, set a package name for all the elements in your project, and set packageoutput to that name. the default package will be ignored. Parsing speed does not improve. If you want to ignore files for speed reasons, use the ignore command-line option

- See [Io](#)

### iNewRender::\$pages

*mixed = array() [line 189]*

### iNewRender::\$parsePrivate

*bool = false [line 107]*

**set in [phpdoc.in](#) to the value of the parserprivate commandline option.**

If this option is true, elements with an @access private tag will be parsed and displayed

**iNewRender::\$pkg\_elements**

*mixed = array() [line 187]*

**iNewRender::\$private\_class**

*mixed = false [line 116]*

**this variable is used to prevent parsing of private elements if \$parsePrivate is false.**

it is also used by the packageoutput setting to prevent parsing of elements that aren't in the desired output packages

- See [iNewRender::\\$parsePrivate](#)
- See [iNewRender::\\$packageoutput](#)

**iNewRender::\$quietMode**

*bool = false [line 246]*

**set in [phpdoc.in](#) to the value of the quitemode commandline option.**

If this option is true, informative output while parsing will not be displayed (documentation is unaffected)

**iNewRender::\$targetDir**

*mixed = [line 159]*

**used to set the output directory**

- See [setTargetDir\(\)](#)

iNewRender::\$vars

array = array() [line 100]

**array of class variables by package, subpackage and class**

format: array(packagename => array(subpackagename =>  
array(classname => array(variablename1 => [parserMethod](#) class,  
variablename2 => [parserMethod](#) class,...) )  
) )

- See [Converter](#)

## Class iParser

[line 45]

- **Package** tests

## Class iparserElement

[line 59]

- **Package** tests

## Class kiddie\_b587733

*[line 25]*

- **Package** tests

Constructor `void` function `kiddie_b587733::kiddie_b587733()` *[line 27]*

## Class mama

*[line 8]*

- **Package** tests

## Class metoo

*[line 26]*

- **Package** tests

**metoo::\$mine**

`mixed = 'but this'` *[line 28]*

## Class multipl [line 18]

- **Package** tests

**multipl::\$manyvars**

*string = [line 24]*

- **Var** first var

*string function multipl::func() [line 30]*

## Class parent\_b587733 [line 6]

**inherited functions with @access private should not be shown in inherited list of child**

- **Package** tests

## Class priv1 [line 44]

- **Package** tests

## Class RecordWarning

*[line 19]*

- **Package** tests

## Class summary\_form

*[line 4]*

- **Package** tests

### **summary\_form::\$dp**

*mixed = [line 6]*

*void function summary\_form::blah() [line 16]*  
*void function summary\_form::get\_header2(\$atate) [line 11]*

**Function Parameters:**

- **\$atate**

- See [blah\(\)](#)

## Class test

*[line 9]*

**I'm a odd test case**

the @ sign is my friend @ at teh beggining of a line

- **Package** tests

## Class test2

*[line 19]*

**tags demonstration, but this @version tag is ignored**

- **Package** tests
- **Author** this tag is parsed

## Class testarraybug

*[line 10]*

**tests variable names with the word 'array' in them**

- **Package** tests

**testarraybug::\$arrayType**

*mixed = 'name should be arrayType' [line 12]*

**testarraybug::\$arrayType1**

*mixed = [line 17]*

**test with no default, should be arrayType1**

**testarraybug::\$myarrayName**

*mixed = 'name should be myarrayName' [line 13]*

**testarraybug::\$myarrayName1**

*mixed = [line 21]*

**test with no default, should be myarrayName1**

## Class testClass

*[line 8]*

- **Package** tests

Constructor **void** function `testClass::testClass()` *[line 10]*

## Class testme

*[line 13]*

- **Package** tests

**testme::\$me**

```
mixed = array('item1' => 2,
    #          'NOTME' => hahaha,
    //          'MENEITHER' => oops,
    'item2' => 3) [line 17]
```

## bug-541886.php

### Test for bug #541886

Multiple @package or @subpackage tags causing a warning

- **Package** tests
- **Sub-Package** blah

## Class test\_541886

*[line 20]*

### Test for bug #541886

Multiple @package or @subpackage tags causing a warning

- **Package** tests
- **Sub-Package** blah

## Class notseen

*[line 10]*

- **Package** tests
- **Sub-Package** notparsed

# HighlightParserGetInlineTagsTests.php

## Unit Tests for the `HighlightParser->getInlineTags()` method

- **Package** tests
- **Sub-Package** `PhpDocumentorUnitTests`
- **Author** Chuck Burgess
- **Since** 1.4.0a2

`require_once 'PhpDocumentor/phpDocumentor/Setup.inc.php'` [line 51]

### **PhpDocumentor Setup**

required by `PhpDocumentor` to instantiate the environment

- **Since** 1.4.0a2

`require_once "PHPUnit/Framework/TestSuite.php"` [line 32]

### **TestSuite**

required by `PHPUnit`

- **Since** 1.4.0a2

`require_once "PHPUnit/Framework\TestCase.php"` [line 25]

### **TestCase**

required by `PHPUnit`

- **Since** 1.4.0a2

# IntermediateParserAddPrivatePageTests.php

## Unit Tests for the IntermediateParser->addPrivatePage() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a2

require\_once 'PhpDocumentor/phpDocumentor/Setup.inc.php' [line 51]

### PhpDocumentor Setup

required by PhpDocumentor to instantiate the environment

- **Since** 1.4.0a2

require\_once "PHPUnit/Framework/TestSuite.php" [line 32]

### TestSuite

required by PHPUnit

- **Since** 1.4.0a2

require\_once "PHPUnit/Framework/TestCase.php" [line 25]

### TestCase

required by PHPUnit

- **Since** 1.4.0a2

# ParserClassGetSourceLocationTests.php

## Unit Tests for the ParserClass->getSourceLocation() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **TODO** research possibility of refactoring ParserClass->getSourceLocation() and ParserPage->getSourceLocation() into a common method... also, there might be more occurrences of similar getSourceLocation() methods in other classes.
- **Since** 1.4.0a1

require\_once 'PhpDocumentor/phpDocumentor/Setup.inc.php' [line 58]

### PhpDocumentor Setup

required by PhpDocumentor to instantiate the environment

- **Since** 1.4.0a1

require\_once "PHPUnit/Framework/TestSuite.php" [line 39]

### TestSuite

required by PHPUnit

- **Since** 1.4.0a1

require\_once "PHPUnit/Framework\TestCase.php" [line 32]

### TestCase

required by PHPUnit

- **Since** 1.4.0a1

# ParserPageGetSourceLocationTests.php

## Unit Tests for the ParserPage->getSourceLocation() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **TODO** research possibility of refactoring ParserClass->getSourceLocation() and ParserPage->getSourceLocation() into a common method... also, there might be more occurrences of similar getSourceLocation() methods in other classes.
- **Since** 1.4.0a1

require\_once 'PhpDocumentor/phpDocumentor/Setup.inc.php' [line 58]

### PhpDocumentor Setup

required by PhpDocumentor to instantiate the environment

- **Since** 1.4.0a1

require\_once "PHPUnit/Framework/TestSuite.php" [line 39]

### TestSuite

required by PHPUnit

- **Since** 1.4.0a1

require\_once "PHPUnit/Framework\TestCase.php" [line 32]

### TestCase

required by PHPUnit

- **Since** 1.4.0a1

## **phpDocumentorSetupCleanConverterNamePieceTests.php** Unit Tests for the `phpDocumentor_setup->cleanConverterNamePiece()` method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.3.2

`require_once 'PhpDocumentor/phpDocumentor/Setup.inc.php' [line 51]`

### **PhpDocumentor Setup**

required by PhpDocumentor to instantiate the environment

- **Since** 1.3.2

`require_once "PHPUnit/Framework/TestSuite.php" [line 32]`

### **TestSuite**

required by PHPUnit

- **Since** 1.3.2

`require_once "PHPUnit/Framework/TestCase.php" [line 25]`

### **TestCase**

required by PHPUnit

- **Since** 1.3.2

# phpDocumentorSetupDecideOnOrOffTests.php

## Unit Tests for the `phpDocumentor_setup->decideOnOrOff()` method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.3.2

`PHPDOCUMENTOR_BASE = dirname(dirname(__FILE__))` [line 42]

### Base directory of code

Needed by some of the objects being tested in the suites.

- **Since** 1.4.1

`PHPUnit_MAIN_METHOD = "phpDocumentorSetupDecideOnOrOffTests::main"` [line 17]

### PHPUnit main() hack

"Call class::main() if this source file is executed directly."

- **Since** 1.3.2

`require_once 'PhpDocumentor/phpDocumentor/Setup.inc.php'` [line 51]

### PhpDocumentor Setup

required by PhpDocumentor to instantiate the environment

- **Since** 1.3.2

require\_once "PHPUnit/Framework/TestSuite.php" [*line 32*]

## **TestSuite**

required by PHPUnit

- **Since** 1.3.2

require\_once "PHPUnit/Framework/TestCase.php" [*line 25*]

## **TestCase**

required by PHPUnit

- **Since** 1.3.2

# phpDocumentorTParserGetInlineTagsTests.php

## Unit Tests for the `phpDocumentorTParser->getInlineTags()` method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a2

`require_once 'PhpDocumentor/phpDocumentor/Setup.inc.php'` [line 51]

### **PhpDocumentor Setup**

required by PhpDocumentor to instantiate the environment

- **Since** 1.4.0a2

`require_once "PHPUnit/Framework/TestSuite.php"` [line 32]

### **TestSuite**

required by PHPUnit

- **Since** 1.4.0a2

`require_once "PHPUnit/Framework\TestCase.php"` [line 25]

### **TestCase**

required by PHPUnit

- **Since** 1.4.0a2

## Class tests\_HighlightParserGetInlineTagsTests

*[line 60]*

### Unit Testing of the HighlightParser's getInlineTags() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a2

*void* function tests\_HighlightParserGetInlineTagsTests::main() *[line 81]*

**Runs the test methods of this class.**

- **Since** 1.4.0a2
- **Static**
- **Access** public

*void* function tests\_HighlightParserGetInlineTagsTests::setUp() *[line 94]*

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.4.0a2

- **Access** protected

*void function tests\_HighlightParserGetInlineTagsTests::tearDown() [line 108]*

**Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.4.0a2
- **Access** protected

*void function tests\_HighlightParserGetInlineTagsTests::testShowCorrectBehaviorWhenGivenOneEmptyArg() [line 131]*

**Shows correct behavior when called with no actual value**

and no \$endinternal flag arg

- **Since** 1.4.0a2
- **Access** public

*void function*

*tests\_HighlightParserGetInlineTagsTests::testShowCorrectBehaviorWhenGivenOneEmptyArgAndFalse() [line 139]*

**Shows correct behavior when called with no actual value**

and a FALSE \$endinternal flag arg

- **Since** 1.4.0a2
- **Access** public

*void function*

*tests\_HighlightParserGetInlineTagsTests::testShowCorrectBehaviorWhenGivenOneEmptyArgAndTrue() [line 147]*

**Shows correct behavior when called with no actual value  
and a TRUE \$endinternal flag arg**

- **Since** 1.4.0a2
- **Access** public

## Class tests\_IntermediateParserAddPrivatePageTests

*[line 60]*

**Unit Testing of the IntermediateParser's addPrivatePage() method**

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a2

*void* function tests\_IntermediateParserAddPrivatePageTests::main() *[line 99]*

**Runs the test methods of this class.**

- **Since** 1.4.0a2
- **Static**
- **Access** public

*void* function tests\_IntermediateParserAddPrivatePageTests::setUp() *[line 112]*

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.4.0a2
- **Access** protected

*void function tests\_IntermediateParserAddPrivatePageTests::tearDown() [line 131]*

**Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.4.0a2
- **Access** protected

*void function*

*tests\_IntermediateParserAddPrivatePageTests::testShowCorrectBehaviorWhenPrivatePageArrayIsEmpty() [line 200]*

**Shows correct behavior for adding a private page object,**

when the privatepages array is completely empty

- **Since** 1.4.0a2
- **Access** public

*void function*

*tests\_IntermediateParserAddPrivatePageTests::testShowCorrectBehaviorWhenPrivatePageArrayIsNotAlreadyEmpty() [line 157]*

**Shows correct behavior for adding a private page object,**

when the privatepages array already has an element

- **Since** 1.4.0a2
- **Access** public

# Class tests\_ParserClassGetSourceLocationTests

*[line 67]*

## Unit Testing of the ParserClass's getSourceLocation() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a1

*void* function tests\_ParserClassGetSourceLocationTests::main() *[line 88]*

**Runs the test methods of this class.**

- **Since** 1.4.0a1
- **Static**
- **Access** public

*void* function tests\_ParserClassGetSourceLocationTests::setUp() *[line 101]*

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.4.0a1
- **Access** protected

*void* function tests\_ParserClassGetSourceLocationTests::tearDown() *[line 116]*

**Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.4.0a1
- **Access** protected

`void function tests_ParserClassGetSourceLocationTests::testWhenLocationNotSetAndPearizeFalse() [line 149]`

**Shows correct behavior when**

sourceLocation is not set yet with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

`void function tests_ParserClassGetSourceLocationTests::testWhenLocationNotSetAndPearizeNull() [line 140]`

**Shows correct behavior when**

sourceLocation is not set yet with no pearize value set

- **Since** 1.4.0a1
- **Access** public

`void function tests_ParserClassGetSourceLocationTests::testWhenLocationNotSetAndPearizeTrue() [line 158]`

**Shows correct behavior when**

sourceLocation is not set yet with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

`void function tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeFalse()  
[line 178]`

**Shows correct behavior when**

sourceLocation is set to an absolute path that is not a "pear" location,  
with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

`void function tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeNull() [line  
168]`

**Shows correct behavior when**

sourceLocation is set to an absolute path that is not a "pear" location,  
with no pearize value set

- **Since** 1.4.0a1
- **Access** public

`void function tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeTrue()  
[line 188]`

**Shows correct behavior when**

sourceLocation is set to an absolute path that is not a "pear" location,  
with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeFalse()  
[line 238]

**Include a ".." in an absolute, non-PEAR path,**  
with pearize explicitly false

- Since 1.4.0a1
- Access public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeNull() [line 229]

**Include a ".." in an absolute, non-PEAR path,**  
with pearize not set

- Since 1.4.0a1
- Access public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeTrue()  
[line 247]

**Include a ".." in an absolute, non-PEAR path,**  
with pearize explicitly true

- Since 1.4.0a1
- Access public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeFalse() [line 266]

**Include a ".." in a relative, non-PEAR path,**

with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void* function

tests\_ParserClassGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeNull() [*line 257*]

**Include a ".." in a relative, non-PEAR path,**  
with pearize not set

- **Since** 1.4.0a1
- **Access** public

*void* function

tests\_ParserClassGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeTrue() [*line 275*]

**Include a ".." in a relative, non-PEAR path,**  
with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void* function tests\_ParserClassGetSourceLocationTests::testWhenPearLocationSetAndPearizeFalse() [*line 209*]

**Show correct behavior when**

sourceLocation is set to an absolute path that IS a "pear" location, with  
pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserClassGetSourceLocationTests::testWhenPearLocationSetAndPearizeNull() [line 199]*

**Show correct behavior when**

sourceLocation is set to an absolute path that IS a "pear" location, with pearize not set

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserClassGetSourceLocationTests::testWhenPearLocationSetAndPearizeTrue() [line 219]*

**Show correct behavior when**

sourceLocation is set to an absolute path that IS a "pear" location, with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

*void function  
tests\_ParserClassGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeFalse() [line 294]*

**Include a ".." in an absolute, PEAR path,  
with pearize explicitly false**

- **Since** 1.4.0a1
- **Access** public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeNull() [*line 285*]

**Include a ".." in an absolute, PEAR path,**  
with pearize not set

- **Since** 1.4.0a1
- **Access** public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeTrue() [*line 303*]

**Include a ".." in an absolute, PEAR path,**  
with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

*void* function  
tests\_ParserClassGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeFalse() [*line 322*]

**Include a ".." in a relative, PEAR path,**  
with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void* function tests\_ParserClassGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeNull() [*line 313*]

**Include a ".." in a relative, PEAR path,**

with pearize not set

- **Since** 1.4.0a1
- **Access** public

*void* function tests\_ParserClassGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeTrue()  
[line 331]

**Include a ".." in a relative, PEAR path,**  
with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

## Class tests\_ParserPageGetSourceLocationTests

[line 67]

**Unit Testing of the ParserPage's getSourceLocation() method**

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a1

*void* function tests\_ParserPageGetSourceLocationTests::main() [line 88]

**Runs the test methods of this class.**

- **Since** 1.4.0a1
- **Static**
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::setUp() [line 101]*

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.4.0a1
- **Access** protected

*void function tests\_ParserPageGetSourceLocationTests::tearDown() [line 116]*

**Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.4.0a1
- **Access** protected

*void function tests\_ParserPageGetSourceLocationTests::testWhenLocationNotSetAndPearizeFalse() [line 148]*

**Shows correct behavior when**

sourceLocation is not set yet with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenLocationNotSetAndPearizeNull() [line 139]*

**Shows correct behavior when**

sourceLocation is not set yet with no pearize value set

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenLocationNotSetAndPearizeTrue() [line 157]*

**Shows correct behavior when**

sourceLocation is not set yet with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeFalse() [line 177]*

**Shows correct behavior when**

sourceLocation is set to an absolute path that is not a "pear" location,  
with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeNull() [line 167]*

**Shows correct behavior when**

sourceLocation is set to an absolute path that is not a "pear" location,  
with no pearize value set

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeTrue() [line 190]*

### **Shows correct behavior when**

sourceLocation is set to an absolute path that is not a "pear" location,  
with pearize explicitly true

- **TODO** Revisit this test... I think it highlights a bug in the getSourceLocation method.  
Compare it with the same test in bug1574047.php against similar method parserClass->getSourceLocation().
- **Since** 1.4.0a1
- **Access** public

*void function  
tests\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeFalse() [line 240]*

### **Include a ".." in an absolute, non-PEAR path, with pearize explicitly false**

- **Since** 1.4.0a1
- **Access** public

*void function  
tests\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeNull() [line 231]*

### **Include a ".." in an absolute, non-PEAR path, with pearize not set**

- **Since** 1.4.0a1
- **Access** public

*void* function

tests\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeTrue()  
[line 252]

**Include a ".." in an absolute, non-PEAR path,**  
with pearize explicitly true

- **TODO** Revisit this test... I think it highlights a bug in the getSourceLocation method.  
Compare it with the same test in bug1574047.php against similar method parserClass->getSourceLocation().
- **Since** 1.4.0a1
- **Access** public

*void* function

tests\_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeFalse() [line 271]

**Include a ".." in a relative, non-PEAR path,**  
with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void* function

tests\_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeNull() [line 262]

**Include a ".." in a relative, non-PEAR path,**  
with pearize not set

- **Since** 1.4.0a1
- **Access** public

*void* function  
tests\_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeTrue() [line 283]  
**Include a ".." in a relative, non-PEAR path,**  
with pearize explicitly false

- **TODO** Revisit this test... I think it highlights a bug in the getSourceLocation method.  
Compare it with the same test in bug1574047.php against similar method parserClass->getSourceLocation().
- **Since** 1.4.0a1
- **Access** public

*void* function tests\_ParserPageGetSourceLocationTests::testWhenPearLocationSetAndPearizeFalse() [line 211]

#### **Show correct behavior when**

sourceLocation is set to an absolute path that IS a "pear" location, with  
pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void* function tests\_ParserPageGetSourceLocationTests::testWhenPearLocationSetAndPearizeNull() [line 201]

#### **Show correct behavior when**

sourceLocation is set to an absolute path that IS a "pear" location, with  
pearize not set

- **Since** 1.4.0a1
- **Access** public

```
void function tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetAndPearizeTrue() [line  
221]
```

### Show correct behavior when

sourceLocation is set to an absolute path that IS a "pear" location, with pearize explicitly true

- Since 1.4.0a1
- Access public

```
void function  
tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeFalse() [line  
302]
```

### Include a ".." in an absolute, PEAR path, with pearize explicitly false

- Since 1.4.0a1
- Access public

```
void function  
tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeNull() [line 293]
```

### Include a ".." in an absolute, PEAR path, with pearize not set

- Since 1.4.0a1
- Access public

```
void function  
tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeTrue() [line  
311]
```

### Include a ".." in an absolute, PEAR path,

with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeFalse()  
[line 330]*

**Include a ".." in a relative, PEAR path,**  
with pearize explicitly false

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeNull()  
[line 321]*

**Include a ".." in a relative, PEAR path,**  
with pearize not set

- **Since** 1.4.0a1
- **Access** public

*void function tests\_ParserPageGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeTrue()  
[line 339]*

**Include a ".." in a relative, PEAR path,**  
with pearize explicitly true

- **Since** 1.4.0a1
- **Access** public

# Class tests\_phpDocumentorSetupCleanConverterNamePiece Tests [line 61]

## Unit Testing of the phpDocumentor\_setup's cleanConverterNamePiece() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.3.2

*void* function tests\_phpDocumentorSetupCleanConverterNamePieceTests::main() [line 97]

**Runs the test methods of this class.**

- **Since** 1.3.2
- **Static**
- **Access** public

*void* function tests\_phpDocumentorSetupCleanConverterNamePieceTests::setUp() [line 110]

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.3.2

- **Access** protected

*void function tests\_phpDocumentorSetupCleanConverterNamePieceTests::tearDown() [line 123]*

**Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.3.2
- **Access** protected

*void function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnPrimaryWithOneArg()  
[line 412]*

**Verify no up-to-parent pathing is allowed...**

the resulting converter names are generally invalid. This test uses one arg with value of "/var/log/HTML"

- **Since** 1.3.2
- **Access** public

*void function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnPrimaryWithTwoArgs()  
[line 421]*

**Verify no up-to-parent pathing is allowed...**

the resulting converter names are generally invalid. This test uses two args with value of "/var/log/HTML"

- **Since** 1.3.2
- **Access** public

```
void function  
tests_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnSecondary() [line  
430]
```

### Verify no up-to-parent pathing is allowed...

the resulting converter names are generally invalid. This test uses two args with value of "/var/log/frames"

- **Since** 1.3.2
- **Access** public

```
void function  
tests_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnTertiary() [line  
439]
```

### Verify no up-to-parent pathing is allowed...

the resulting converter names are generally invalid. This test uses two args with value of "/var/log/default"

- **Since** 1.3.2
- **Access** public

```
void function  
tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleAndInvalidSecondary()  
[line 478]
```

### Extreme example of messy input...

the resulting converter names are generally invalid. This test uses two args with value of "..//.frames"

- **Since** 1.3.2
- **Access** public

```
void function  
tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleAndInvalidTertiaryA() [line  
487]
```

### **Extreme example of messy input...**

the resulting converter names are generally invalid. This test uses two arg with value of "./default/.##/"

- **Since** 1.3.2
- **Access** public

```
void function  
tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleAndInvalidTertiaryB() [line  
496]
```

### **Extreme example of messy input...**

the resulting converter names are generally invalid. This test uses two arg with value of "//default//"

- **Since** 1.3.2
- **Access** public

```
void function  
tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleButValidPrimaryWithOneAr  
g() [line 451]
```

### **Extreme example of messy input...**

the resulting converter names are generally invalid. This test uses one arg with value of "H/.T./M##L"

- **Since** 1.3.2
- **Access** public

```
void function
```

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleButValidPrimaryWithTwoArgs() [line 460]

### **Extreme example of messy input...**

the resulting converter names are generally invalid. This test uses two args with value of "H/.T./M##L"

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleButValidSecondary() [line 469]

### **Extreme example of messy input...**

the resulting converter names are generally invalid. This test uses two args with value of "...frames"

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondaryDefault() [line 230]

### **Shows correct behavior for handling the perfect expected "default" secondary value**

when called with two args

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondaryDocbookPardoc2() [line 238]

**Shows correct behavior for handling the perfect expected "DocBook/peardoc2" secondary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondaryFrames()  
[line 214]*

**Shows correct behavior for handling the perfect expected "frames" secondary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondarySmarty()  
[line 222]*

**Shows correct behavior for handling the perfect expected "Smarty" secondary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDefault()  
[line 247]*

**Shows correct behavior for handling the perfect expected "default" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomDefault()  
[line 312]`

**Shows correct behavior for handling the perfect expected "DOM/default" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomEarthli()  
[line 320]`

**Shows correct behavior for handling the perfect expected "DOM/earthli" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomL0l33t()  
[line 328]`

**Shows correct behavior for handling the perfect expected "DOM/l0l33t" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomPhpdocde() [line 336]

**Shows correct behavior for handling the perfect expected "DOM/phpdoc.de" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomPhptmllib() [line 344]

**Shows correct behavior for handling the perfect expected "DOM/phptmllib" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryEarthli() [line 255]

**Shows correct behavior for handling the perfect expected "earthli" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryHands() [line 287]`

**Shows correct behavior for handling the perfect expected "HandS" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryL0l33t() [line 263]`

**Shows correct behavior for handling the perfect expected "l0l33t" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPear() [line 295]`

**Shows correct behavior for handling the perfect expected "PEAR" tertiary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

`void function tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPhp() [line 303]`

**Shows correct behavior for handling the perfect expected "PHP" tertiary**

**value**

when called with two args

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPhpdode()  
[line 271]*

**Shows correct behavior for handling the perfect expected "phpdoc.de"**

**tertiary value**

when called with two args

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPhphmlib()  
[line 279]*

**Shows correct behavior for handling the perfect expected "phphmlib"**

**tertiary value**

when called with two args

- **Since** 1.3.2
- **Access** public

*void function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryCHM() [line 148]*

**Shows correct behavior for handling the perfect expected "CHM"**

**primary value**

when called with one arg

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryHTML() [line 156]

**Shows correct behavior for handling the perfect expected "HTML"**  
**primary value**  
when called with one arg

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryPDF() [line 164]

**Shows correct behavior for handling the perfect expected "PDF"**  
**primary value**  
when called with one arg

- **Since** 1.3.2
- **Access** public

*void* function

tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryXML() [line 172]

**Shows correct behavior for handling the perfect expected "XML"**  
**primary value**  
when called with one arg

- **Since** 1.3.2
- **Access** public

*void* function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryCHM() [*line 181*]  
**Shows correct behavior for handling the perfect expected "CHM"**  
**primary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void* function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryHTML() [*line 189*]  
**Shows correct behavior for handling the perfect expected "HTML"**  
**primary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void* function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryPDF() [*line 197*]  
**Shows correct behavior for handling the perfect expected "PDF"**  
**primary value**  
when called with two args

- **Since** 1.3.2

- **Access** public

*void function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryXML() [line 205]*  
**Shows correct behavior for handling the perfect expected "XML"**  
**primary value**  
when called with two args

- **Since** 1.3.2
- **Access** public

*void function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnPrimaryWithOneArg() [line 375]*

**Verify no up-to-parent pathing is allowed...**

the resulting converter names are generally invalid. This test uses one arg with value of "../HTML"

- **Since** 1.3.2
- **Access** public

*void function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnPrimaryWithTwoArgs() [line 384]*

**Verify no up-to-parent pathing is allowed...**

the resulting converter names are generally invalid. This test uses two args with value of "../HTML"

- **Since** 1.3.2
- **Access** public

*void* function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnSecondary()  
[line 393]

**Verify no up-to-parent pathing is allowed...**

the resulting converter names are generally invalid. This test uses two args with value of "../frames"

- **Since** 1.3.2
- **Access** public

*void* function  
tests\_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnTertiary() [line 402]

**Verify no up-to-parent pathing is allowed...**

the resulting converter names are generally invalid. This test uses two args with value of "../default"

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupCleanConverterNamePieceTests::testUserDefinedTertiaryValue()  
[line 353]

**Shows correct behavior for handling the perfect expected "b2evo.v-1-10" tertiary value**

(an example of a user-defined template not packaged with PhpDocumentor) when called with two args

- **Since** 1.4.0
- **Access** public

# Class

## tests\_phpDocumentorSetupDecideOnOrOffTests [line 60]

### Unit Testing of the phpDocumentor\_setup's decideOnOrOff() method

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.3.2

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::main() [line 75]

**Runs the test methods of this class.**

- **Since** 1.3.2
- **Static**
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::setUp() [line 88]

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.3.2
- **Access** protected

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::tearDown() [line 99]

## **Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.3.2
- **Access** protected

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testBasicOff() [line 120]*

**Shows correct behavior for handling the perfect expected "off" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testBasicOn() [line 127]*

**Shows correct behavior for handling the perfect expected "on" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyEmpty() [line 199]*

**Shows correct behavior for handling the fuzzy "" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyFalseA() [line 170]*

**Shows correct behavior for handling the fuzzy "false" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyFalseB() [line 177]*

**Shows correct behavior for handling the fuzzy "False" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyFalseC() [line 184]*

**Shows correct behavior for handling the fuzzy "FALSE" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyNoA() [line 149]*

**Shows correct behavior for handling the fuzzy "no" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyNoB() [line 156]*

**Shows correct behavior for handling the fuzzy "No" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyNoC() [line 163]*

**Shows correct behavior for handling the fuzzy "NO" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOffA() [line 135]*

**Shows correct behavior for handling the fuzzy "Off" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOffB() [line 142]*

**Shows correct behavior for handling the fuzzy "OFF" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOnA() [line 206]*

**Shows correct behavior for handling the fuzzy "On" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOnB() [*line 213*]

**Shows correct behavior for handling the fuzzy "ON" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOne() [*line 276*]

**Shows correct behavior for handling the fuzzy "1" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyTrueA() [*line 255*]

**Shows correct behavior for handling the fuzzy "true" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyTrueB() [*line 262*]

**Shows correct behavior for handling the fuzzy "True" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyTrueC() [*line 269*]

**Shows correct behavior for handling the fuzzy "TRUE" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesA() [line 220]

**Shows correct behavior for handling the fuzzy "y" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesB() [line 227]

**Shows correct behavior for handling the fuzzy "Y" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesC() [line 234]

**Shows correct behavior for handling the fuzzy "yes" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesD() [line 241]

**Shows correct behavior for handling the fuzzy "Yes" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesE() [line 248]*

**Shows correct behavior for handling the fuzzy "YES" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testFuzzyZero() [line 191]*

**Shows correct behavior for handling the fuzzy "0" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedGreatLiterature() [line 324]*

**Shows correct behavior for handling an odd "ash nazg durbatuluk" value**

- **Since** 1.3.2
- **Access** public

*void function tests\_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedLargeNumber() [line 317]*

**Shows correct behavior for handling an odd "10" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedNegative() [*line 310*]

**Shows correct behavior for handling an odd "-1" value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedNull() [*line 303*]

**Shows correct behavior for handling an odd NULL value**

- **Since** 1.3.2
- **Access** public

*void* function tests\_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedSpaces() [*line 296*]

**Shows correct behavior for handling an odd " " value**

- **Since** 1.3.2
- **Access** public

## Class

### tests\_phpDocumentorTParserGetInlineTagsTests

[*line 60*]

**Unit Testing of the phpDocumentorTParser's getInlineTags() method**

- **Package** tests
- **Sub-Package** PhpDocumentorUnitTests
- **Author** Chuck Burgess
- **Since** 1.4.0a2

*void* function tests\_phpDocumentorTParserGetInlineTagsTests::main() [line 87]

**Runs the test methods of this class.**

- **Since** 1.4.0a2
- **Static**
- **Access** public

*void* function tests\_phpDocumentorTParserGetInlineTagsTests::setUp() [line 100]

**Sets up the fixture, for example, open a network connection.**

This method is called before a test is executed.

- **Since** 1.4.0a2
- **Access** protected

*void* function tests\_phpDocumentorTParserGetInlineTagsTests::tearDown() [line 115]

**Tears down the fixture, for example, close a network connection.**

This method is called after a test is executed.

- **Since** 1.4.0a2
- **Access** protected

*void* function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineExampleWhenParsePrivateOff() [line 209]

**Shows correct behavior for handling an inline example tag e.g. {at-example path-to-file}**

There should be NO difference in results due to --parseprivate flag

- **Since** 1.4.0a2
- **Access** public

*void* function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineExampleWhenParsePrivateOn() [line 183]

**Shows correct behavior for handling an inline example tag e.g. {at-example path-to-file}**

There should be NO difference in results due to --parseprivate flag

- **Since** 1.4.0a2
- **Access** public

*void* function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineInternalWhenParsePrivateOff() [line 347]

**Shows correct behavior for handling an inline internal tag e.g. {at-internal blah-blah-blah}**

There SHOULD be differences in results due to --parseprivate flag

- **Since** 1.4.0a2
- **Access** public

*void* function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineInternalWhenParsePrivateOn()

n() [*line 331*]

**Shows correct behavior for handling an inline internal tag e.g. {at-internal blah-blah}**

There SHOULD be differences in results due to --parseprivate flag

This test demonstrates PEAR Bug #10871

- Since 1.4.0a2
- Access public

void function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineLinkWhenParsePrivateOff()  
[*line 162*]

**Shows correct behavior for handling an inline link tag e.g. {at-link URL link-text}**

There should be NO difference in results due to --parseprivate flag

- Since 1.4.0a2
- Access public

void function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineLinkWhenParsePrivateOn()  
[*line 142*]

**Shows correct behavior for handling an inline link tag e.g. {at-link URL link-text}**

There should be NO difference in results due to --parseprivate flag

- Since 1.4.0a2
- Access public

void function

tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineSourceWhenParsePrivateOff()  
() [*line 260*]

## **Shows correct behavior for handling an inline source tag e.g. {at-source}**

There should be NO difference in results due to --parseprivate flag

- Since 1.4.0a2
- Access public

*void* function  
tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineSourceWhenParsePrivateOn()  
() [line 236]

## **Shows correct behavior for handling an inline source tag e.g. {at-source}**

There should be NO difference in results due to --parseprivate flag

- Since 1.4.0a2
- Access public

*void* function  
tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineTutorialWhenParsePrivateOf  
f() [line 306]

## **Shows correct behavior for handling an inline tutorial tag e.g. {at-tutorial path-to-pkgfile}**

There should be NO difference in results due to --parseprivate flag

- Since 1.4.0a2
- Access public

*void* function  
tests\_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineTutorialWhenParsePrivateO  
n()  
n() [line 285]

## **Shows correct behavior for handling an inline tutorial tag e.g. {at-tutorial path-to-pkgfile}**

There should be NO difference in results due to --parseprivate flag

- **Since** 1.4.0a2
- **Access** public

## Class bug\_556894\_base [line 11]

### Base Class

- **Package** tests
- **Sub-Package** test1

**bug\_556894\_base::\$test**

*mixed = [line 16]*

**I'm a test var**

*void function bug\_556894\_base::test() [line 21]*

**I'm a test method**

## Class bug\_556894\_sub1 [line 32]

**Subclass in same subpackage**

- **Package** tests
- **Sub-Package** test1

## Class bug\_556894\_sub2 *[line 42]*

### **Subclass in different subpackage**

- **Package** tests
- **Sub-Package** test2



# Package XML\_Beautifier Procedural Elements

## Plain.php

**XML/Beautifier/Renderer/Plain.php**

- **Package** XML\_Beautifier
- **Author** Stephan Schmidt < [schst@php.net](mailto:schst@php.net)>

require\_once 'XML/Beautifier/Renderer.php' [line 34]

### Renderer base class

require\_once 'XML/Util.php' [line 29]

**XML\_Util is needed to create the tags**

# Package XML\_Beautifier Classes

## Class PHPDoc\_XML\_Beautifier\_Renderer\_Plain [line 45]

### Basic XML Renderer for XML Beautifier

- **Package** XML\_Beautifier
- **Author** Stephan Schmidt < [schst@php.net](mailto:schst@php.net)>
- **TODO** option to specify inline tags
- **TODO** automatically create <![CDATA[ ]]> sections
- **TODO** option to specify treatment of whitespace in data sections

*string* function PHPDoc\_XML\_Beautifier\_Renderer\_Plain::serialize(\$tokens) [line 54]

#### **Function Parameters:**

- **array \$tokens** XML tokens

### Serialize the XML tokens

- **Access** public



# Package phpDocumentor Procedural Elements

## actions.php

**phpDocumentor :: docBuilder Web Interface**  
PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [celog@php.net](mailto:celog@php.net)>
- **Author** Andrew Eddie
- **Version** CVS: \$Id: actions.php 212211 2006-04-30 22:18:14Z celog \$
- **Copyright** 2003-2006 Andrew Eddie, Greg Beaver
- **License** [LGPL](#)

include\_once "\$path/includes/utilities.php" [line 38]

include\_once "@WEB-DIR@/PhpDocumentor/docbuilder/includes/utilities.php" [line 36]

# builder.php

## phpDocumentor :: docBuilder Web Interface

Advanced Web Interface to phpDocumentor

PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Author** Andrew Eddie
- **Version** CVS: \$Id: builder.php 212211 2006-04-30 22:18:14Z cellog \$
- **Copyright** 2003-2006 Andrew Eddie, Greg Beaver
- **See** [phpdoc.php](#)
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

include "\$root\_dir/phpDocumentor/phpdoc.inc" [line [122](#)]

## phpdoc.inc

include\_once "\$root\_dir/phpDocumentor/common.inc.php" [line [69](#)]

## common file information

include\_once "\$root\_dir/phpDocumentor/common.inc.php" [line 52]

## common file information

# config.php

## phpDocumentor :: docBuilder Web Interface

Advanced Web Interface to phpDocumentor

PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Author** Andrew Eddie
- **Version** CVS: \$Id: config.php 234145 2007-04-19 20:20:57Z ashnazg \$
- **Copyright** 2003-2006 Andrew Eddie, Greg Beaver
- **See** [phpdoc.php](#)
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

include\_once "\$root\_dir/docbuilder/includes/utilities.php" [line [79](#)]

include\_once "\$root\_dir/phpDocumentor/common.inc.php" [line [78](#)]

### common file information

include\_once "@WEB-DIR@/PhpDocumentor/docbuilder/includes/utilities.php" [line [53](#)]

include\_once "PhpDocumentor/phpDocumentor/common.inc.php" [line 52]

## common file information

# utilities.php

## phpDocumentor :: docBuilder Web Interface

PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Author** Andrew Eddie
- **Version** CVS: \$Id: utilities.php 212211 2006-04-30 22:18:14Z celflog \$
- **Copyright** 2003-2006 Andrew Eddie, Greg Beaver
- **License** [LGPL](#)

*void function getDir(\$path, &\$node) [line 74]*

#### **Function Parameters:**

- **\$path**
- **&\$node**

*void function htmlArraySelect(&\$arr, \$select\_name, \$select\_atrrib, \$selected, \$arr) [line 63]*

#### **Function Parameters:**

- **array \$arr** array of the key-text pairs
- **string \$select\_name** The name of the select box

- **string \$select\_atrbs** Additional attributes to insert into the html select tag
- **string \$selected** The key value of the selected eleme
- **&\$arr**

**Returns a select box based on an key,value array where selected is based on key**

include\_once 'PhpDocumentor/HTML\_TreeMenu-1.1.2/TreeMenu.php' [line 36]

include\_once dirname(realpath(\_\_FILE\_\_))."/../../HTML\_TreeMenu-1.1.2/TreeMenu.php" [line 38]

void function recurseDir(\$path, &\$node) [line 96]

**Function Parameters:**

- **\$path**
- **&\$node**

void function showImage(\$src, [\$wid = ""], [\$hgt = ""], [\$alt = ""]) [line 48]

**Function Parameters:**

- **string \$src** path to the source image
- **int \$wid** width on the image [optional]
- **int \$hgt** height on the image [optional]
- **string \$alt** hover text for the image [optional]

**Allows png's with alpha transparency to be displayed in IE 6**

void function switchDirTree(\$path, &\$node) [line 125]

**Function Parameters:**

- **\$path**
- **&\$node**

void function vdump\_par(\$tree) [line 205]

**Function Parameters:**

- **\$tree**



# top.php

## phpDocumentor :: docBuilder Web Interface

Advanced Web Interface to phpDocumentor

PHP versions 4 and 5

Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Author** Andrew Eddie
- **Version** CVS: \$Id: top.php 212211 2006-04-30 22:18:14Z cellog \$
- **Copyright** 2003-2006 Andrew Eddie, Greg Beaver
- **See** [phpdoc.php](#)
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

include\_once "\$path/phpDocumentor/common.inc.php" [line [43](#)]

include\_once "PhpDocumentor/phpDocumentor/common.inc.php" [line [40](#)]

# new\_phpdoc.php

## Advanced Web Interface to phpDocumentor

PHP versions 4 and 5

- **Package** phpDocumentor
- **Author** Chuck Burgess < [ashnatz@php.net](mailto:ashnatz@php.net)>
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Juan Pablo Morales < [ju-moral@uniandes.edu.co](mailto:ju-moral@uniandes.edu.co)>
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **See** [phpdoc.php](#)
- **Link** <http://pear.php.net/package/PhpDocumentor>
- **Deprecated** redirects automatically to docbuilder (see docbuilder/index.html)
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

# phpdoc.php

Original Web Interface to **phpDocumentor**

PHP versions 4 and 5

- **Package** `phpDocumentor`
- **Author** Chuck Burgess < [ashnazg@php.net](mailto:ashnazg@php.net)>
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Juan Pablo Morales < [ju-moral@uniandes.edu.co](mailto:ju-moral@uniandes.edu.co)>
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Link** <http://pear.php.net/package/PhpDocumentor>
- **Deprecated** redirects automatically to `docbuilder` (see `docbuilder/index.html`)
- **TODO** CS cleanup - change package to `PhpDocumentor`
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

# Classes.inc

## Intermediate class parsing structure.

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2001-2007 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: Classes.inc 243933 2007-10-10 01:18:25Z ashnazg \$
- **Copyright** 2001-2007 Gregory Beaver
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- TODO CS cleanup - change package to PhpDocumentor
- Since 1.0rc1
- License [LGPL](#)
- Filesource [Source Code for this file](#)

# clone.inc.php

## Object clone method

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2001-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: clone.inc.php 243202 2007-09-30 02:08:08Z ashnazg \$
- **Copyright** 2001-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.0rc1
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

*object* the function phpDocumentor\_clone(\$obj) [/line [48](#)]

**Function Parameters:**

- *object* **\$obj** the object to be cloned

## Clone an object in PHP 4

- **TODO** CS cleanup - rename function to PhpDocumentor\_clone

# clone5.inc.php

## Object clone method

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2001-2006 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: clone5.inc.php 243202 2007-09-30 02:08:08Z ashnazg \$
- **Copyright** 2001-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.0rc1
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

# common.inc.php

## Common information needed by all portions of the application

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2001-2008 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: common.inc.php 288074 2009-09-05 02:16:26Z ashnazg \$
- **Copyright** 2001-2008 Gregory Beaver
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename constant to TOKENIZER\_EXT
- **Since** 1.0rc1
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

void function debug(\$s) [[line 266](#)]

#### Function Parameters:

- **string \$s** the "debug message" string to echo out

## Debugging output

- **TODO CS Cleanup** - can't avoid "prefixed by package" error

*void* function fancy\_debug(*\$s, \$v*) [line [279](#)]

**Function Parameters:**

- *string* ***\$s*** string to display
- *mixed* ***\$v*** unlimited number of variables to display with var\_dump()

**Returns a formatted var\_dump for debugging purposes.**

*PATH\_DELIMITER* = '/' [line [62](#)]

*array* function phpDocumentor\_ConfigFileList(*\$directory*) [line [97](#)]

**Function Parameters:**

- *string* ***\$directory*** a directory string

**used in `phpdoc.php` and `new_phpdoc.php`**

- **TODO CS cleanup** - rename function to PhpDocumentor\_ConfigFileList

*string* function phpDocumentor\_get\_class(*\$object*) [line [304](#)]

**Function Parameters:**

- *object* ***\$object*** the object to get the classname for

**Returns a lower-cased version of `get_class` for PHP 5**

`get_class()` returns case as declared in the file in PHP 5

- **TODO** CS cleanup - rename function to PhpDocumentor\_get\_class

array function phpDocumentor\_parse\_ini\_file(\$filename, [\$process\_sections = false]) [line [146](#)]

**Function Parameters:**

- *string* **\$filename** full path to the ini file
- *bool* **\$process\_sections** add an associative index  
for each section [in brackets]

### Parse an .ini file

Works like [http://www.php.net/parse\\_ini\\_file](http://www.php.net/parse_ini_file), except it will take a section like:

```
[MYVAR]
value1
value2
value3
```

and return an associative array(MYVAR => array(value1, value2, value3))

- **TODO** CS cleanup - rename function to PhpDocumentor\_parse\_ini\_file

PHPDOCUMENTOR\_VER = "1.4.3" [line [49](#)]

PHPDOCUMENTOR\_WEBSITE = "http://www.phpdoc.org" [line [53](#)]

PHPDOCUMENTOR\_WINDOWS = substr(PHP\_OS,0,3)=='WIN' [line [64](#)]

require\_once \$cloneClassDir.DIRECTORY\_SEPARATOR.\$cloneClassFile [line [81](#)]

SMART\_PATH\_DELIMITER = DIRECTORY\_SEPARATOR [line [56](#)]

tokenizer\_ext = extension\_loaded('tokenizer')&&version\_compare(phpversion(),"4.3.0",">=") [line [58](#)]

\_IN\_PHP5 = phpversion()=='5.0.0RC1-dev'||phpversion()=='5.0.0RC2-dev'||version\_compare(phpversion(),'5.0.0','ge') [line [66](#)]

# EventStack.inc

An Event Stack for inter-program communication, particularly for parsing  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2007 Joshua Eichorn

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** CVS: \$Id: EventStack.inc 243937 2007-10-10 02:27:42Z ashnazg \$
- **Copyright** 2000-2007 Joshua Eichorn
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 0.1
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

# IntermediateParser.inc

## The phpDocumentor\_IntermediateParser Class

The Intermediary Data Parser (intermediate between Parse and Converter)

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** [phpDocumentor](#)
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: IntermediateParser.inc 247821 2007-12-09 06:11:35Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.1
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

# lo.inc

## File and input handling routines

This class parses command-line options, and works with files to generate lists of files to parse based on the ignore/include options

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Gregory Beaver < [celfog@php.net](mailto:celfog@php.net)>
- **Version** CVS: \$Id: lo.inc 286921 2009-08-08 05:01:24Z ashnazg \$
- **Copyright** 2000-2006 Joshua Eichorn, Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 0.1
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

*void* function get\_include\_path() [*line 987*]  
*void* function loinc\_mystrucsort(\$a, \$b) [*line 913*]

### Function Parameters:

- *string* \$a

- *string \$b*

## Sorting functions for the file list

*void function loinc\_sortfiles(\$a, \$b) [line [908](#)]*

**Function Parameters:**

- *string \$a*
- *string \$b*

## Sorting functions for the file list

*array function setup\_dirs(\$struc, \$dir, \$contents) [line [956](#)]*

**Function Parameters:**

- *array \$struc* struc is array('dir' => array of files in dir,'dir/subdir' => array of files in dir/subdir,...)
- *array \$dir* array form of 'dir/subdir/subdir2' array('dir','subdir','subdir2')
- *\$contents*

## Recursively move contents of \$struc into associative array

The contents of \$struc have many indexes like 'dir/subdir/subdir2'. This function converts them to array('dir' => array('subdir' => array('subdir2')))

- **Used by** [lo::getDirTree\(\)](#)

*array function set\_dir(\$dir, \$contents) [line [934](#)]*

**Function Parameters:**

- *array \$dir*
- *array \$contents*

## Recursively add all the subdirectories of \$contents to \$dir without erasing anything

**in**  
\$dir

# ParserDescCleanup.inc

All of the functions to clean up and handle the long description of a DocBlock are in this file.

The primary functionality is based on Parser and WordParser, and modified to recognize only the tokens defined in the PHPDOCUMENTOR\_PDP\_\* constants

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: ParserDescCleanup.inc 286923 2009-08-08 06:00:39Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **See** [Parser](#), [WordParser](#)
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.2
- **License** [LGPL](#)

PHPDOCUMENTOR\_PDP\_EVENT\_B = 605 [/line 68]

**when <b> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_BR = 607 [line 76]

**when <br> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_CODE = 600 [line 48]

**when <code> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_DOUBLECR = 602 [line 56]

**when \n\n is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_ESCAPE = 608 [line 80]

**when the <> potential escape for tags is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_ESCAPE\_CODE = 610 [line 88]

**when << /code>> is found in a <code></code> section**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_ESCAPE\_PRE = 609 [line 84]

**when << /pre>> is found in a <pre></pre> section**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_I = 606 [line 72]

**when <i> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_KBD = 613 [line 100]

**when <kbd> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_LIST = 604 [line 64]

**when <ul>/<ol> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_P = 601 [line 52]

**when <p> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_PRE = 603 [line 60]

**when <pre> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_SAMP = 612 [line 96]

**when <samp> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_EVENT\_SIMLIST = 614 [line 111]

**when a simple list is found in a desc**

[parserDescParser](#) token constants like o item 1

o item 2

PHPDOCUMENTOR\_PDP\_EVENT\_VAR = 611 [line 92]

**when <var> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_B = 705 [line 70]

**when <b> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_BR = 707 [line 78]

**when <br> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_CODE = 700 [line 50]

**when <code> is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_DOUBLECR = 702 [line 58]

**when \n\n is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_ESCAPE = 708 [line 82]

**when the <> potential escape for tags is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_ESCAPE\_CODE = 710 [line 90]

**when << /code>> is found in a <code></code> section**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_ESCAPE\_PRE = 709 [line 86]

**when << /pre>> is found in a <pre></pre> section**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_I = 706 [line 74]

**when <i> is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_KBD = 713 [line 102]

**when <kbd> is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_LIST = 704 [line 66]

**when <ul>/<ol> is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_P = 701 [line 54]

**when <p> is found in a desc**  
[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_PRE = 703 [line 62]

**when <pre> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_SAMP = 712 [line 98]

**when <samp> is found in a desc**

[parserDescParser](#) token constants

PHPDOCUMENTOR\_PDP\_STATE\_SIMLIST = 714 [line 120]

**when a simple list is found in a desc**

[parserDescParser](#) token constants like

- o item 1
- o item 2

PHPDOCUMENTOR\_PDP\_STATE\_VAR = 711 [line 94]

**when <var> is found in a desc**

[parserDescParser](#) token constants

# phpdoc.inc

## startup file

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2007 Joshua Eichorn, Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: phpdoc.inc 243933 2007-10-10 01:18:25Z ashnazg \$
- **Copyright** 2000-2007 Joshua Eichorn, Gregory Beaver
- **Link** <http://www.phpdoc.org>
- **Link** <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 0.1
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

require\_once "[phpDocumentor/Setup.inc.php](#)"[the 60]

**All command-line handling from previous version has moved to here**  
Many settings also moved to phpDocumentor.ini

# ProceduralPages.inc

## Intermediate procedural page parsing structure.

This structure parses defines, functions, and global variables by file, and then iterates over the elements to document conflicts.

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: ProceduralPages.inc 253641 2008-02-24 02:35:44Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.1
- **License** [LGPL](#)

**Publisher.inc**  
**a class for handling the publishing of data**  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2007 Kellin, Joshua Eichorn

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Kellin < [passionplay@hotmail.com](mailto:passionplay@hotmail.com)>
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** CVS: \$Id: Publisher.inc 244033 2007-10-11 03:30:34Z ashnazg \$
- **Copyright** 2000-2007 Kellin, Joshua Eichorn
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 0.1
- **License** [LGPL](#)

# Setup.inc.php

This was all in [phpdoc.inc](#) and now encapsulates the complexity  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: Setup.inc.php 258122 2008-04-22 15:48:55Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.2
- **License** [LGPL](#)

## \$interface

*array = \$\_GET['interface'] [line 75]*

**\$interface is either 'web' or is not set at all**

## \$phpDocumentor\_DefaultCategoryName

*string = 'default' [line 95]*

## **default package name, set using -dn --defaultcategoryname**

- **Name \$phpDocumentor\_DefaultCategoryName**

### **\$phpDocumentor\_DefaultPackageName**

*string = 'default' [line 88]*

## **default package name, set using -dn --defaultpackagename**

- **Name \$phpDocumentor\_DefaultPackageName**

### **\$\_phpDocumentor\_setting**

*array = \$\_GET['setting'] [line 80]*

## **\$\_phpDocumentor\_setting is either the value from the web interface, or is set up by [lo::parseArgv\(\)](#)**

*void function checkForBugCondition(\$php\_version, [\$php\_bug\_number = 'none'], [\$pear\_bug\_number = 'none']) [line 971]*

### **Function Parameters:**

- **string \$php\_version** the PHP version that contains the bug
- **string \$php\_bug\_number** the PHP bug number (if any)
- **string \$pear\_bug\_number** the PEAR bug number (if any)

## **Crash in case of known, dangerous bug condition**

Checks the PHP version that is executing PhpDocumentor, in case a known PHP/PEAR bug condition could be triggered by the PhpDocumentor execution.

*boolean function decideOnOrOff([\$value\_to\_guess = 'NO VALUE WAS PASSED']) [line 930]*

### **Function Parameters:**

- *string \$value\_to\_guess* the command-line option to analyze

## Fuzzy logic to interpret the boolean args' intent

```
include_once "phpDocumentor/ParserDescCleanup.in[the 63]  
  
include_once "phpDocumentor/TutorialHighlightParser.in[the 62]  
  
include_once "phpDocumentor/HighlightParser.in[the 61]  
  
include_once "phpDocumentor/PackagePageElements.in[the 64]  
  
include_once "phpDocumentor/Errors.in[the 68]  
  
include_once "phpDocumentor/phpDocumentorTParser.in[the 60]  
  
include_once "phpDocumentor/Converter.in[the 67]  
  
include_once "phpDocumentor/LinkClasses.in[the 66]  
  
include_once "phpDocumentor/XMLpackagePageParser.in[the 65]  
  
include_once "phpDocumentor/phpDocumentorTWordParser.in[the 59]  
  
include_once "phpDocumentor/IntermediateParser.in[the 49]  
  
include_once "phpDocumentor/WordParser.in[the 50]  
  
include_once "phpDocumentor/ProceduralPages.in[the 48]  
  
include_once "phpDocumentor/Classes.in[the 47]  
  
include_once "phpDocumentor/lo.in[the 45]  
  
include_once "phpDocumentor/Publisher.in[the 46]  
  
include_once "phpDocumentor/EventStack.in[the 51]
```

```
include_once "phpDocumentor/ParserData.inc" [line 52]  
  
include_once "phpDocumentor/ParserDocBlock.inc" [line 56]  
  
include_once "phpDocumentor/ParserElements.inc" [line 57]  
  
include_once "phpDocumentor/Parser.inc" [line 58]  
  
include_once "phpDocumentor/DescHTML.inc" [line 55]  
  
include_once "phpDocumentor/DocBlockTags.inc" [line 54]  
  
include_once "phpDocumentor/common.inc.php" [line 43]
```

## common settings

```
include_once "phpDocumentor/InlineTags.inc" [line 53]
```

*void* function phpDocumentor\_out(\$string) [line 951]

**Function Parameters:**

- \$string

## Print parse information if quiet setting is off

# HighlightParserTests.php

- **Package** phpDocumentor

```
require_once 'HighlightParserGetInlineTagsTests.php' [line 10]
```

```
require_once 'PHPUnit/TextUI/TestRunner.php' [line 7]
```

```
require_once 'PHPUnit/Framework.php' [line 6]
```

# IntermediateParserTests.php

- **Package** phpDocumentor

```
require_once 'IntermediateParserAddPrivatePageTests.php' [line 10]
```

```
require_once 'PHPUnit/TextUI/TestRunner.php' [line 7]
```

```
require_once 'PHPUnit/Framework.php' [line 6]
```

# ParserClassTests.php

- **Package** phpDocumentor

```
require_once 'ParserClassGetSourceLocationTests.php' [line 10]
```

```
require_once 'PHPUnit/TextUI/TestRunner.php' [line 7]
```

```
require_once 'PHPUnit/Framework.php' [line 6]
```

# ParserPageTests.php

- **Package** phpDocumentor

```
require_once 'ParserPageGetSourceLocationTests.php' [line 10]
```

```
require_once 'PHPUnit/TextUI/TestRunner.php' [line 7]
```

```
require_once 'PHPUnit/Framework.php' [line 6]
```

# phpDocumentorSetupTests.php

- **Package** phpDocumentor

```
require_once 'phpDocumentorSetupDecideOnOrOffTests.php' [line 11]
```

```
require_once 'phpDocumentorSetupCleanConverterNamePieceTests.php' [line 10]
```

```
require_once 'PHPUnit/TextUI/TestRunner.php' [line 7]
```

```
require_once 'PHPUnit/Framework.php' [line 6]
```

# phpDocumentorTParserTests.php

- **Package** phpDocumentor

```
PHPUnit_MAIN_METHOD = 'phpDocumentorTParserTests::main' [line 3]
require_once 'phpDocumentorTParserGetInlineTagsTests.php' [line 10]
```

```
require_once 'PHPUnit/TextUI/TestRunner.php' [line 7]
```

```
require_once 'PHPUnit/Framework.php' [line 6]
```

# Package phpDocumentor Classes

## Class bug\_772441 [line 2]

- **Package** phpDocumentor

## Class Classes [line 70]

**Intermediate class parsing structure.**

The [phpDocumentor IntermediateParser](#) class uses this class and its cousin, [ProceduralPages](#) to organize all parsed source code elements. Data is fed to each immediately after it is parsed, and at conversion time, everything is organized.

The Classes class is responsible for all inheritance, including resolving name conflicts between classes, determining which classes extend other classes, and is responsible for all inheritance of documentation.

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2001-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>

- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.0rc1
- **License** [LGPL](#)

`void function Classes::addClass(&$element) [line 393]`

**Function Parameters:**

- [parserClass](#) &\$element element is a [parserClass](#)

**While parsing, add a class to the list of parsed classes**

sets up the \$classesbyfile, \$classesbynamefile, \$extendsbyfile, \$classchildrenbyfile, \$roots arrays, and sets \$curclass

- **Uses** [Classes::addPackageToFile\(\)](#) - marks the current class's package as being present in a file

`void function Classes::addConst(&$element) [line 466]`

**Function Parameters:**

- [parserConst](#) &\$element element is a [parserConst](#)

**While parsing, add a variable to the list of parsed variables**

sets up the \$constsbyfile array using \$curfile and \$curclass

`void function Classes::addMethod(&$element) [line 434]`

**Function Parameters:**

- [parserMethod](#) &\$element element is a [parserMethod](#)

**While parsing, add a method to the list of parsed methods**

sets up the \$methodsbyfile array using \$curfile and \$curclass

*void* function Classes::addPackageToFile(\$package) [line 497]

**Function Parameters:**

- *string* **\$package** package name

**Mark a package as being used in a class**

```
1  function addPackageToFile($package)
2  {
3      if (!isset( $this)    revcpbf[$this]    curfile]$package))
4      $this    classpackagebyfile[ $this]    curfile][] = $package
5      $this    revcpbf[$this]    curfile]$package  =  1;
6  }
```

- **Used by** [Classes::addClass\(\)](#) - marks the current class's package as being present in a file

*void* function Classes::addVar(&\$element) [line 450]

**Function Parameters:**

- [parserVar](#) &\$element element is a [parserVar](#)

**While parsing, add a variable to the list of parsed variables**

sets up the \$varsbyfile array using \$curfile and \$curclass

*parserClass* function Classes::getClass(\$class, \$file) [line 943]

**Function Parameters:**

- *string* **\$class** classname
- *string* **\$file** file classname is located in

**Get the parserClass representation of a class from its name and file**

*mixed* function Classes::getClassByPackage(\$class, \$package) [line [1189](#)]

**Function Parameters:**

- *string* **\$class** classname
- *string* **\$package** package classname is in

## Search for a class in a package

*mixed* function Classes::getClassesInPath(\$path) [line [960](#)]

**Function Parameters:**

- *string* **\$path** full path to filename

## Used by [parserData::getClasses\(\)](#) to retrieve classes defined in file \$path

retrieves the array entry from \$classesbyfile for \$path

*mixed* function Classes::getConflicts(\$class) [line [630](#)]

**Function Parameters:**

- *mixed* **\$class** the class name to search for

## If a package contains two classes with the same name, this function finds that conflict

Returns the \$classconflicts entry for class \$class, minus its own path

*mixed* function Classes::getDefiniteChildren(\$parclass, \$file) [line [1349](#)]

**Function Parameters:**

- *string* **\$parclass** name of parent class
- *string* **\$file** file parent class is found in

## Get all classes confirmed in parsing to be descended class \$parclass in file \$file

- See [parserClass::getChildClassList\(\)](#)
- Uses \$definitechild

mixed function Classes::getParentClass(\$class, \$file) [line [1227](#)]

**Function Parameters:**

- *string* **\$class** classname
- *string* **\$file** file classname is located in

### Find the parent class of a class in file \$file

uses 3 tests to find the parent classname:

1. only one class with the parent classname
2. more than one class, but only one in the same file as the child
3. only one parent class in the same package as the child

- Used by [Classes::setClassParent\(\)](#) - to find the parent class

array function Classes::getRoots([\$all = false]) [line [1292](#)]

**Function Parameters:**

- *boolean* **\$all** [since phpDocumentor 1.3.0RC6] determines whether to return class trees that extend non-parsed classes

### Get a list of all root classes indexed by package. Used to generate class trees by [Converter](#)

void function Classes::Inherit(&\$render) [line [546](#)]

**Function Parameters:**

- [phpDocumentor\\_IntermediateParser](#) &\$render the renderer object

## Main processing engine for setting up class inheritance.

This function uses \$roots to traverse the inheritance tree via [processChild\(\)](#) and returns the data structures `phpDocumentor_IntermediateParser` needs to convert parsed data to output using [phpDocumentor\\_IntermediateParser::Convert\(\)](#)

- **TODO** CS Cleanup - rename to "inherit" for CamelCaps naming standard
- **Uses** [Classes::processChild\(\)](#) - set up inheritance

*void* function Classes::nextFile(\$file) [[line 482](#)]

**Function Parameters:**

- **string** **\$file** file currently being parsed

## Prepare to parse a new file

sets \$curfile to \$file and \$curclass to false (no class being parsed)

*void* function Classes::processChild(&\$render, \$class, \$file, [\$furb = false]) [[line 690](#)]

**Function Parameters:**

- [phpDocumentor\\_IntermediateParser](#) &**\$render** the renderer object
- **string** **\$class** class to process
- **string** **\$file** name of file \$class
- **boolean** **\$furb** flag used privately
  - processing
  - is located in
  - output while parsing
  - leftover classes in
  - to control informational
  - (used when
  - [Inherit\(\)](#)

**This function recursively climbs up the class tree, setting inherited information like package and adds the elements to [phpDocumentor\\_IntermediateParser](#).**

Using structures defined in [Classes](#), the function first sets package information, and then seeks out child classes. It uses 3 tests to determine whether a class is a child class.

1. child class is in the same file as the parent class  
and extends parent class
2. child class is in a different file and specifies

- the parent's @package in its docblock
3. child class is in a different file and is in a different @package, with one possible parent class

- **Global Variable Used** string [\\$phpDocumentor\\_DefaultPackageName](#): default package, usually "default"
- **Usedby** [Classes::Inherit\(\)](#) - set up inheritance

*void* function Classes::setClassParent(\$class, \$file) [[line 518](#)]

**Function Parameters:**

- *string* **\$class** child class to find parent class
- *string* **\$file** file child class is located in

**Find the parent class of \$class, and set up structures to note this fact**

Modifies the [parserClass](#) element in \$classesbyfile to use the parent's package, and inherit methods/vars

- **Uses** [Classes::getParentClass\(\)](#) - to find the parent class
- **Uses** \$definitechild - if a match is made between a parent class and parameter \$class in file \$file, then definitechild is set here

## Class EventStack

[[line 52](#)]

### An event Stack

- **Package** phpDocumentor
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** Release: @VER@
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [GPL](#)

### EventStack::\$num

*integer = 0 [line 64]*

#### The number of events in the stack

### EventStack::\$stack

*array = array(PARSER\_EVENT\_NOEVENTS) [line 58]*

#### The stack

*int function EventStack::getEvent() [line 94]*

#### Get the current event

*int function EventStack::popEvent() [line 83]*

#### Pop an event from the stack

*void function EventStack::pushEvent(\$event) [line 73]*

##### Function Parameters:

- *int \$event* All events must be constants

#### Push an event onto the stack

## Class Io

*[line 50]*

## Class to handle file and user io opperations

- **Package** phpDocumentor
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** \$Id: lo.inc 286921 2009-08-08 05:01:24Z ashnazg \$

### lo::\$ignore

*false|array = [line 63]*

**Format:** array(array(**regexp-ready string to search for whole path**,  
**regexp-ready string to search for basename of ignore strings**),...)

### lo::\$phpDocOptions

*mixed = [line 57]*

**Holds all the options that are avaiable to the cmd line interface**  
and to the different web interfaces

### lo::\$valid\_booleans

```
array = array
(
    "", ' ', 'on', 'y', 'yes', 'true', '1',
    'off', 'n', 'no', 'false', '0'
)
```

*) [line 71]*

**A specific array of values that boolean-based arguments can understand, aided by  
the [decideOnOrOff\(\)](#) helper method.**  
Use lowercase letters always, to simplify string comparisons

Constructor **void** function **lo::lo()** *[line 87]*

**creates an array **\$this->phpDocOptions** and sets program options in it.**

Array is in the format of: **[filename][tag][] = "f";**  
**[filename][tag][] = "-file";**  
**[filename][desc] "name of file to parse"**

*bool* function **lo::checkIgnore(\$file, \$path, \$ignore, [\$ignore\_no\_ext = true], [\$ignoresymlinks = false])** [/line [717](#)]

**Function Parameters:**

- *string* **\$file** just the file name of the file or directory, this is the last dir in the case of directories
- *string* **\$path** the path to consider (should be checked by and may be relative) realpath() before,
- *array* **\$ignore**
- *bool* **\$ignore\_no\_ext**
- *bool* **\$ignoresymlinks** ignore symlinks?

**Tell whether to ignore a file or a directory allows \* and ? wildcards**

- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Usedby** [lo::getDirTree\(\)](#)

*array* function **lo::dirList(\$orig\_directory, [\$hidden = false], [\$ignore\_symlinks = false], \$directory)** [/line [377](#)]

**Function Parameters:**

- *string* **\$directory** full path to the directory you want the list of
- *bool* **\$hidden** whether to list files that begin with . like .bash\_history
- *bool* **\$ignore\_symlinks** whether to ignore symlinks
- *\$orig\_directory*

- **Usedby** [lo::getAllFiles\(\)](#)
- **Usedby** [lo::getDirTree\(\)](#)

*string* function **lo::displayHelpMsg()** [/line [204](#)]

**create the help message for display on the command-line**

*mixed* function **lo::getAllFiles(\$file)** [/line [888](#)]

**Function Parameters:**

- *string \$file* a full path from the -f command-line parameter, with potential \* and ? wildcards.

## Take a filename with wildcards and return all files that match the wildcards

- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Uses** [Io::removeNonMatches\(\)](#)
- **Uses** [Io::dirList\(\)](#)

*void function Io::getBase(\$filelist) [/line 444]*

### Function Parameters:

- *array \$filelist* array of strings

## Retrieve common directory (case-insensitive in windows)

takes the list of files, and returns the subdirectory they share in common, so in this list:

```

1      array(
2          "/dir1/dir2/subdir/dir3/filename.ext"
3          "/dir1/dir2/subdir/dir4/filename.ext"
4          "/dir1/dir2/mydir/dir5/filename.ext"
);
```

getBase will return "/dir1/dir2"

*void function Io::getDirTree(\$dir, \$base\_dir, [\$ignore = array()], [\$hidden = false], [\$ignoresymlinks = false]) [/line 564]*

### Function Parameters:

- *string \$dir* directory
- *string \$base\_dir* base directory
- *array \$ignore* array of ignored items
- *boolean \$hidden* the "hidden" flag
- *boolean \$ignoresymlinks* the "ignoresymlinks" flag

- **Uses** [setup\\_dirs\(\)](#)
- **Uses** [lo::dirList\(\)](#)
- **Uses** [lo::checkIgnore\(\)](#)

array function lo::getReadmeInstallChangelog(\$base, \$filelist) [[line 537](#)]

**Function Parameters:**

- *string* **\$base** base directory from [getBase\(\)](#)
- *array* **\$filelist** file list from [dirList\(\)](#)

string function lo::getRegExableSearchString(\$s) [[line 832](#)]

**Function Parameters:**

- *string* **\$s** string with wildcards ? and \*

**Converts \$s into a string that can be used with preg\_match**

- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>

array function lo::getTutorials(\$filelist) [[line 493](#)]

**Function Parameters:**

- *array* **\$filelist** array of paths (strings)

**Retrieve tutorial subdirectories and their contents from the list of files to parse**

boolean function lo::isIncludeable(\$filename) [[line 245](#)]

**Function Parameters:**

- *string* **\$filename**

calls [http://www.php.net/file\\_exists](http://www.php.net/file_exists) for each value in include\_path,  
then calls [http://www.php.net/is\\_readable](http://www.php.net/is_readable) when it finds the file

array function lo::parseArgv() [line [277](#)]

Parses `$_SERVER['argv']` and creates a setup array

- **Global Variable Used** array `$argv`: command-line arguments
- **TODO** replace with `Console_*` ?

void function lo::readPhpFile(\$path, [\$quietMode = false]) [line [648](#)]

**Function Parameters:**

- *string* `$path`
- `$quietMode`

Reads a file and returns it as a string Does basic error checking

file extensions are set in [phpdoc.inc](#)

- **Global Variable Used** array `$_phpDocumentor_cvspHPfile_exts`: PHP File extensions, used to validate that `$path` is a PHP File
- **Global Variable Used** array `$_phpDocumentor_phpfile_exts`: PHP File extensions in a CVS repository, used to validate that `$path` is a PHP File

*string/array* function lo::removeNonMatches(\$dir, \$match) [line [859](#)]

**Function Parameters:**

- `array` `$dir` array of filenames (full path)
- *string* `$match` search string with wildcards

**Removes files from the \$dir array that do not match the search string in \$match**

- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Usedby** [Io::getAllFiles\(\)](#)

*void function Io::\_setupIgnore(\$ignore) [line 796]*

**Function Parameters:**

- **array \$ignore** strings of files/paths/wildcards to ignore

**Construct the [\\$ignore](#) array**

- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Access** protected

## Class phpDocumentor\_IntermediateParser

*[line 60]*

**The phpDocumentor\_IntermediateParser Class**

This class performs the work of organizing raw data from the parser in the format of descendants of the [parserElement](#) class. This is also where processing of package pages occurs, in [phpDocumentor\\_IntermediateParser::handleClass\(\)](#) for class-level packages and [phpDocumentor\\_IntermediateParser::handleDocBlock\(\)](#) for page-level packages.

Most of the work of this parser goes to matching up DocBlocks with the elements that they are documenting. Since DocBlocks are passed before the element they document, the last DocBlock is stored in [phpDocumentor\\_IntermediateParser::\\$last](#) and then placed into the \$docblock parameter of the parserElement descendant object.

- **Package** phpDocumentor
- **Author** Gregory Beaver
- **Version** \$Id: IntermediateParser.inc 247821 2007-12-09 06:11:35Z ashnazg \$
- **Copyright** 2002 Gregory Beaver

### **phpDocumentor\_IntermediateParser::\$all\_packages**

*array = array() [line [186](#)]*

**list of all packages encountered while documenting. Used in automatic linking.**

Converter::getLink() first checks if an ambiguous link is found in the current package. If not, it then checks in parent packages, and if still not found, uses this array to check in the rest of the packages before giving up

- **Var Format:** array(packagename => 1, packagename => 1,...)
- **See** [Converter::getLink\(\)](#)

### **phpDocumentor\_IntermediateParser::\$classes**

*Classes = false [line [268](#)]*

**used to keep track of inheritance at the smartest level possible for a dumb computer**

### **phpDocumentor\_IntermediateParser::\$converters**

*array = false [line [287](#)]*

**an array of template names indexed by converter name**

For example, if the default HTMLframesConverter is using the DOM/10133t template, the array will be

1      *\$converters['frames'] = 'DOM/10133t'*

- Var Format: array(Convertername1 => templatename)
- See [Converter](#)

### **phpDocumentor\_IntermediateParser::\$cur\_class**

*string = " [line [88](#)]*

#### **Name of the class currently being parsed.**

It is only used (and only valid) when phpDocumentor\_IntermediateParser is parsing a class

### **phpDocumentor\_IntermediateParser::\$data**

*parserData = [line [237](#)]*

#### **\$data contains parsed structures for the current page being parsed**

In version 1.1+, \$data is only used to store the current page information. All handling of documented elements is handled by the [ProceduralPages](#) and [Classes](#) classes.

### **phpDocumentor\_IntermediateParser::\$db\_template**

*mixed = [line [295](#)]*

### **phpDocumentor\_IntermediateParser::\$event\_handlers**

*mixed = array(  
 'docblock' => 'handleDocBlock',  
 'page' => 'handlePage',  
 'class' => 'handleClass',  
 'define' => 'handleDefine',  
 'function' => 'handleFunction',  
 'method' => 'handleMethod',  
 'var' => 'handleVar',  
 'const' => 'handleConst',  
 'packagepage' => 'handlePackagePage',  
 'include' => 'handleInclude',  
 'global' => 'handleGlobal',  
 'tutorial' => 'handleTutorial',  
) [line [214](#)]*

### **the functions which handle output from [Parser](#)**

- See [phpDocumentor\\_IntermediateParser::handlePackagePage\(\)](#),  
[phpDocumentor\\_IntermediateParser::handleInclude\(\)](#),

- [phpDocumentor\\_IntermediateParser::handleTutorial\(\)](#)
- See [phpDocumentor\\_IntermediateParser::handleDefine\(\)](#),  
[phpDocumentor\\_IntermediateParser::handleFunction\(\)](#),  
[phpDocumentor\\_IntermediateParser::handleMethod\(\)](#),  
[phpDocumentor\\_IntermediateParser::handleVar\(\)](#)
- See [handleEvent\(\)](#), [phpDocumentor\\_IntermediateParser::handleDocBlock\(\)](#),  
[phpDocumentor\\_IntermediateParser::handlePage\(\)](#),  
[phpDocumentor\\_IntermediateParser::handleClass\(\)](#)

### **phpDocumentor\_IntermediateParser::\$last**

*parserDocBlock = [line 65]*

### **phpDocumentor\_IntermediateParser::\$lasttype**

*string = " [line 80]*

#### **type of the last parser Element handled**

This is used in handleDocBlock to determine whether a DocBlock is a page-level DocBlock in conjunction with the [parserData::\\$clean](#) var. A page-level DocBlock is always the first DocBlock in a file, and must be followed by another DocBlock. The first test is handled by parserData::\$clean, which is set to false on the first encounter of an element, and the second test is handled by this variable, which must be equal to "docblock"

- See [phpDocumentor\\_IntermediateParser::handleDocBlock\(\)](#)

### **phpDocumentor\_IntermediateParser::\$packagecategories**

*array = array() [line 173]*

#### **Used to determine the category for tutorials.**

**WARNING:** If more than one category exists, the last category encountered will overwrite the previous and will raise a big warning

- **Var Format:** packagename => categoryname

### **phpDocumentor\_IntermediateParser::\$packageoutput**

*false|array = false [line 206]*

**array of packages to parser and output documentation for, if not all packages should be documented**

Format:

array(package1,package2,...)

or false if not set

Use this option to limit output similar to ignoring files. If you have some temporary files that you don't want to specify by name but don't want included in output, set a package name for all the elements in your project, and set packageoutput to that name. the default package will be ignored. Parsing speed does not improve. If you want to ignore files for speed reasons, use the ignore command-line option

- [Tutorial -po, --packageoutput](#)
- See [lo](#)

### **phpDocumentor\_IntermediateParser::\$package\_pages**

*array = array() [line 133]*

**array of parsed package pages**

used by [Convert\(\)](#) to convert all package pages into output

### **phpDocumentor\_IntermediateParser::\$package\_parents**

*array = array() [line 164]*

**Keeps track of packages of classes that have parent classes in another package.**

**Used in automatic linking.**

This array is updated by [addPackageParent\(\)](#), which is called in [Classes::processChild\(\)](#) to keep track of classes that descend from classes in different packages. In other words, if class foo is in package one, and class bar is in package two, an entry \$package\_parents['two'] = 'one' will be made.

- **Var** Format: packagename => parentpackagename
- **See** [Converter::getLink\(\)](#)

### **phpDocumentor\_IntermediateParser::\$pages**

*array = array() [line [138](#)]*

- **Var** array of all [parserData](#) containing page information

### **phpDocumentor\_IntermediateParser::\$parsePrivate**

*boolean = false [line [106](#)]*

#### **set in [Setup.inc.php](#) to the value of the parseprivate commandline**

option. If this option is true, elements with an @access private tag will be parsed and displayed

- **Tutorial** [-pp, --parseprivate](#)

### **phpDocumentor\_IntermediateParser::\$privatelpages**

*array = array() [line [151](#)]*

#### **Put away a page that has been @ignored or @access private [\\$parsePrivate](#)**

When a page has @access private in its DocBlock, it is placed here instead of in [\\$pages](#), to allow for proper Class parsing. Since classes and pages are parsed as if they were separate, this array allows public classes on private pages to retrieve information needed about the page that holds the class and to [addPageIfNecessary\(\)](#) to the \$pages array

### **phpDocumentor\_IntermediateParser::\$private\_class**

*mixed = false [line [113](#)]*

**this variable is used to prevent parsing of elements with an @ignore tag**

- See [phpDocumentor\\_IntermediateParser::\\$parsePrivate](#)
- See [phpDocumentor\\_IntermediateParser::\\$packageoutput](#)

### **phpDocumentor\_IntermediateParser::\$proceduralpages**

*ProceduralPages = false [line [276](#)]*

**used to keep track of all elements in a procedural page. Handles name conflicts with elegance**

- Since 1.1

### **phpDocumentor\_IntermediateParser::\$quietMode**

*boolean = false [line [248](#)]*

**set in [Setup.inc.php](#) to the value of the quitemode commandline option.**

If this option is true, informative output while parsing will not be displayed (documentation is unaffected)

- Tutorial [-q, --quiet](#)

### **phpDocumentor\_IntermediateParser::\$ric**

*array = array() [line [303](#)]*

**Stores parsed CHANGELOG/INSTALL/README files**

- Var Format: array(CHANGELOG => contents, INSTALL => contents,

### **phpDocumentor\_IntermediateParser::\$targetDir**

*mixed = [line [119](#)]*

**used to set the output directory**

- See [phpDocumentor\\_IntermediateParser::setTargetDir\(\)](#)

### **phpDocumentor\_IntermediateParser::\$templateBase**

*mixed = [line [125](#)]*

**used to set the template base directory**

- See [phpDocumentor\\_IntermediateParser::setTemplateBase\(\)](#)

### **phpDocumentor\_IntermediateParser::\$title**

*string = " [line [291](#)]*

- Var Title of generated documentation, passed to Converters

### **phpDocumentor\_IntermediateParser::\$type**

*string = " [line [97](#)]*

**type of the current parser Element being handled**

This is used by [HandleEvent\(\)](#) to set the [\\$lasttype](#) var, which is used to detect page-level DocBlocks

## **phpDocumentor\_IntermediateParser::\$undocumentedElementWarnings**

*boolean = false [line 261]*

**set in [Setup.inc.php](#) to the value of the undocumentedElementWarnings commandline option.**

If this option is true, warnings about certain elements (classes, methods) that are not documented with DocBlocks will be shown while parsing, and will also be displayed in the errors.html page (other documentation is unaffected)

- **Tutorial [-ue, --undocumentedelements](#)**

## **phpDocumentor\_IntermediateParser::\$uses**

*mixed = array() [line 293]*

Constructor *void* function phpDocumentor\_IntermediateParser::phpDocumentor\_IntermediateParser([*\$title* = 'Generated Documentation']) [line 326]

**Function Parameters:**

- *string \$title* Title of generated documentation, passed to Converters

## **sets up basic data structures**

- See [phpDocumentor\\_IntermediateParser::\\$title](#), [phpDocumentor\\_IntermediateParser::\\$data](#), [phpDocumentor\\_IntermediateParser::\\$classes](#), [phpDocumentor\\_IntermediateParser::\\$proceduralpages](#)

*void* function phpDocumentor\_IntermediateParser::addConverter(*\$output*, *\$name*, *\$template*) [line 1708]

**Function Parameters:**

- *string \$output* output format (HTML, PDF, XML). Must be all caps
- *string \$name* Converter name (frames, for example, is the name of HTMLframesConverter)
- *string \$template* template to use, should be a relative path to the templates

dir (like DOM/default)

### Add a converter name to use to the list of converters

Sets up the [\\$converters](#) array.

*void* function [phpDocumentor\\_IntermediateParser::addElementToPage\(\\$element, \\$path\)](#) [line [1474](#)]

**Function Parameters:**

- [parserElement](#) **\$element** this will actually be a descendant of parserElement
- [string](#) **\$path**

**adds a processed descendant of parserElement to the \$pages array or \$privatepages array**

This function expects the page to exist in either \$pages or \$privatepages. It calls the [parserData::addElement\(\)](#) method to add \$element to the page.

*void* function [phpDocumentor\\_IntermediateParser::addPackageParent\(&\\$class\)](#) [line [1685](#)]

**Function Parameters:**

- [parserClass](#) **&\$class**

**If the parent class of \$class is in a different package, adds it to the \$package\_parents array**

*void* function [phpDocumentor\\_IntermediateParser::addPage\(\\$page, \\$path\)](#) [line [1368](#)]

**Function Parameters:**

- [parserPage](#) **\$page**
- [string](#) **\$path** full path to the file

**Replaces the parserPage represented by \$this->pages[\$path] with \$page**

Called by [addPageIfNecessary\(\)](#), [handleDocBlock\(\)](#) and [ProceduralPages::setupPages\(\)](#), this method first checks to see if the page has been added. If not, it assumes that the page has either been @ignored or set with @access private with --parseprivate off, and returns [addPrivatePage\(\)](#). Otherwise, it sets the

`pages[$path]` to be the `parserPage` `$page` and sets the package and subpackage to that of `$page`

- See [phpDocumentor\\_IntermediateParser::\\$pages](#)

*void* function `phpDocumentor_IntermediateParser::addPageIfNecessary($path, &$class)` [[line 1398](#)]

**Function Parameters:**

- `string $path` full path of page
- `&$class`

**add a new [parserPage](#) to the `$pages` array if none is found**

This method is used when a page has been `@ignored` or marked with `@access private`, and a public class is in the page (a class with no `@access private` in its DocBlock). The method first creates a new page in the `$pages` array and then copies path information, and calls [addPage\(\)](#) to set up packages

*void* function `phpDocumentor_IntermediateParser::addPrivatePage($page, $path)` [[line 1436](#)]

**Function Parameters:**

- [parserPage](#) `$page`
- `string $path` full path to the page

**Adds a [parserPage](#) element to the `parserData` element in `$this->privatePages[$path]`**

Performs a similar function to `addPage`, but adds to the `$privatePages` array

- See [phpDocumentor\\_IntermediateParser::addPage\(\)](#)

*void* function `phpDocumentor_IntermediateParser::addUses($element, $path)` [[line 1505](#)]

**Function Parameters:**

- *parserElement* **\$element** descendant of parserElement
- *string* **\$path** full path to the file

**Add all the @uses tags from \$element to the \$uses array so that @usedby virtual tags can be added**

- **Uses** `parserUsesTag::getSeeElement()` - used to initialize **\$uses**
- **Uses** `parserUsesTag::getDescription()` - used to initialize **\$uses**

*int* function `phpDocumentor_IntermediateParser::ClasselementCmp($a, $b)` [line [1775](#)]

**Function Parameters:**

- *mixed* **\$a**
- *mixed* **\$b**

**does a natural case sort on two class elements (either [parserClass](#), [parserMethod](#) or [parserVar](#)**

- **See** `generateElementIndex()`

*void* function `phpDocumentor_IntermediateParser::Convert($title, $converter)` [line [1650](#)]

**Function Parameters:**

- **\$title**
- **\$converter**

## Interface to the Converter

This function simply passes [\\$pages](#) and [package\\_pages](#) to the walk() method, and

then calls the Output() method. Note that Output() is not required to do anything, and in fact doesn't in HTMLframesConverter.

- **Uses** [Converter::walk\(\)](#) - passes \$pages and \$package\_pages
- **Uses** [Converter::Output\(\)](#)

*int* function phpDocumentor\_IntermediateParser::elementCmp(\$a, \$b) [line [1761](#)]

**Function Parameters:**

- *mixed* **\$a**
- *mixed* **\$b**

**does a natural case sort on two [parserElement](#) descendants**

- **See** generateElementIndex()

*void* function phpDocumentor\_IntermediateParser::handleClass(\$event, \$data) [line [905](#)]

**Function Parameters:**

- *integer* **\$event** Event number from [Parser.inc](#)
- [parserClass](#) **\$data**

**handles post-parsing of classes**

This function sets **\$data->clean** to false to tell the phpDocumentor\_IntermediateParser that a page-level DocBlock can't be found after this point on this page. It sets **\$cur\_class** to its name, and if an @ignore tag is found in the DocBlock, it sets **\$private\_class** to true, to prevent post-parsing of any of the class's vars or methods. Then it checks for the existence of a package page for the class's package

*void* function phpDocumentor\_IntermediateParser::handleConst(\$event, \$data) [line [609](#)]

**Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- [\*parserVar \\$data\*](#)

### handles post-parsing of class constants

This function aligns \$data's \$path var and packages to match the parent object

*void* function `phpDocumentor_IntermediateParser::handleDefine($event, $data)` [[line 849](#)]

#### **Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- [\*parserDefine \\$data\*](#)

### handles post-parsing of defines

This function sets [\*\\$data->clean\*](#) to false to tell the `phpDocumentor_IntermediateParser` that a page-level DocBlock can't be found after this point on this page. It then sets the package to be the same as the page and adds itself to the [ProceduralPages](#) class

*void* function `phpDocumentor_IntermediateParser::handleDocBlock($event, $data)` [[line 1046](#)]

#### **Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- [\*parserDocBlock \\$data\*](#)

### handles post-parsing of DocBlocks

This function sets [\*\\$last\*](#) to the DocBlock represented by \$data, to allow the next documentable element passed to `phpDocumentor_IntermediateParser` to link the DocBlock into its `$docblock` property. This function also checks for two special cases of DocBlocks:

1. First DocBlock in the file contains a @package tag
2. First DocBlock in the file is immediately followed by another DocBlock

In both cases, the function extracts this tag and uses it as the page-level package. If the @package tag is in the DocBlock of an element (function, global variable, whatever) that isn't a page-level DocBlock, a warning will be raised to notify the author that a @package tag belongs in a page-level DocBlock.

**New** in version 1.2.2, if the first DocBlock in a file contains a @package tag, it is a page-level DocBlock.

If the DocBlock is page-level, it is processed with `_processPageLevelDocBlock`

Finally, the function replaces the old `parserPage` in `parserData::$data->parent` with the new one containing information from the DocBlock by calling [addPage\(\)](#), and checks for package-level docs.

`void function phpDocumentor_IntermediateParser::HandleEvent($event, $data) [line 1226]`

**Function Parameters:**

- *integer* `$event` event number from [Parser.inc](#)
- *mixed* `$data` if `$event` is [PHPDOCUMENTOR\\_EVENT\\_NEWSTATE](#), `$data` is a [PHP\\_DOC\\_EVENT\\_END\\_PAGE](#) or [STATE\\_END\\_CLASS](#), otherwise `$data` is either a [parserDocBlock](#), [parserPage](#) or descendant of [parserElement](#)

**called via [Parser::parse\(\)](#) and Parser's inherited method [Publisher::publishEvent\(\)](#)**

`$event` is one of the PHPDOC constants from `Parser.inc`. If it is not `PHPDOCUMENTOR_EVENT_NEWSTATE`, then a function name is retrieved from the [\\$event handlers](#) array and called to handle the `$data`

- **Global Variable Used** array  
`$_phpDocumentor_setting$phpDocumentor_DefaultPackageName`: we use 'sourcecode' to determine whether to highlight the source of the current file if it has no file-level docblock

`void function phpDocumentor_IntermediateParser::handleFunction($event, $data) [line 763]`

**Function Parameters:**

- *integer* `$event` Event number from [Parser.inc](#)
- *parserFunction* `$data`

**handles post-parsing of functions**

This function sets `$data->clean` to false to tell the `phpDocumentor_IntermediateParser` that a page-level DocBlock can't be found after this

point on this page. It then sets the package to be the same as the page, aligns the docblock's @param, @global, and @staticvar tags with the information parsed from the function source code.

If source code has been parsed by a {@source} tag, the source is added to its docblock, and then the parserFunction adds itself to the [ProceduralPages](#) class

`void function phpDocumentor_IntermediateParser::handleGlobal($event, $data) [line 459]`

**Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- *parserGlobal \$data*

### handles post-parsing of global variables

This function sets `$data->clean` to false to tell the `phpDocumentor_IntermediateParser` that a page-level DocBlock can't be found after this point on this page. It then sets the package to be the same as the page, and adds itself to the [ProceduralPages](#) class

`void function phpDocumentor_IntermediateParser::handleInclude($event, $data) [line 403]`

**Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- *parserInclude \$data*

### handles post-parsing of include/require/include\_once/require\_once

This function sets `$data->clean` to false to tell the `phpDocumentor_IntermediateParser` that a page-level DocBlock can't be found after this point on this page. It then sets the package to be the same as the page, and adds itself to the [ProceduralPages](#) class

`void function phpDocumentor_IntermediateParser::handleMethod($event, $data) [line 659]`

**Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- *parserMethod \$data*

### **handles post-parsing of class methods**

This function first aligns \$data's path and package to match the parent object, and also aligns the docblock's @param, @global, and @staticvar tags with the information parsed from the method source code. It also checks to see if the method is a constructor and sets the \$isConstructor flag. If source code has been parsed by a {@source} tag, the source is added to its docblock

Finally, it adds the method to the [Classes](#) class.

*void* function phpDocumentor\_IntermediateParser::handlePackagePage(*\$event*, *\$data*) [[line 515](#)]

#### **Function Parameters:**

- *integer* **\$event** Event number from [Parser.inc](#)
- [\*parserPackagePage\*](#) **\$data**

### **handles post-parsing of Package-level documentation pages.**

sets the [\\$package pages](#)[\$data->package] to \$data

*void* function phpDocumentor\_IntermediateParser::handlePage(*\$event*, *\$data*) [[line 1000](#)]

#### **Function Parameters:**

- *integer* **\$event** Event number from [Parser.inc](#)
- [\*parserPage\*](#) **\$data**

### **handles post-parsing of procedural pages**

this event is called at the start of a new page, before the Parser knows whether the page will contain any procedural pages or not

*void* function phpDocumentor\_IntermediateParser::handleTutorial(*\$event*, *\$data*) [[line 530](#)]

#### **Function Parameters:**

- *integer* **\$event** Event Number
- [\*parserTutorial\*](#) **\$data**

### **handle post-parsing of Tutorials.**

This adds the parsed tutorial to the tutorial tree

- Since 1.2
- Uses \$tutorials - sets the value of tutorials to parameter \$data

void function phpDocumentor\_IntermediateParser::handleVar(\$event, \$data) [[line 550](#)]

**Function Parameters:**

- *integer \$event* Event number from [Parser.inc](#)
- [\*parserVar\*](#) \$data

**handles post-parsing of class vars**

This function sets up a @var tag if none is found, and aligns \$data's \$path var and packages to match the parent object

void function phpDocumentor\_IntermediateParser::Output([*\$title* = "Generated Documentation"]) [[line 1805](#)]

**Function Parameters:**

- *\$title*

**call this method once parsing has completed.**

This method calls the private methods fixClasses and fixProcPages, both of which adjust inheritance and package information based on complicated post-parsing rules described in [ProceduralPages::setupPages\(\)](#) and [Classes::Inherit\(\)](#). Then, it sorts elements of the \$pages array and calls Convert for each Converter in the \$converters array

- See [phpDocumentor\\_IntermediateParser::Convert\(\)](#)
- See [phpDocumentor\\_IntermediateParser::\\$pages](#)
- See [phpDocumentor\\_IntermediateParser::\\$converters](#)

*void* function `phpDocumentor_IntermediateParser::parsePackagePage($package, $path)` [line [1180](#)]

**Function Parameters:**

- *string* **\$package** package name of package file to parse
- *string* **\$path** directory of file that contains package name

**Backward-compatibility only, use the new tutorials for more power**

- **Tutorial** [phpDocumentor Tutorials](#)

*void* function `phpDocumentor_IntermediateParser::setParsePrivate($parse)` [line [1929](#)]

**Function Parameters:**

- *bool* **\$parse**

**set display of elements marked with @access private**

If set to true, elements will be displayed

*void* function `phpDocumentor_IntermediateParser::setQuietMode($quietMode)` [line [1906](#)]

**Function Parameters:**

- *bool* **\$quietMode**

**set parsing information output mode (quiet or verbose)**

If set to false, no parsing information (parsing /php/file/thisfile.php, Converting etc.) will be displayed. Useful for cron jobs

*void* function `phpDocumentor_IntermediateParser::setTargetDir($dir)` [line [1882](#)]

**Function Parameters:**

- *string* **\$dir** the output directory

## Sets the output directory

void function phpDocumentor\_IntermediateParser::setTemplateBase(\$dir) [[line 1893](#)]

### Function Parameters:

- *string \$dir* the template base directory

## Sets the template base directory

- [Tutorial -tb, --templatebase](#)

void function  
phpDocumentor\_IntermediateParser::setUndocumentedElementWarningsMode(\$undocumentedElementWarnings) [[line 1918](#)]

### Function Parameters:

- *bool \$undocumentedElementWarnings*

## show warnings for undocumented elements

If set to false, no warnings will be shown for undocumented elements. Useful for identifying classes and methods that haven't yet been documented.

void function phpDocumentor\_IntermediateParser::\_\_guessPackage(\$path, \$sourceloc) [[line 376](#)]

### Function Parameters:

- *string \$path* full path of file
- *template-ready \$sourceloc* source location Program\_Root/dir/file.php

## Guess the package/subpackage based on subdirectory if the --pear option

A file in pear/dir/file.php will be in package "dir." A file in pear/dir/subdir/file.php will be in package "dir," subpackage "subdir."

- **Global Variable Used** array [\\$\\_phpDocumentor\\_setting](#): uses the 'pear' option to determine whether to guess based on subdirectory
- **Tutorial** [-p, --pear](#)

## Class phpDocumentor\_setup *[line 100]*

- **Package** phpDocumentor

**phpDocumentor\_setup::\$dirs**

*string = " [line 133]"*

**contents of --directory commandline**

- **Tutorial** [-d, --directory](#)

**phpDocumentor\_setup::\$files**

*string = " [line 127]"*

**contents of --filename commandline**

- **Tutorial** [-f, --filename](#)

**phpDocumentor\_setup::\$hidden**

*boolean = false [line 139]*

**contents of --hidden commandline**

- Tutorial [-dh, --hidden](#)

**phpDocumentor\_setup::\$ignoresymlinks**

*boolean = false [line 155]*

**contents of --ignoresymlinks commandline**

**phpDocumentor\_setup::\$ignore\_files**

*string = array() [line 150]*

**contents of --ignore commandline**

- Tutorial [-i, --ignore](#)

**phpDocumentor\_setup::\$packages**

*string = false [line 121]*

**Packages to create documentation for**

**phpDocumentor\_setup::\$parse**

*Parser|phpDocumentorTParser = [line 106]*

**The main parser**

**phpDocumentor\_setup::\$render**

*phpDocumentor\_IntermediateParser = false [line 116]*

## Used to organize output from the Parser before Conversion

`phpDocumentor_setup::$setup`

*Io = [line 111]*

## Used to parse command-line options

Constructor `void` function `phpDocumentor_setup::phpDocumentor_setup()` *[line 162]*

**Checks PHP version, makes sure it is 4.2.0+, and chooses the  
phpDocumentorTParser if version is 4.3.0+**

- [Uses `phpDocumentor\_setup::parseln\(\)`](#)

`void` function `phpDocumentor_setup::checkIgnoreTag($tagname, [$inline = false])` *[line 404]*

**Function Parameters:**

- `$tagname`
- `$inline`

`void` function `phpDocumentor_setup::createDocs([$title = false])` *[line 537]*

**Function Parameters:**

- `$title`

`void` function `phpDocumentor_setup::parseHiddenFiles([$flag = true])` *[line 520]*

**Function Parameters:**

- `$flag`

`void` function `phpDocumentor_setup::parseln()` *[line 778]*

**Parse configuration file `phpDocumentor.ini`**

- Used by [phpDocumentor\\_setup::phpDocumentor\\_setup\(\)](#)

*void* function phpDocumentor\_setup::readCommandLineSettings() [line 267]

### Get phpDocumentor settings from command-line or web interface

*void* function phpDocumentor\_setup::readConfigFile(\$file) [line 233]

#### Function Parameters:

- **\$file** user configuration file

### Get phpDocumentor settings from a user configuration file

*void* function phpDocumentor\_setup::setDirectoriesToParse(\$dirs) [line 515]

#### Function Parameters:

- **\$dirs**

*void* function phpDocumentor\_setup::setFilesToParse(\$files) [line 510]

#### Function Parameters:

- **\$files**

*void* function phpDocumentor\_setup::setIgnore(\$ig) [line 525]

#### Function Parameters:

- **\$ig**

*void* function phpDocumentor\_setup::setJavadocDesc() [line 469]

*void* function phpDocumentor\_setup::setMemoryLimit() [line 417]

### Allow a memory\_limit setting in phpDocumentor.ini to override php.ini or default memory limit

- **TODO** recognize "K" and "G" in memory\_limit settings, rather than just "M"

*void function phpDocumentor\_setup::setPackageOutput(\$po) [line 499]*

**Function Parameters:**

- **\$po**

*void function phpDocumentor\_setup::setParsePrivate([\$flag = true]) [line 474]*

**Function Parameters:**

- **\$flag**

*void function phpDocumentor\_setup::setQuietMode([\$flag = true]) [line 479]*

**Function Parameters:**

- **\$flag**

*void function phpDocumentor\_setup::setTargetDir(\$target) [line 489]*

**Function Parameters:**

- **\$target**

*void function phpDocumentor\_setup::setTemplateBase(\$dir) [line 494]*

**Function Parameters:**

- **\$dir**

*void function phpDocumentor\_setup::setTitle(\$ti) [line 505]*

**Function Parameters:**

- **\$ti**

*void function phpDocumentor\_setup::setUndocumentedElementWarnings([\$flag = true]) [line 484]*

**Function Parameters:**

- **\$flag**

## Class ProceduralPages

*[line 58]*

### Intermediate procedural page parsing structure.

This structure parses defines, functions, and global variables by file, and then iterates over the elements to document conflicts.

- **Package** phpDocumentor
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.1
- **License** [LGPL](#)

### ProceduralPages::\$curfile

*string = [line 67]*

**file being parsed, used in every add function to match up elements with the file that contains them**

- **See** addClass(), addMethod(), addVar(), nextFile()

### ProceduralPages::\$defineconflicts

*array = array() [line 314]*

## **Namespace conflicts within all documented packages of functions**

Format: array(  
    functionname => array(  
        full path,  
        full path,  
        ...  
    )  
)

### **ProceduralPages::\$definesbyfile**

array = array() [/line 157]

**array of parsed defines organized by the full path of the file that contains the define.**

Format: array(  
    full path => array(  
        definename => [parserDefine](#)  
    )  
)

### **ProceduralPages::\$definesbynamefile**

array = array() [/line 211]

**array of file names organized by defines that are in the file.**

This structure is designed to handle name conflicts. Two files can contain defines with the same name, and this array will record both filenames to help control namespace errors

Format: array(  
    definename => array(  
        full path of file containing definename,  
        full path of file 2 containing definename,  
        ...  
    )  
)

### **ProceduralPages::\$functionconflicts**

array = array() [/line 282]

## **Namespace conflicts within all documented packages of functions**

Format: array(  
    functionname => array(  
        full path,

```
    full path,  
    ...  
)  
)
```

### ProceduralPages::\$functionsbyfile

*array = array() [line 142]*

**array of parsed functions organized by the full path of the file that contains the function.**

Format: array(  
 full path => array(  
 functionname => [parserFunction](#)  
 )  
)

### ProceduralPages::\$functionsbynamefile

*array = array() [line 192]*

**array of file names organized by functions that are in the file.**

This structure is designed to handle name conflicts. Two files can contain functions with the same name, and this array will record both filenames to help control namespace errors

Format: array(  
 functionname => array(  
 full path of file containing functionname,  
 full path of file 2 containing functionname,  
 ...  
 )  
)

### ProceduralPages::\$globalconflicts

*array = array() [line 330]*

**Namespace conflicts within all documented packages of functions**

Format: array(  
 functionname => array(  
 full path,  
 full path,  
 ...  
 )  
)

### **ProceduralPages::\$globalsbyfile**

*array = array() [line 172]*

**array of parsed global variables organized by the full path of the file that contains the global variable definition.**

Format: array(  
    full path => array(  
        globalname => [parserGlobal](#)  
    )  
)

### **ProceduralPages::\$globalsbynamefile**

*array = array() [line 231]*

**array of file names organized by global variables that are in the file.**

This structure is designed to handle name conflicts. Two files can contain global variables with the same name, and this array will record both filenames to help control namespace errors

Format: array(  
    global variablename => array(  
        full path of file containing global variablename,  
        full path of file 2 containing global variablename,  
        ...  
    )  
)

### **ProceduralPages::\$ignorepages**

*array = array() [line 100]*

**array of all procedural pages ordered by name**

that have been ignored via -po or @access private or @ignore Format: array(  
    name => array(  
        fullpath => parserPage,  
        fullpath => parserPage2 [if there are name conflicts],  
        ...  
    )  
)

### **ProceduralPages::\$includesbyfile**

*array = array() [line 127]*

**array of parsed includes organized by the full path of the file that contains the include.**

Format: array(  
full path => array(  
includename => [parserInclude](#)  
)  
)

### **ProceduralPages::\$pageclasspackages**

*array = array() [line 266]*

**array of packages assigned to classes in a file, ordered by fullname**

Format: array(  
fullname => array(  
packagename => array(  
subpackagename => 1,  
subpackagename => 1,  
"  
,  
packagename2 => array(...  
)  
)  
)

### **ProceduralPages::\$pageconflicts**

*array = array() [line 298]*

**Namespace conflicts within all documented pages**

Format: array(  
pagename => array(  
fullpath,  
fullpath,  
"  
,  
...  
)  
)

### **ProceduralPages::\$pagepackages**

*array = array() [line 246]*

**array of packages ordered by full path**

Format: array(  
fullpath => array(

```
    packagename,  
    subpackagename  
)  
)
```

### **ProceduralPages::\$pages**

*array = array() [line 83]*

#### **array of all procedural pages ordered by name**

Format: array(  
 name => array(  
 fullpath => parserPage,  
 fullpath => parserPage2 [if there are name conflicts],  
 ...  
 )  
)

### **ProceduralPages::\$pathpages**

*array = array() [line 112]*

#### **array of all procedural page names ordered by full path to the file**

Format: array(  
 fullpath => name  
)

*void function ProceduralPages::addClassPackageToFile(\$file, \$package, \$subpackage) [line 613]*

#### **Function Parameters:**

- **string \$file** full path to the file that contains the class
- **string \$package** package name
- **string \$subpackage** subpackage name

### **adds a package from a class to the current file**

*void function ProceduralPages::addDefine(&\$element) [line 552]*

#### **Function Parameters:**

- **parserDefine &\$element** the "define" element

## **sets up the [\\$definesbyfile](#) array using [\\$curfile](#)**

*void function ProceduralPages::addFunction(&\$element) [line 508]*

### ***Function Parameters:***

- [parserFunction](#) &\$element the "function" object

## **sets up the [\\$functionsbyfile](#) array using [\\$curfile](#)**

*void function ProceduralPages::addGlobal(&\$element) [line 530]*

### ***Function Parameters:***

- [parserGlobal](#) &\$element the "global" element

## **sets up the [\\$globalsbyfile](#) array using [\\$curfile](#)**

*void function ProceduralPages::addInclude(&\$element) [line 496]*

### ***Function Parameters:***

- [parserInclude](#) &\$element the "include" element object

## **sets up the [\\$includesbyfile](#) array using [\\$curfile](#)**

*void function ProceduralPages::addPage(&\$element) [line 349]*

### ***Function Parameters:***

- [parserPage](#) &\$element the parser page element

## **sets up the [\\$pages](#) array**

*void function ProceduralPages::addPagePackage(\$path, \$package, \$subpackage) [line 450]*

### ***Function Parameters:***

- *string \$path* full path
- *string \$package* the package name
- *string \$subpackage* the subpackage name

### Changes the package of the page represented by \$path

changes package in both the [\\$pages](#) array and the [pagepackages](#) array

*array/bool* function ProceduralPages::getPathInfo(\$path, &\$c) [line 385]

#### Function Parameters:

- *string \$path* path to the element
- *mixed &\$c ???*

### gathers path-related info about a given element

- **TODO** figure out what &\$c is and update the param tag

*array/string* function ProceduralPages::getRealPath(\$path, \$file) [line 1033]

#### Function Parameters:

- *string \$path* path to the file
- *string \$file* the file name

### Ensures the path to the file is an absolute path

*void* function ProceduralPages::ignorePage(&\$element) [line 368]

#### Function Parameters:

- [parserPage](#) &\$element the parser page element

### moves a page from the [\\$pages](#) array to the [\\$ignorepages](#) array

*parserPage*|*bool* function ProceduralPages::pathMatchesParsedFile(*\$path*, *\$infile*) [line 983]

**Function Parameters:**

- *string \$path* the path to look for
- *string \$infile* the file to check

**checks to see if the parsed file matches the given path**

*void* function ProceduralPages::replaceElement(&*\$element*) [line 575]

**Function Parameters:**

- *parserElement &\$element* the element to align

**Used to align an element with the package of its parent page prior to Conversion.**

*void* function ProceduralPages::setName(*\$name*) [line 421]

**Function Parameters:**

- *string \$name* the alias

**Change a page's name from its file to alias \$name**

This function is used to handle a @name tag in a page-level DocBlock

*void* function ProceduralPages::setParseBase(*\$pbase*) [line 969]

**Function Parameters:**

- *mixed \$pbase* the parser base

**sets the parser base**

*void* function ProceduralPages::setupPagePackages() [line 628]

**if there is one class package in a file, the parent path inherits the package if its package is default.**

helps with -po to avoid dumb bugs

*void* function ProceduralPages::setupPages(&\$render) [line 911]

**Function Parameters:**

- [phpDocumentor\\_IntermediateParser](#) &\$render the parser

### Adjusts packages of all pages and removes name conflicts within a package

Automatic linking requires that each linkable name have exactly one element associated with it. In other words, there cannot be two functions named foo() in the same package.

This also adheres to php rules with one exception:

```
1  if ($test == 3) {
2      define('whatever', 'this thing');
3  } else {
4      define('whatever', 'this other thing');
5  }
```

phpDocumentor is not aware of conditional control structures because it would slow things down considerably. So, what phpDocumentor does is automatically ignore the second define and raise a warning. The warning can be eliminated with an @ignore tag on the second element like so:

```
1  if ($test == 3) {
2      define('whatever', 'this thing');
3  } else {
4      /**
5      * @ignore
6      */
7      define('whatever', 'this other thing');
8  }
```

If there are two files that contain the same procedural elements in the same package (for example, a common configuration file common.php), they will also be ignored as if they were in the same file. The reasoning behind this is simple. A package is an indivisible set of files and classes that a user will include in their code. Name conflicts must be avoided to allow successful execution.

This function also plays the all-important role of calling [phpDocumentor\\_IntermediateParser::addElementToPage\(\)](#) in order to add processed elements to their pages for Conversion.

## Class Publisher

[line 54]

a class for handling the publishing of data

- **Package** phpDocumentor
- **Author** Kellin < [passionplay@hotmail.com](mailto:passionplay@hotmail.com)>
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** Release: @VER@
- **Copyright** 2000-2007 Kellin, Joshua Eichorn
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

### Publisher::\$popEvent

*array = array() [line 67]*

### Publisher::\$pushEvent

*array = array() [line 66]*

### Publisher::\$subscriber

*array = array() [line 62]*

## Array of references objects that have Subscribed to this publisher

### Publisher::\$tokens

*array = array() [line 64]*

*void function Publisher::publishEvent(\$event, \$data) [line 96]*

#### **Function Parameters:**

- *integer \$event* see [Parser.inc](#) PARSER\_EVENT\_\* constants
- *mixed \$data* anything the subscribed event handler is expecting

## Publish an event

*void function Publisher::subscribe(\$event, &\$object) [line 83]*

#### **Function Parameters:**

- *integer \$event* see [Parser.inc](#) PARSER\_EVENT\_\* constants
- *class &\$object* any class that has a HandleEvent() method like [phpDocumentor\\_IntermediateParser::HandleEvent\(\)](#) or

Classes::HandleEvent()

**Adds a subscriber to the [\\$subscriberarray\(\)](#).**

if \$event is '\*', the publisher will use \$object as the default event handler

- **TODO** CS Cleanup - there's no way I can get the &\$object desc under 85 chars

## Class tests\_HighlightParserTests [line 12]

- **Package** phpDocumentor

*void* function tests\_HighlightParserTests::main() [line 14]

- **Static**
- **Access** public

*void* function tests\_HighlightParserTests::suite() [line 19]

- **Static**
- **Access** public

## Class tests\_IntermediateParserTests

[line 12]

- **Package** phpDocumentor

*void* function tests\_IntermediateParserTests::main() [line 14]

- **Static**
- **Access** public

*void* function tests\_IntermediateParserTests::suite() [line 19]

- **Static**
- **Access** public

## Class tests\_ParserClassTests

[line 12]

- **Package** phpDocumentor

*void* function tests\_ParserClassTests::main() [line 14]

- **Static**
- **Access** public

*void* function tests\_ParserClassTests::suite() [*line 19*]

- **Static**
- **Access** public

## Class tests\_ParserPageTests [*line 12*]

- **Package** phpDocumentor

*void* function tests\_ParserPageTests::main() [*line 14*]

- **Static**
- **Access** public

*void* function tests\_ParserPageTests::suite() [*line 19*]

- **Static**
- **Access** public

## Class tests\_phpDocumentorSetupTests [*line 13*]

- **Package** phpDocumentor

*void* function tests\_phpDocumentorSetupTests::main() [*line 15*]

- **Static**
- **Access** public

*void* function tests\_phpDocumentorSetupTests::suite() [*line 20*]

- **Static**
- **Access** public

## Class tests\_phpDocumentorTParserTests [*line 12*]

- **Package** phpDocumentor

*void* function tests\_phpDocumentorTParserTests::main() [*line 14*]

- **Static**
- **Access** public

*void* function tests\_phpDocumentorTParserTests::suite() [*line 19*]

- **Static**
- **Access** public



# DescHTML.inc

All abstract representations of html tags in DocBlocks are handled by the classes in this file

Before version 1.2, phpDocumentor simply passed html to converters, without much thought, except the [adv\\_htmlentities\(\)](#) function was provided along with a list of allowed html. That list is no longer used, in favor of these classes.

The PDF Converter output looked wretched in version 1.1.0 because line breaks in DocBlocks were honored. This meant that output often had just a few words on every other line! To fix this problem, DocBlock descriptions are now parsed using the ParserDescParser, and split into paragraphs. In addition, html in DocBlocks are parsed into these objects to allow for easy conversion in destination converters. This design also allows different conversion for different templates within a converter, which separates design from logic almost 100%

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2007 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: DescHTML.inc 246329 2007-11-17 03:07:00Z ashnazg \$
- **Copyright** 2002-2007 Gregory Beaver
- **See** [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- **See** [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- **Link** <http://pear.php.net/PhpDocumentor>

- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

## Class parserB [line [161](#)]

**Used for <b> in a description**

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserB
- **Since** 1.2
- **License** [LGPL](#)

*string* function parserB::Convert(&\$c) [line [172](#)]

**Function Parameters:**

- ***Converter* &\$c** the converter object

**performs the conversion of bold tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::Bolden\(\)](#)

## Class parserBr [line 326]

Used for <br> in a description

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver <[cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserBr
- **Since** 1.2
- **License** [LGPL](#)

string function parserBr::Convert(&\$c) [line 337]

**Function Parameters:**

- **Converter** &\$c the converter object

**performs the conversion of linebreak tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::Br\(\)](#)

## Class parserCode

[line 74]

Used for <code> in a description

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserCode
- **Since** 1.2
- **License** [GPL](#)

string function parserCode::Convert(&\$c) [line 85]

**Function Parameters:**

- [Converter](#) &\$c the converter object

performs the conversion of code tags

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::ProgramExample\(\)](#)

## Class parserDescVar

[line 227]

Used for <var> in a description

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserDescVar
- **Since** 1.2
- **License** [LGPL](#)

*string* function parserDescVar::Convert(&\$c) [[line 238](#)]

**Function Parameters:**

- [Converter](#) &\$c the converter object

**performs the conversion of variable tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::Varize\(\)](#)

## Class parser

[[line 194](#)]

**Used for <i> in a description**

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>

- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserI
- **Since** 1.2
- **License** [GPL](#)

*string* function parserI::Convert(&\$c) [[line 205](#)]

**Function Parameters:**

- **Converter** &\$c the converter object

**performs the conversion of italic tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::Italicize\(\)](#)

## Class parserKbd

[[line 293](#)]

**Used for** <kbd> in a description

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserKbd
- **Since** 1.2
- **License** [GPL](#)

*string* function parserKbd::Convert(&\$c) [line [304](#)]

**Function Parameters:**

- [Converter](#) &\$c the converter object

**performs the conversion of keyboard tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::Kbdize\(\)](#)

## Class parserList [line [359](#)]

**Used for lists <ol> and <ul>**

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserList
- **Since** 1.2
- **License** [LGPL](#)

### parserList::\$items

*integer* = 0 [line [368](#)]

### parserList::\$numbered

*boolean = [line [364](#)]*

Constructor *void function parserList::parserList(\$numbered) [line [374](#)]*

**Function Parameters:**

- *integer \$numbered* a reference number for the new list

## Constructor - create a new list

*void function parserList::addItem(\$item) [line [386](#)]*

**Function Parameters:**

- *[parserStringWithInlineTags](#) \$item* the item to add

## add an item to a list

*void function parserList::addList(\$list) [line [398](#)]*

**Function Parameters:**

- *[parserList](#) \$list* the list to add

## add a list

*string function parserList::Convert(&\$c) [line [413](#)]*

**Function Parameters:**

- *[Converter](#) &\$c* the converter object

## performs the conversion of list tags

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::ListItem\(\)](#) - enclose each item of the list
- **Uses** [Converter::EncloseList\(\)](#) - enclose the list

## Class parserPre

[line [127](#)]

**Used for <pre> in a description**

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserPre
- **Since** 1.2
- **License** [GPL](#)

*string* function parserPre::Convert(&\$c) [line [138](#)]

**Function Parameters:**

- [Converter](#) &\$c the converter object

**performs the conversion of code tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::PreserveWhiteSpace\(\)](#)

# Class parserSamp

[line 260]

Used for <samp> in a description

- **Package** phpDocumentor
- **Sub-Package** DescHTML
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - rename class to ParserSamp
- **Since** 1.2
- **License** [LGPL](#)

string function parserSamp::Convert(&\$c) [line 271]

**Function Parameters:**

- [Converter](#) &\$c the converter object

**performs the conversion of sample tags**

- **TODO** CS cleanup - rename method to convert()
- **Uses** [Converter::Sampize\(\)](#)

# DocBlockTags.inc

All abstract representations of DocBlock tags are defined by the classes in this file  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: DocBlockTags.inc 287889 2009-08-30 07:27:39Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** separate file since version 1.2
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

## Class parserAccessTag

[line 253]

This class represents the @access tag

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@access](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

**parserAccessTag::\$isValid**

*boolean = false* [line 266]

**set to true if the returned tag has a value type of private, protected or public, false otherwise**

**parserAccessTag::\$keyword**

*string = 'access'* [line 259]

**tag name**

Constructor *void* function parserAccessTag::parserAccessTag(\$value) [line 276]

**Function Parameters:**

- [parserStringWithInlineTags](#) **\$value** the tag value

**checks \$value to make sure it is private, protected or public, otherwise it's not a valid @access tag**

- See [parserAccessTag::\\$isValid](#)

*string* function parserAccessTag::Convert(&\$converter) [[line 308](#)]

**Function Parameters:**

- [Converter](#) &\$converter the output converter

**process this tag through the given output converter**

- See [parserStringWithInlineTags::Convert\(\)](#)
- TODO CS cleanup - rename to convert for camelCase rule

*string* function parserAccessTag::getString() [[line 318](#)]

**No inline tags are possible, returns 'public', 'protected' or 'private'**

## Class parserExampleTag

[[line 1180](#)]

**represents "@example"**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@example](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### **parserExampleTag::\$keyword**

*string = 'example' [line [1186](#)]*

**always "example"**

Constructor *void function parserExampleTag::parserExampleTag(\$value, \$current\_path) [line [1203](#)]*

#### **Function Parameters:**

- [parserStringWithInlineTags](#) **\$value** tag value
- *string \$current\_path* path of file containing tag this @example

### **Reads and parses the example file indicated**

The example tag takes one parameter: the full path to a php file that should be parsed and included as an example.

- **TODO** does this "x = y = z = false" still work as expected in PHP5?
- **TODO** CS cleanup - rename constant to TOKENIZER\_EXT
- **Uses** [phpDocumentorTWordParser::getFileSource\(\)](#) - uses to parse an example and retrieve all tokens by line number

*void function parserExampleTag::ConvertSource(&\$c) [line [1353](#)]*

#### **Function Parameters:**

- [Converter](#) **&\$c** the output converter

### **convert the source code**

- **TODO** CS cleanup - rename to convertSource for camelCase rule
- **TODO** what's up with all the "return" statements? can they \_all\_ be removed?
- **Uses** [parserFileSourceTag::writeSource\(\)](#)
- **Uses** [phpDocumentor\\_HighlightParser](#) - highlights source code

*string* function parserExampleTag::getSourceLink(&\$c) [[line 1389](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter

## Retrieve a converter-specific link to the example

- **Uses** Converter::getExampleLink() - retrieve the link to the example

## Class parserFileSourceTag

[[line 1061](#)]

**represents** "@filesource"

Use this to create a link to a highlighted phpxref-style source file listing

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@filesource](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.2
- **License** [LGPL](#)

## parserFileSourceTag::\$keyword

*string* = 'filesource' [[line 1067](#)]

## Always "filesource"

**parserFileSourceTag::\$path**

*string = [line [1075](#)]*

**parserFileSourceTag::\$source**

*array = [line [1071](#)]*

Constructor **void** function **parserFileSourceTag::parserFileSourceTag(\$filepath, \$value)** [[line 1090](#)]

**Function Parameters:**

- **string \$filepath** the file's path
- **array \$value** output from

[phpDocumentorTWordParser::getFileSource\(\)](#)

## Set **\$source**to **\$value**, and set up path

*string* function **parserFileSourceTag::Convert(&\$c)** [[line 1106](#)]

**Function Parameters:**

- [\*\*Converter\*\* &\\$c](#) the output converter

## Return a link to the highlighted source and generate the source

- **TODO** CS cleanup - rename to convert for camelCase rule
- [\*\*Uses parserFileSourceTag::ConvertSource\(\)\*\*](#) - generate source code and write it out

**void** function **parserFileSourceTag::ConvertSource(&\$c)** [[line 1124](#)]

**Function Parameters:**

- [\*\*Converter\*\* &\\$c](#) the output converter

## convert the source code

- **TODO** CS cleanup - rename to convertSource for camelCase rule
- **TODO** what's up with all the "return" statements? can they all be removed?
- **Usedby** [parserFileSourceTag::Convert\(\)](#) - generate source code and write it out
- **Uses** [parserFileSourceTag::writeSource\(\)](#)
- **Uses** [phpDocumentor\\_HighlightParser](#) - highlights source code

*output* function parserFileSourceTag::getSourceLink(&\$c) [[line 1158](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter

**gets path to the sourcecode file**

- **Uses** [Converter::getSourceLink\(\)](#)

*void* function parserFileSourceTag::writeSource(&\$c, \$source) [[line 1145](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter
- **string** \$source highlighted source code

**have the output converter write the source code**

- **Usedby** [parserExampleTag::ConvertSource\(\)](#)
- **Usedby** [parserFileSourceTag::ConvertSource\(\)](#)
- **Uses** [Converter::writeSource\(\)](#) - export highlighted file source

## Class parserLicenseTag

[line [789](#)]

**represents the "@see" tag**

Link to a license, instead of including lines and lines of license information in every file

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@license](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [GPL](#)

**parserLicenseTag::\$keyword**

*string* = 'license' [line [795](#)]

**always 'license'**

Constructor *void* function parserLicenseTag::parserLicenseTag(\$name, \$link) [line [803](#)]

**Function Parameters:**

- *string* **\$name** unused?
- *string* **\$link** URL to link to

**set up the license tag**

## Class parserLinkTag

[line [657](#)]

**represents the "@link" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@link](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1
- **License** [LGPL](#)

**parserLinkTag::\$keyword**

*string = 'link' [line [663](#)]*

**always 'link'**

Constructor *void* function **parserLinkTag::parserLinkTag(\$link)** [*line [672](#)*]

**Function Parameters:**

- *string \$link* URL to link to (might also contain the URL's description text)

**sets up the link tag**

## Class **parserMethodTag** [*line [545](#)*]

**represents the "@method" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@method](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.4.0a1
- **License** [LGPL](#)

#### parserMethodTag::\$keyword

*string = 'method' [line [551](#)]*

**always 'method'**

#### parserMethodTag::\$returnType

*string = 'void' [line [556](#)]*

**the return type a method has**

## Class parserNameTag [line [203](#)]

**This class represents the @name tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@name](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

**parserNameTag::\$keyword**

*string* = 'name' [line [209](#)]

**tag name**

Constructor void function parserNameTag::parserNameTag(\$name, \$value) [line [217](#)]

**Function Parameters:**

- *string* **\$name** tag name (not used)
- *string* **\$value** tag value

**set up the name tag**

*string* function parserNameTag::Convert(&\$c) [line [231](#)]

**Function Parameters:**

- [Converter](#) &\$c output converter

**process this tag through the given output converter**

- See [parserStringWithInlineTags::Convert\(\)](#)
- TODO CS cleanup - rename to convert for camelCase rule

## Class parserParamTag [line [606](#)]

**represents the "@param" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@param](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### parserParamTag::\$keyword

*string = 'param' [line [612](#)]*

**always 'param'**

## Class parserPropertyReadTag

*[line [493](#)]*

**represents the "@property-read" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@property](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.4.0a1
- **License** [LGPL](#)

### parserPropertyReadTag::\$keyword

*string = 'property-read' [line [499](#)]*

**always 'property-read'**

## Class parserPropertyTag [line [450](#)] represents the "@property" tag

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@property](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.4.0a1
- **License** [LGPL](#)

**parserPropertyTag::\$keyword**

*string = 'property' [line [456](#)]*

**always 'property'**

**parserPropertyTag::\$returnType**

*string = 'mixed' [line [461](#)]*

**the type a property has**

Constructor **void** function **parserPropertyTag::parserPropertyTag(\$returnType, \$value)** [line [469](#)]

**Function Parameters:**

- *string \$returnType* the tag value's datatype
- *parserStringWithInlineTags \$value* the tag value

**set up the property tag**

## Class parserPropertyWriteTag

[line [519](#)]

**represents the "@property-write" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@property](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.4.0a1
- **License** [LGPL](#)

**parserPropertyWriteTag::\$keyword**

*string = 'property-write' [line [525](#)]*

**always 'property-write'**

## Class parserReturnTag

[line [341](#)]

**represents the "@return" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@return](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1
- **License** [LGPL](#)

### parserReturnTag::\$converted\_returnType

*mixed = false* [line [374](#)]

**contains a link to the documentation for a class passed as a type in @return, @var or @param**

Example:

```

1   class myclass
2   {
3   ...
4   }
5

```

In this case, \$converted\_returnType will contain a link to myclass instead of the string 'myclass'

- **Var** either the same as \$returnType or a link to the docs for a class
- See [parserReturnTag::\\$returnType](#)

### parserReturnTag::\$keyword

*string = 'return'* [line [347](#)]

**always 'return'**

### parserReturnTag::\$returnType

*mixed = 'void'* [line [351](#)]

**the type a function returns**

Constructor `void` function `parserReturnTag::parserReturnTag($returnType, $value)` [line [382](#)]

**Function Parameters:**

- `string` **\$returnType** returned datatype
- `parserStringWithInlineTags` **\$value** tag value

**set up the tag**

`string` function `parserReturnTag::Convert(&$converter)` [line [398](#)]

**Function Parameters:**

- `Converter` **&\$converter** the output converter

`process this tag through the given output converter (sets up the $converted_returnType)`

- See [parserStringWithInlineTags::Convert\(\)](#), [parserReturnTag::\\$converted\\_returnType](#)
- TODO CS cleanup - rename to convert for camelCase rule

## Class parserSeeTag

[line [704](#)]

represents the "@see" tag

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@see](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net) >
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>

- [Link http://www.phpdoc.org](http://www.phpdoc.org)
- TODO CS cleanup - change package to PhpDocumentor
- TODO CS cleanup - change classname to PhpDocumentor\_\*
- Since 1.0rc1
- License [LGPL](#)

### **parserSeeTag::\$keyword**

*string* = 'see' [[line 710](#)]

**always 'see'**

Constructor *void* function parserSeeTag::parserSeeTag(\$name) [[line 717](#)]

**Function Parameters:**

- *string \$name* element to link to

**sets up the see tag**

*string* function parserSeeTag::Convert(&\$converter) [[line 731](#)]

**Function Parameters:**

- [Converter](#) &\$converter the output converter

**process this tag through the given output converter**

- See [parserStringWithInlineTags::Convert\(\)](#)
- TODO CS cleanup - rename to convert for camelCase rule

## **Class parserStaticvarTag**

[[line 631](#)]

**represents the "@staticvar" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@staticvar](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

**parserStaticvarTag::\$keyword**

*string = 'staticvar' [line [637](#)]*

**always 'staticvar'**

## Class parserTag

*[line [62](#)]*

**used to represent standard tags like @access, etc.**

This class is aware of inline tags, and will automatically handle them using inherited functions

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1
- **License** [LGPL](#)

**parserTag::\$keyword**

*string = " [line 74]*

**tag name (see, access, etc.)**

**parserTag::\$type**

*string = '\_tag' [line 69]*

**Type is used by many functions to skip the hassle of**

**if phpDocumentor\_get\_class(\$blah) == 'parserBlah' always '\_tag'**

Constructor void function parserTag::parserTag(\$keyword, \$value, [\$noparse = false]) [line 86]

**Function Parameters:**

- **string \$keyword** tag name
- **[parserStringWithInlineTags](#) \$value** tag value
- **boolean \$noparse** whether to parse the \$value

for html tags

**Set up the tag**

```
1  function parserTag($keyword $value $noparse= false)
2  {
3      $this>> keyword = $keyword
4      if (!$noparse) {
5          $parser= new parserDescParser;
6          $parser>> subscribe('*', $this);
7          $parser>> parse($value>> value, true, 'parserstringwithinlinetags');
8      } else {
9          $this>> value = $value
10     }
11 }
```

*string* function parserTag::Convert(&\$converter) [line 108]

**Function Parameters:**

- **[Converter](#) &\$converter** the converter object

**Perform the output conversion on [parserTag](#) using the[output converter](#)that is passed in**

- See [Converter](#)
- **TODO** CS cleanup - rename to convert for camelCase rule

*string* function `parserTag::getString()` [[line 173](#)]

**Returns the text minus any inline tags**

- See [parserStringWithInlineTags::getString\(\)](#)

*void* function `parserTag::HandleEvent($a, $desc)` [[line 162](#)]

**Function Parameters:**

- *integer* `$a` not used
- *array* `$desc` array of [parserStringWithInlineTags](#) representing paragraphs in the tag description

**Called by the[parserDescParse](#)when processing a description.**

- See [parserTag::parserTag\(\)](#)
- **TODO** CS cleanup - rename to handleEvent for camelCase rule

## Class `parserTutorialTag`

[[line 995](#)]

## **represents "@tutorial"**

This is exactly like @see except that it only links to tutorials

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [phpDocumentor Tutorials](#)
- **Tutorial** [@tutorial](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.2
- **License** [LGPL](#)

## **parserTutorialTag::\$keyword**

*string = 'tutorial' [line [1001](#)]*

**Always "tutorial"**

*string/bool function parserTutorialTag::Convert(&\$converter) [line [1012](#)]*

### **Function Parameters:**

- [Converter](#) **&\$converter** the output converter

**process this tag through the given output converter**

- See [parserStringWithInlineTags::Convert\(\)](#)
- **TODO** CS cleanup - rename to convert for camelCase rule

# Class parserUsedByTag

[line 932]

This is a virtual tag, it is created by @uses to cross-reference the used element  
This is exactly like @uses.

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.2
- **License** [LGPL](#)

## parserUsedByTag::\$keyword

*string* = 'usedby' [line 938]

**Always "usedby"**

Constructor *void* function parserUsedByTag::parserUsedByTag(\$link, \$description) [line 950]

**Function Parameters:**

- **[abstractLink](#) \$link** link of element that uses this element
- ***string* \$description** description of how the element is used

## set up the usedby tag

*string* function parserUsedByTag::Convert(&\$c) [line 964]

**Function Parameters:**

- **[Converter](#) &\$c** the output converter

## process this tag through the given output converter

- **TODO** CS cleanup - rename to convert for camelCase rule

## Class parserUsesTag [line 838]

**represents the "@uses" tag**

This is exactly like @see except that the element used has a @useby link to this element added to its docblock

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@uses](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.2
- **License** [LGPL](#)

### parserUsesTag::\$keyword

*string = 'uses' [line 844]*

**Always "uses"**

Constructor **void** function parserUsesTag::parserUsesTag(\$seeel, \$description) [line 857]  
**Function Parameters:**

- **string \$seeel** element to link to
- **[parserStringWithInlineTags](#) \$description** description of how

the element is used

## set up the uses tag

*string* function parserUsesTag::Convert(&\$c) [line [879](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter

## Return a link to documentation for other element, and description of how it is used

Works exactly like [parent::Convert\(\)](#) except that it also includes a description of how the element used is used.

- **TODO** CS cleanup - rename to convert for camelCase rule

*parserStringWithInlineTags* function parserUsesTag::getDescription() [line [908](#)]

**Get the description of how the element used is being used.**

- **Used by** [phpDocumentor\\_IntermediateParser::addUses\(\)](#) - used to initialize \$uses

## Class parserVarTag [line [576](#)]

**represents the "@var" tag**

- **Package** phpDocumentor
- **Sub-Package** DocBlockTags
- **Tutorial** [@var](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1
- **License** [LGPL](#)

### **parserVarTag::\$keyword**

*string* = 'var' [line [582](#)]

**always 'var'**

### **parserVarTag::\$returnType**

*string* = 'mixed' [line [587](#)]

**the type a var has**

# Errors.inc

## Error handling for phpDocumentor

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2001-2008 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Errors
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: Errors.inc 253641 2008-02-24 02:35:44Z ashnazg \$
- **Copyright** 2001-2008 Gregory Beaver
- See [parserDocBlock](#), [parserInclude](#), [parserPage](#), [parserClass](#)
- See [parserDefine](#), [parserFunction](#), [parserMethod](#), [parserVar](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 0.4
- **License** [LGPL](#)
- **Filesource** [Source Code for this file](#)

\$phpDocumentor\_errors

[ErrorTracker](#) = new [ErrorTracker](#) [line 1097]

- **Name** \$phpDocumentor\_errors

### \$phpDocumentor\_error\_descrip

```

array = array(
    PDERROR_UNTERMINATED_INLINE_TAG =>
        'Inline tag {@%s} in tag %s is unterminated, "%s"'

    ,
    PDERROR_CLASS_EXISTS =>
        'Class %s already exists in package "%s"'

    ,
    PDERROR_CONVERTER_NOT_FOUND =>
        'Converter %s specified by --output command-line option is not a class'

    ,
    PDERROR_NO_CONVERTERS =>
        'No Converters have been specified by --output command-line option'

    ,
    PDERROR_ACCESS_WRONG_PARAM =>
        '@access was passed neither "public" nor "private."
Was passed: "%s"'

    ,
    PDERROR_MULTIPLE_ACCESS_TAGS =>
        'DocBlock has multiple @access tags, illegal.
        'ignoring additional tag "@access %s"'

    ,
    PDERROR_MULTIPLE_RETURN_TAGS =>
        'DocBlock has multiple @return tags, illegal.
        'ignoring additional tag "@return %s %s"'

    ,
    PDERROR_MULTIPLE_VAR_TAGS =>
        'DocBlock has multiple @var tags, illegal.
        'ignoring additional tag "@var %s %s"'

    ,
    PDERROR_MULTIPLE_PACKAGE_TAGS =>
        'DocBlock has multiple @package tags, illegal.
        'ignoring additional tag "@package %s"'

    ,
    PDERROR_MULTIPLE_SUBPACKAGE_TAGS =>
        'DocBlock has multiple @subpackage tags, illegal.
        'ignoring additional tag "@subpackage %s"'

    ,
    PDERROR_ILLEGAL_PACKAGENAME =>
        '@%s tag has illegal %s name "%s"'

    ,
    PDERROR_OVERRIDDEN_PACKAGE_TAGS =>
        '%s %s\'s DocBlock has @package tag, illegal.
        'ignoring tag "@package %s"'

    ,
    PDERROR_OVERRIDDEN_SUBPACKAGE_TAGS =>
        '"%s\'s DocBlock has @subpackage tags, illegal.
        'ignoring tag "@subpackage %s"'

    ,
    PDERROR_CLASS_CONFLICT =>
        'class "%s" has multiple declarations in package %s,
        'in file %s and file %s, documentation will have output errors!'

    ,
)

```

```

PDERROR_MULTIPLE_NAME_TAGS =>;
'DocBlock has multiple @name tags, illegal. '.
'ignoring additional tag "@name %s"'

, PDERROR_PACKAGEOUTPUT_DELETES_PARENT_FILE =>;
'-po (packageoutput) option deletes parent file "%s" containing class' .
'"%s." . "\n"; ' Try using --defaultpackagename (-dn)

%s to '.
'include the parent file in the same package as the class'

, PDERROR_GLOBAL_NOT_FOUND =>;
'global variable %s specified in @global tag was never found'

, PDERROR_MULTIPLE_GLOBAL_TAGS =>;
'@global define tag already used for global variable "%s"; '.
'ignoring @global %s'

, PDERROR_MALFORMED_GLOBAL_TAG =>;
'incorrect @global syntax. '.
'Should be @global vartype $varname or @global vartype description'

, PDERROR_UNKNOWN_COMMANDLINE =>;
'Unknown command-line option "%s" encountered, use phpdoc -h for help'

, PDERROR_NEED_WHITESPACE =>;
'phpDocumentor programmer error - wordparser whitespace set to false '.
'in handleDocBlock, notify developers. You should never see this error'

, PDERROR_SOURCE_TAG_FUNCTION_NOT_FOUND =>;
'{@source} tag used in a docblock that isn't preceding a function'

, PDERROR_UNMATCHED_LIST_TAG =>;
'unmatched ol or ul tag in DocBlock, parsing will be incorrect'

, PDERROR_CANT_NEST_IN_B =>;
'Can\'t nest a code, pre, ul, or ol tag in a b tag in '.
'phpDocumentor DocBlock (%s tag nested)'

, PDERROR_UNMATCHED_TUTORIAL_TAG =>;
'While parsing extended documentation, "%s" tag was matched '.
'with "%s" endtag, missing endtag'."\ntag
contents:"%s"';

, PDERROR_CANT_HAVE_INLINE_IN_TAGNAME =>;
'Can\'t have an inline tag inside a package page XML tag!'

, PDERROR_TUTORIAL_IS_OWN_CHILD =>;
'Tutorial %s lists itself as its own child in %s, illegal'

, PDERROR_TUTORIAL_IS_OWN_GRANDPA =>;
'Tutorial %s's child %s lists %s as its child in %s, illegal'

, PDERROR_PDFFUNCTION_NO_FUNC =>;
'Invalid pdffunction syntax: "&lt;pdffunction: /&gt;"; '.
'should be "&lt;pdffunction:functionname
[arg="%s" value="%s" ...]&gt;"'

, PDERROR_PDF_TEMPVAR_DOESNT_EXIST =>;
'&lt;pdffunction:%s arg=%s /&gt; called '.

```

'but temporary variable "%s" doesn't exist'

, PDERROR\_UNTERMINATED\_ATTRIB =>  
'Tutorial tag %s attribute %s is unterminated, current value "%s"'

, PDERROR\_NO\_CONVERTER\_HANDLER =>  
'Handler for element of type "%s" called, but %s is not a method of %s'

, PDERROR\_INLINETAG\_IN\_SEE =>  
'Inline tags are not allowed in a @see tag'

, PDERROR\_ID\_MUST\_BE\_INLINE =>  
'<%s id=%s> must be <%s id=%s{@id %s}>'

, PDERROR\_INTERNAL\_NOT\_CLOSED =>  
'{@internal was never terminated with }}'

, PDERROR\_CONVERTER\_OVR\_GFCT =>  
'Converter "%s" must override getFormattedClassTrees() but doesn't'

, PDERROR\_TEXT\_OUTSIDE\_LI =>  
'Text cannot be outside of li tag in a DocBlock list, '.  
'parsing will be incorrect'

, PDERROR\_UNCLOSED\_TAG =>  
'Unclosed %s tag in DocBlock, parsing will be incorrect'

, PDERROR\_TAG\_NOT\_HANDLED =>  
'"%s" tag is not available in PHP built without tokenizer support, tag ignored'

, PDERROR\_MALFORMED\_TAG =>  
'"%s" tag was used without any parameters, illegal'

, PDERROR\_MULTIPLE\_CATEGORY\_TAGS =>  
'package has multiple @category tags, ignoring "%s"'

, PDERROR\_TEMPLATEDIR\_DOESNT\_EXIST =>  
'template directory "%s" does not exist'

, PDERROR\_UNTERMINATED\_ENTITY =>  
'entity &%s is unterminated'

, PDERROR\_FUNCTION\_HAS\_NONAME =>  
'function has no name (PHP error - test your file before parsing!)'

, PDERROR\_CANNOT\_EXTEND\_SELF =>  
'class %s cannot extend itself - TEST YOUR CODE BEFORE PARSING'

, PDERROR\_DUMBUSES =>  
'@uses can only link to string data'

, PDERROR\_UL\_IN\_UL =>  
'ul/ol tags cannot be directly nested inside ul/ol, nest inside li'

, PDERROR\_INVALID\_VALUES =>  
'command %s was passed "%s" but must be one of %s'

, PDERROR\_NESTED\_INTERNAL =>

```
'{@internal} cannot be nested inside {@internal}'  
,
```

PDERROR\_DANGEROUS\_PHP\_BUG\_EXISTS =&gt;  
 'Dangerous PHP Bug exists in PHP version %s that can be triggered '.  
 'by this parse (see PHP Bug #%s and PEAR Bug #%s)'  
,

PDERROR\_LOOP\_RECURSION\_LIMIT\_REACHED =&gt;  
 'An internal loop in PhpDocumentor has reached its preset '.  
 'recursion limit, preventing a possible infinite loop condition.'

) [line [555](#)]

## Error messages for phpDocumentor parser errors

- **Name** \$phpDocumentor\_error\_descrip

### \$phpDocumentor\_warning\_descrip

```
array = array(  

    PDERROR_MULTIPLE_PARENT =&gt;  

        'Class %s has multiple possible parents, package inheritance aborted'  
,
```

PDERROR\_PARENT\_NOT\_FOUND =&gt;  
 'Class %s parent %s not found'

,

PDERROR\_INHERITANCE\_CONFLICT =&gt;  
 'Class %s in file %s has multiple possible parents named %s. '.  
 'Cannot resolve name conflict,' . "n".  
 ' try ignoring a file that contains the conflicting parent class'

,

PDERROR\_UNKNOWN\_TAG =&gt;  
 'Unknown tag "@%s" used'

,

PDERROR\_IGNORE\_TAG\_IGNORED =&gt;  
 '@ignore tag used for %s element "%s" will be ignored'

,

PDERROR\_ELEMENT\_IGNORED =&gt;  
 "n". 'duplicate %s element "%s" in file %s will be  
 ignored.' . "n".  
 'Use an @ignore tag on the original '.  
 'if you want this case to be documented.'

,

PDERROR\_PARSEPRIVATE =&gt;  
 "n". 'entire page %s ignored because of @access private.' .  
 'Choose -pp to enable parsing of private elements'.

,

PDERROR\_CLASS\_PARENT\_NOT\_FOUND =&gt;  
 "n". 'class %s in package %s parent not found in @see parent::%s'.

,

PDERROR\_CLASS\_NOT\_IN\_PACKAGE =&gt;  
 "n". 'class %s was not found in package %s'.

, PDERROR\_DB\_TEMPLATE\_UNTERMINATED =&gt;  
  'docblock template never terminated with /\*\*#@-\*/'

, PDERROR\_PDF\_METHOD\_DOESNT\_EXIST =&gt;  
  '&lt;pdffunction:%s /&gt; called, but pdf method &quot;%s&quot; doesn\'t exist'

, PDERROR\_TUTORIAL\_NOT\_FOUND =&gt;  
  '&quot;tutorial \&quot;%s\&quot; not found, does it exist?&quot;'

, PDERROR\_CHILD\_TUTORIAL\_NOT\_FOUND =&gt;  
  'child tutorial &quot;%s&quot; listed in %s not found '.  
  'in parent package &quot;%s&quot; subpackage &quot;%s&quot;'

, PDERROR\_TUTORIAL\_SUBSECTION\_NOT\_FOUND =&gt;  
  'tutorial %s subsection &quot;%s&quot; doesn\'t exist, '.  
  'but its title was asked for'

, PDERROR\_NO\_PACKAGE\_TAG =&gt;  
  'no @package tag was used in a DocBlock for %s %s'

, PDERROR\_PRIVATE\_ASSUMED =&gt;  
  '%s &quot;%s&quot; is assumed to be @access private because its name '.  
  'starts with \_, but has no @access tag'

, PDERROR\_EXAMPLE\_NOT\_FOUND =&gt;  
  'example file &quot;%s&quot; does not exist'

, PDERROR\_SOURCE\_TAG\_IGNORED =&gt;  
  '{@source} can only be used in the long description, '.  
  'not in the short description: &quot;%s&quot;'

, PDERROR\_PACKAGECAT\_SET =&gt;  
  'package %s is already in category %s, '.  
  'will now replace with category %s'

, PDERROR\_SOURCECODE\_IGNORED =&gt;  
  'sourcecode command-line option is ignored '.  
  'when your PHP build has no tokenizer support'

, PDERROR\_INHERITDOC\_DONT\_WORK\_HERE =&gt;  
  '{@inheritdoc} can only be used in the docblock of a child class'

, PDERROR\_EMPTY\_EXAMPLE\_TITLE =&gt;  
  'Example file found at &quot;%s&quot; has no title, using &quot;%s&quot;'

, PDERROR\_DOCBLOCK\_CONFLICT =&gt;  
  'Page-level DocBlock precedes &quot;%s %s&quot;, '.  
  'use another DocBlock to document the source element'

, PDERROR\_NO\_PAGE\_LEVELDOCBLOCK =&gt;  
  'File &quot;%s&quot; has no page-level DocBlock, '.  
  'use @package in the first DocBlock to create one'

, PDERROR\_DOCBLOCK\_Goes\_CLASS =&gt;  
  'DocBlock would be page-level, but precedes class &quot;%s&quot;, '.  
  'use another DocBlock to document the file'

, PDERROR\_NO\_DOCBOOK\_ID =&gt;

```
'Tutorial section %s "%s" has no id="%s"{@id
subsection}" tag '.
'({@id} for refentry)'

',
PDERROR_BEAUTIFYING_FAILED =>
'Beautifying failed: %s'

',
PDERROR_NOTODO_INCLUDE =>
'@todo on an include element is ignored (line %s, file %s)'

',
PDERROR_UNDOCUMENTED_ELEMENT =>
'%s "%s" has no %s-level DocBlock.'

',
PDERROR_MISSING_PROPERTY_TAG_NAME =>
'@%s magic tag does not have name, illegal. Ignoring tag "%s %s %s"'

',
PDERROR_NAME_ALIAS_SAME_AS_TARGET =>
'@name value is the same as the filename it is supposed to alias'
) [line 433]
```

## Error messages for phpDocumentor parser warnings

- **Name** \$phpDocumentor\_warning\_descrip

*void* function addError(\$num, \$data...) [line [1110](#)]

**Function Parameters:**

- *integer* **\$num** error number from [Errors.inc](#)
- *string* **\$data...** up to 4 string parameters to sprintf() into the error string for error number \$num

## add an Error

- See [ErrorTracker::addError\(\)](#)
- **TODO** CS Cleanup - do I need to add \$data to the method signature?

*void* function addErrorDie(\$num, \$data...) [line [1134](#)]

**Function Parameters:**

- *integer \$num* error number from [Errors.inc](#)
  - *string \$data...* up to 4 string parameters to sprintf()  
error number \$num
- into the error string for

like [addError\(\)](#) but exits parsing

- **Global Variable Used** [ErrorTracker::\\$phpDocumentor\\_errors](#): repository for all errors generated by phpDocumentor
- See [ErrorTracker::addError\(\)](#)
- **TODO** CS Cleanup - do I need to add \$data to the method signature?

*void* function addWarning(\$num, \$data...) [[line 1160](#)]

**Function Parameters:**

- *integer \$num* warning number from [Errors.inc](#)
  - *string \$data...* up to 4 string parameters to sprintf()  
error number \$num
- into the error string for

**add a Warning**

- **Global Variable Used** [ErrorTracker::\\$phpDocumentor\\_errors](#): repository for all errors generated by phpDocumentor
- See [ErrorTracker::addWarning\(\)](#)
- **TODO** CS Cleanup - do I need to add \$data to the method signature?

PDERROR\_ACCESS\_WRONG\_PARAM = 8 [[line 77](#)]

**warning triggered when the arguments to @access are neither public nor private**

PDERROR\_BEAUTIFYING\_FAILED = 66 [[line 343](#)]

**warning triggered by an unterminated entity in a tutorial**

PDERROR\_CANNOT\_EXTEND\_SELF = 72 [[line 384](#)]

**warning triggered if someone brilliant tries "class X extends X {"**

PDERROR\_CANT\_HAVE\_INLINE\_IN\_TAGNAME = 36 [line [204](#)]

**warning triggered when an inline tag is found inside an xml tag name  
in a package page**

PDERROR\_CANT\_NEST\_IN\_B = 34 [line [195](#)]

**warning triggered when another tag is nested in &lt;b&gt;  
(not allowed in phpDocumentor)**

PDERROR\_CHILD\_TUTORIAL\_NOT\_FOUND = 40 [line [223](#)]

**warning triggered when a tutorial's child in the .ini file doesn't exist in the  
package and subpackage of the parent**

PDERROR\_CLASS\_CONFLICT = 17 [line [115](#)]

**warning triggered when classes in the same package have the same name**

PDERROR\_CLASS\_EXISTS = 4 [line [59](#)]

**warning triggered when inheritance could be from more than one class**

PDERROR\_CLASS\_NOT\_IN\_PACKAGE = 30 [line [176](#)]

**warning triggered when a getClassByPackage is called and can't find the class**

PDERROR\_CLASS\_PARENT\_NOT\_FOUND = 29 [line [172](#)]

**warning triggered when an entire page is ignored because of @access private**

- TODO I think this description is a copy/paste that was never updated

PDERROR\_CONVERTER\_NOT\_FOUND = 6 [line [68](#)]

**warning triggered when a converter is passed to  
[phpDocumentor\\_IntermediateParser::addConverter\(\)](#) that is not a class**

PDERROR\_CONVERTER\_OVR\_GFCT = 54 [line [293](#)]

**warning triggered when a child converter doesn't override  
getFormattedClassTrees()**

PDERROR\_DANGEROUS\_PHP\_BUG\_EXISTS = 80 [line [417](#)]

**warning triggered when the PHP version being used has dangerous bug/behavior**

PDERROR\_DB\_TEMPLATE\_UNTERMINATED = 32 [line [186](#)]

**warning triggered when a docblock template is never turned off**  
with /\*\*#@-\* / (no space)

PDERROR\_DOCBLOCK\_CONFLICT = 68 [line [367](#)]

**warning triggered by a page-level docblock preceding a source element**

```
1 <?php
2 /**
3  * Page-level DocBlock
4  * @package pagepackage
5  */
6 include 'file.php';
```

PDERROR\_DOCBLOCK\_Goes\_CLASS = 70 [line [376](#)]

**warning triggered when the first docblock in a file with a @package tag precedes a class. In this case, the class gets the docblock.**

PDERROR\_DUMB\_USES = 73 [line [388](#)]

**warning triggered by improper "@uses [blah](#)"**

PDERROR\_ELEMENT\_IGNORED = 25 [line [153](#)]

**warning triggered when a duplicate element is encountered that will be ignored by the documentor**

PDERROR\_EMPTY\_EXAMPLE\_TITLE = 63 [line [331](#)]

**warning triggered by @example path/to/example with no title**

PDERROR\_EXAMPLE\_NOT\_FOUND = 48 [line [265](#)]

**warning triggered when an example's path from @example /path/to/example.php is not found**

PDERROR\_FUNCTION\_HAS\_NONAME = 67 [line [354](#)]

**warning triggered by a function with no name**

```
function ($params)
{
} triggers this error
```

PDERROR\_GLOBAL\_NOT\_FOUND = 21 [line [133](#)]

**warning triggered when there are multiple @name tags in a docblock**

- **TODO** I think this description is a copy/paste that was never updated

PDERROR\_ID\_MUST\_INLINE = 51 [line [280](#)]

**warning triggered when an id attribute in a tutorial docbook tag is not an {@id} inline tag**

PDERROR\_IGNORE\_TAG\_IGNORED = 24 [line [148](#)]

**warning triggered when an @ignore tag is used in a DocBlock preceding a method, variable, include, or global variable**

PDERROR\_ILLEGAL\_PACKAGENAME = 14 [line [101](#)]

**warning triggered when the package or subpackage name is illegal**

PDERROR\_INHERITANCE\_CONFLICT = 5 [line [63](#)]

**warning triggered when inheritance could be from more than one class**

PDERROR\_INHERITDOC\_DONT\_WORK\_HERE = 62 [line [327](#)]

**warning triggered by {@inheritdoc} in a non-inheritable situation**

PDERROR\_INLINETAG\_IN\_SEE = 50 [line [275](#)]

**warning triggered when an example's path from @example /path/to/example.php is not found**

PDERROR\_INTERNAL\_NOT\_CLOSED = 52 [line [284](#)]

**warning triggered when an {@internal} tag is not closed**

PDERROR\_INVALID\_VALUES = 75 [line [396](#)]

**warning triggered if a command line option does not have a valid value passed in**

PDERROR\_LOOP\_RECURSION\_LIMIT\_REACHED = 82 [line [426](#)]

**warning triggered when the a loop recursion tripwire has been tripped**

PDERROR\_MALFORMED\_GLOBAL\_TAG = 23 [line [143](#)]

**warning triggered when there are multiple @name tags in a docblock**

- **TODO** I think this description is a copy/paste that was never updated

PDERROR\_MALFORMED\_TAG = 60 [line [319](#)]

**warning triggered by an empty tag**

PDERROR\_MISSING\_PROPERTY\_TAG\_NAME = 79 [line [413](#)]

**warning triggered when any of {@property}, {@property-read}, {@property-write}}, or {@method} tag does not have name**

PDERROR\_MULTIPLE\_ACCESS\_TAGS = 9 [line [81](#)]

**warning triggered when there are multiple @access tags in a docblock**

PDERROR\_MULTIPLE\_CATEGORY\_TAGS = 61 [line [323](#)]

**warning triggered by more than 1 @category tag**

PDERROR\_MULTIPLE\_GLOBAL\_TAGS = 22 [line [138](#)]

**warning triggered when there are multiple @name tags in a docblock**

- **TODO** I think this description is a copy/paste that was never updated

PDERROR\_MULTIPLE\_NAME\_TAGS = 19 [line [123](#)]

**warning triggered when there are multiple @name tags in a docblock**

PDERROR\_MULTIPLE\_PACKAGE\_TAGS = 12 [line [93](#)]

**warning triggered when there are multiple @package tags in a docblock**

PDERROR\_MULTIPLE\_PARENT = 1 [line [46](#)]

**warning triggered when inheritance could be from more than one class**

PDERROR\_MULTIPLE\_RETURN\_TAGS = 10 [line [85](#)]

**warning triggered when there are multiple @return tags in a docblock**

PDERROR\_MULTIPLE\_SUBPACKAGE\_TAGS = 13 [line [97](#)]

**warning triggered when there are multiple @subpackage tags in a docblock**

PDERROR\_MULTIPLE\_VAR\_TAGS = 11 [line [89](#)]

**warning triggered when there are multiple @var tags in a docblock**

PDERROR\_NAME\_ALIAS\_SAME\_AS\_TARGET = 81 [line [422](#)]

**warning triggered when the alias value in an page-level docblock's @name tag  
is the same value as the target filename is it supposed to alias**

PDERROR\_NEED\_WHITESPACE = 28 [line [167](#)]

**warning triggered when an entire page is ignored because of @access private**

- **TODO** I think this description is a copy/paste that was never updated

PDERROR\_NESTED\_INTERNAL = 76 [line [400](#)]

**warning triggered when {@internal} is nested inside another {@internal}**

PDERROR\_NOTODO\_INCLUDE = 77 [line [404](#)]

**warning triggered when @todo is used on an include element**

PDERROR\_NO\_CONVERTERS = 7 [line [73](#)]

**warning triggered when a converter is passed to**

[phpDocumentor\\_IntermediateParser::addConverter\(\)](#) that is not a class

PDERROR\_NO\_CONVERTER\_HANDLER = 49 [line [270](#)]

**warning triggered when an example's path from @example /path/to/example.php  
is not found**

PDERROR\_NO\_DOCBOOK\_ID = 71 [line [380](#)]

**warning triggered in tutorial parsing if there is a missing {@id} inline tag**

PDERROR\_NO\_PACKAGE\_TAG = 46 [line [254](#)]

**warning triggered when no @package tag is used in a page-level  
or class-level DocBlock**

PDERROR\_NO\_PAGE\_LEVELDOCBLOCK = 69 [line [371](#)]

**warning triggered when a file does not contain a page-level docblock**

PDERROR\_OVERRIDDEN\_PACKAGE\_TAGS = 15 [line [106](#)]

**warning triggered when there a @package tag is used in a function,  
define, method, var or include**

PDERROR\_OVERRIDDEN\_SUBPACKAGE\_TAGS = 16 [line [111](#)]

**warning triggered when there a @subpackage tag is used in a function,  
define, method, var or include**

PDERROR\_PACKAGECAT\_SET = 55 [line [298](#)]

**warning triggered when a package is already associated with a category, and  
a new association is found**

PDERROR\_PACKAGEOUTPUT\_DELETES\_PARENT\_FILE = 20 [line [128](#)]

**warning triggered when there are multiple @name tags in a docblock**

- **TODO** I think this description is a copy/paste that was never updated

PDERROR\_PARENT\_NOT\_FOUND = 2 [line [50](#)]

## **warning triggered when parent class doesn't exist**

PDERROR\_PARSEPRIVATE = 26 [line [157](#)]

**warning triggered when an entire page is ignored because of @access private**

PDERROR\_PDFFUNCTION\_NO\_FUNC = 41 [line [228](#)]

**warning triggered when a <pdffunction:funcname /> tag is used in the PDF**

Converter and no funcname is present (<pdffunction: />)

PDERROR\_PDF\_METHOD\_DOESNT\_EXIST = 42 [line [233](#)]

**warning triggered when a <pdffunction:funcname /> tag is used in the PDF**

Converter and funcname is not a [Cezpdf](#) method

PDERROR\_PDF\_TEMPVAR\_DOESNT\_EXIST = 43 [line [239](#)]

**warning triggered when a <pdffunction:funcname arg=\$tempvar/> tag**

is used in the PDF Converter and "tempvar" is not set from the return of a previous pdffunction tag

PDERROR\_PRIVATE\_ASSUMED = 47 [line [260](#)]

**warning triggered when no @access private tag is used in a**

global variable/method/var with \_ as first char in name and --pear was specified

PDERROR\_SOURCECODE\_IGNORED = 59 [line [315](#)]

**warning triggered by sourcecode="on", if PHP < 4.3.0**

PDERROR\_SOURCE\_TAG\_FUNCTION\_NOT\_FOUND = 31 [line [181](#)]

**warning triggered when a { @source } inline tag is used in a docblock not preceding a function**

PDERROR\_SOURCE\_TAG\_IGNORED = 53 [line [288](#)]

**warning triggered when an {@source} tag is found in a short description**

PDERROR\_TAG\_NOT\_HANDLED = 58 [line [311](#)]

**warning triggered by @filesource, if PHP < 4.3.0**

PDERROR\_TEMPLATEDIR\_DOESNT\_EXIST = 64 [line [335](#)]

## **warning triggered by non-existent template directory**

PDERROR\_TEXT\_OUTSIDE\_LI = 56 [line [303](#)]

**warning triggered when text in a docblock list is not contained in  
an <li> opening tag**

PDERROR\_TUTORIAL\_IS\_OWN\_CHILD = 38 [line [213](#)]

**warning triggered when a tutorial lists itself as a child tutorial**

PDERROR\_TUTORIAL\_IS\_OWN\_GRANDPA = 39 [line [218](#)]

**warning triggered when a tutorial's child lists the parent tutorial  
as a child tutorial**

PDERROR\_TUTORIAL\_NOT\_FOUND = 37 [line [209](#)]

**warning triggered when a tutorial is referenced  
via @tutorial/{ @tutorial} and is not found**

PDERROR\_TUTORIAL\_SUBSECTION\_NOT\_FOUND = 44 [line [244](#)]

**warning triggered when a subsection's title is asked for, but the subsection  
is not found**

PDERROR\_UL\_IN\_UL = 74 [line [392](#)]

**warning triggered if <ul> is nested inside <ul> and not <li>**

PDERROR\_UNCLOSED\_TAG = 57 [line [307](#)]

**warning triggered when a DocBlock html tag is unclosed**

PDERROR\_UNDOCUMENTED\_ELEMENT = 78 [line [408](#)]

**warning triggered when a class or method hasn't got docblock**

PDERROR\_UNKNOWN\_COMMANDLINE = 27 [line [162](#)]

**warning triggered when an entire page is ignored because of @access private**

- **TODO** I think this description is a copy/paste that was never updated

PDERROR\_UNKNOWN\_TAG = 18 [line [119](#)]

**warning triggered when classes in the same package have the same name**

PDERROR\_UNMATCHED\_LIST\_TAG = 33 [line [190](#)]

**warning triggered when a docblock has an unmatched &lt;ol&gt; or &lt;ul&gt;;**

PDERROR\_UNMATCHED\_TUTORIAL\_TAG = 35 [line [199](#)]

**warning triggered when a docbook tag is not properly matched**

PDERROR\_UNTERMINATED\_ATTRIB = 45 [line [249](#)]

**warning triggered when a subsection's title is asked for, but the subsection is not found**

PDERROR\_UNTERMINATED\_ENTITY = 65 [line [339](#)]

**warning triggered by an unterminated entity in a tutorial**

PDERROR\_UNTERMINATED\_INLINE\_TAG = 3 [line [55](#)]

**warning triggered when an {@inline tag} is not terminated**  
(no } before the \* / ending the comment)

## Class ErrorTracker [line [894](#)]

**contains all the errors/warnings**

- **Package** phpDocumentor
- **Sub-Package** Errors
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net) >
- **Version** Release: @VER@
- **Copyright** 2001-2008 Gregory Beaver
- See [ErrorTracker::\\$errors](#), [ErrorTracker::\\$warnings](#)

- [Link http://pear.php.net/PhpDocumentor](http://pear.php.net/PhpDocumentor)
- [Link http://www.phpdoc.org](http://www.phpdoc.org)
- TODO CS cleanup - change package to PhpDocumentor
- License [LGPL](#)

### ErrorTracker::\$curfile

*string* = " [line [909](#)]

### ErrorTracker::\$errors

*array* = *array()* [line [900](#)]

array of [RecordErrors](#)

### ErrorTracker::\$lasterror

*integer|false* = false [line [919](#)]

index in [\\$errors](#) of last error triggered

### ErrorTracker::\$lastwarning

*integer|false* = false [line [925](#)]

index in [\\$warnings](#) of last warning triggered

### ErrorTracker::\$linenum

*integer* = 0 [line [913](#)]

### ErrorTracker::\$warnings

*array* = *array()* [line [905](#)]

array of [RecordWarnings](#)

*void* function ErrorTracker::addError(\$num, \$data...) [line [993](#)]

**Function Parameters:**

- *integer \$num* error number from [Errors.inc](#)
- *string \$data...* up to 4 string parameters to sprintf()  
error number \$num

into the error string for

add a new error to the [\\$errors](#) array

- **TODO** CS Cleanup - do I need to add \$data to the method signature?

*void* function ErrorTracker::addErrorReturn(\$num, \$data...) [line [1016](#)]

**Function Parameters:**

- *integer* **\$num** error number from [Errors.inc](#)
- *string* **\$data...** up to 4 string parameters to sprintf() into the error string for error number \$num

**add a new error to the [\\$errors](#) array and returns the error string**

- **TODO** CS Cleanup - do I need to add \$data to the method signature?

*void* function ErrorTracker::addWarning(\$num, \$data...) [line [970](#)]

**Function Parameters:**

- *integer* **\$num** error number from [Errors.inc](#)
- *string* **\$data...** up to 4 string parameters to sprintf() into the error string for error number \$num

**add a new warning to the [\\$warnings](#) array**

- **TODO** CS Cleanup - do I need to add \$data to the method signature?

*void* function ErrorTracker::handleEvent(\$num, \$data) [line [940](#)]

**Function Parameters:**

- *integer \$num* parser-passed event (see [PHPDOCUMENATOR\\_EVENT\\_NEWLINENUM](#), [PHPDOCUMENATOR\\_EVENT\\_NEWFILE](#))
- *mixed \$data* either a line number if \$num is [PHPDOCUMENATOR\\_EVENT\\_NEWLINENUM](#) or a file name [PHPDOCUMENATOR\\_EVENT\\_NEWFILE](#) if \$num is

**This function subscribes to two events in the Parser in order to keep track of line number information and file name.**

*array* function ErrorTracker::returnErrors() [[line 1045](#)]

**Get sorted array of all non-fatal errors in parsing/conversion**

*string* function ErrorTracker::returnLastError() [[line 1077](#)]

**Get the error message of the last error**

*string* function ErrorTracker::returnLastWarning() [[line 1087](#)]

**Get the warning message of the last warning**

*array* function ErrorTracker::returnWarnings() [[line 1034](#)]

**Get sorted array of all warnings in parsing/conversion**

## Class RecordError [[line 843](#)]

**encapsulates error information**

- **Package** phpDocumentor
- **Sub-Package** Errors
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2001-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

### **RecordError::\$type**

*string = 'phpDocumentor\_error\_descrip' [line 849]*

**name of global variable that descriptors for this warning/error is kept**

*string function RecordError::output([\$string = false]) [line 858]*

**Function Parameters:**

- *string \$string* the error to print

**prints the error**

## **Class RecordWarning**

*[line 746]*

**encapsulates warning information**

- **Package** phpDocumentor
- **Sub-Package** Errors
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2001-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

### **RecordWarning::\$data**

*string = [line 767]*

**error string**

### **RecordWarning::\$file**

*string = false* [line [757](#)]

### **file this error occurred in**

#### **RecordWarning::\$linenum**

*integer =* [line [762](#)]

### **line number of the file this error occurred in**

#### **RecordWarning::\$num**

*string =* [line [773](#)]

### **error number**

- See [Errors.inc](#)

#### **RecordWarning::\$type**

*string = 'phpDocumentor\_warning\_descrip'* [line [752](#)]

### **name of global variable that descriptors for this warning/error is kept**

Constructor *void* function RecordWarning::RecordWarning(\$file, \$linenum, \$num, \$data...) [line [785](#)]

#### **Function Parameters:**

- *string \$file* filename this error occurred in ([\\$file](#))
- *integer \$linenum* line number this error occurred on ([\\$linenum](#))
- *integer \$num* Error number defined in [Errors.inc](#)
- *string \$data...* variable number of strings, up to 4,

### **Constructor**

- **TODO CS Cleanup** - do I need to add \$data to the method signature? to  
sprintf based on the error number

*void* function RecordWarning::output([*\$string* = false]) [*line 809*]

**Function Parameters:**

- *string* **\$string** the warning to print

**prints the warning**

# InlineTags.inc

All abstract representations of inline tags are in this file

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: InlineTags.inc 286921 2009-08-08 05:01:24Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** separate file since 1.2
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

## Class parserExampleInlineTag

[line [567](#)]

**Represents the example inline tag, used to display an example file inside a docblock or tutorial**

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@example}](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **See** [parserStringWithInlineTags](#)
- **Link** <http://www.phpdoc.org>
- **Link** <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

Constructor *mixed* function parserExampleInlineTag::parserExampleInlineTag(\$value, \$current\_path, [\$isTutorial = false]) [line [582](#)]

**Function Parameters:**

- **string \$value** format "filepath[ start [end]]"  
where start and end  
are line numbers with the end line number optional
- **string \$current\_path** full path to the current file,  
directory locations used to check relative
- **bool \$isTutorial** if true, then this is in a tutorial

## constructor

- **TODO** replace tokenizer\_ext constant with TOKENIZER\_EXT for CS rule

*string* function parserExampleInlineTag::arrayConvert(&\$c) [line [733](#)]

**Function Parameters:**

- [Converter](#) &\$c output converter

## converter helper for PHP 4.3.0+

- **Uses** [phpDocumentor\\_HighlightParser](#) - Parses the tokenized source

*string* function parserExampleInlineTag::getProgramListing() [[line 752](#)]

**Return the source for the example file, enclosed in a <programlisting> tag to use in a tutorial**

*void* function parserExampleInlineTag::setSource(\$source, [\$class = false]) [[line 719](#)]

**Function Parameters:**

- **string/array \$source** source code
- **string/bool \$class** class name if this is a method,  
if this is a method this will be true  
boolean in php 4.3.0,

**sets the source**

## Class parserIdInlineTag

[[line 835](#)]

**Represents the inline {@id} tag for tutorials**

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@id}](#)
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **See** [parserStringWithInlineTags](#)
- **Link** <http://www.phpdoc.org>
- **Link** <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

**parserIdInlineTag::\$category**

*string = 'default' [line [851](#)]*

**category of the {@id}**

**parserIdInlineTag::\$id**

*string = [line [866](#)]*

**section/subsection name**

**parserIdInlineTag::\$inlinetype**

*string = 'id' [line [841](#)]*

**always 'id'**

**parserIdInlineTag::\$package**

*string = 'default' [line [846](#)]*

**package of the {@id}**

**parserIdInlineTag::\$subpackage**

*string = " [line [856](#)]*

**subpackage of the {@id}**

**parserIdInlineTag::\$tutorial**

*string = [line [861](#)]*

**full name of the tutorial**

Constructor `void` function `parserIdInlineTag::parserIdInlineTag($category, $package, $subpackage, $tutorial, [$id = false])` [line [877](#)]

**Function Parameters:**

- `string $category` category name
- `string $package` package name
- `string $subpackage` subpackage name
- `string $tutorial` tutorial name

- *string \$id* section/subsection name

## constructor

*string* function parserIdInlineTag::Convert(&\$c) [line [895](#)]

**Function Parameters:**

- [Converter](#) &\$c output converter

## converter

- **TODO** CS cleanup - rename to convert for camelCase rule
- **Uses** [Converter::getTutorialId\(\)](#) - retrieve converter-specific ID

# Class parserInheritdocInlineTag

[line [789](#)]

Represents the inheritdoc inline tag, used by classes/methods/vars to inherit documentation from the parent class if possible

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@inheritDoc}](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **See** [parserStringWithInlineTags](#)
- **Link** <http://www.phpdoc.org>
- **Link** <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

### **parserInheritdocInlineTag::\$inlinetype**

*string = 'inheritdoc' [line [795](#)]*

**always 'inheritdoc'**

Constructor *void* function `parserInheritdocInlineTag::parserInheritdocInlineTag()` [[line 800](#)]

**Does nothing, overrides parent constructor**

*string* function `parserInheritdocInlineTag::Convert()` [[line 810](#)]

**only sets a warning and returns empty**

- **TODO** CS cleanup - rename to convert for camelCase rule

## **Class parserInlineTag**

*[line 61]*

**Use this element to represent an {@inline tag} like {@link}**

- **Package** [phpDocumentor](#)
- **Sub-Package** [InlineTags](#)
- **Tutorial** [phpDocumentor Inline tags](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **See** [parserStringWithInlineTags](#)
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1
- **License** [LGPL](#)

### **parserInlineTag::\$inlinetype**

*string = " [line [79](#)]*

**the name of the inline tag (like link)**

### **parserInlineTag::\$type**

*string = 'inlinetag' [line [74](#)]*

#### **Element type**

Type is used by many functions to skip the hassle of

```
1 if phpDocumentor_get_class($blah) == 'parserBlah'  
always "inlinetag"
```

Constructor void function parserInlineTag::parserInlineTag(\$type, \$value) [line [87](#)]

**Function Parameters:**

- **string \$type** tag type (example: link)
- **string \$value** tag value (example: what to link to)

### **sets up the tag**

*string* function parserInlineTag::getString() [line [117](#)]

**always gets an empty string**

- See [parserStringWithInlineTags::getString\(\)](#)
- See [parserStringWithInlineTags::trimmedStrlen\(\)](#)
- Used by [parserStringWithInlineTags::getString\(\)](#) - removes inline tag length, as it is indeterminate until conversion.

*integer* function parserInlineTag::Strlen() [line [99](#)]

**get length of the tag**

- **TODO** CS cleanup - rename to strLen for camelCase rule

## Class parserLinkInlineTag [line [142](#)]

**represents inline links**

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@link}](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **See** [parserStringWithInlineTags](#)
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1
- **License** [LGPL](#)

### parserLinkInlineTag::\$linktext

*string = " [line [149](#)]*

**text to display in the link, can be different from the link for standard  
links like websites**

Constructor void function parserLinkInlineTag::parserLinkInlineTag(\$link, \$text) [line [157](#)]

**Function Parameters:**

- *string* **\$link** stored in **\$value**, see [parserBase::\\$value](#)
- *string* **\$text** see [\\$linktext](#)

### sets up the tag

*false|*string function parserLinkInlineTag::Convert(&\$c) [[line 179](#)]

**Function Parameters:**

- [Converter](#) &\$c converter used to change the abstract link into text for display

**calls the output conversion**

- **TODO** CS cleanup - rename to convert for camelCase rule

*string* function parserLinkInlineTag::ConvertPart(&\$c, \$value) [[line 204](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter
- *string* \$value the tag value

**convert part of the tag**

- **TODO** CS cleanup - rename to convertPart for camelCase rule

## Class parserSourceInlineTag

[[line 377](#)]

**represents inline source tag, used for function/method source**

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@source}](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- See [parserStringWithInlineTags](#)
- Link <http://www.phpdoc.org>
- Link <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

### parserSourceInlineTag::\$end

`'*'|integer = '*' [line 395]`

#### Last line to display

- **Var** If '\*' then the whole source will be used, otherwise numbers will be displayed the [\\$start](#) to \$end line

### parserSourceInlineTag::\$inlinetype

`string = 'source' [line 383]`

#### always 'source'

### parserSourceInlineTag::\$source

`string|array = false [line 401]`

**tokenized source organized by line numbers for php 4.3.0+, the old {@source} tag used a string**

### parserSourceInlineTag::\$start

`integer = 1 [line 389]`

#### First line of source code to display

- See [parserSourceInlineTag::\\$end](#)

Constructor void function parserSourceInlineTag::parserSourceInlineTag(\$value) [line [414](#)]

**Function Parameters:**

- **string \$value** format "start [end]", where start and end are line numbers with the end line number optional

## constructor

string function parserSourceInlineTag::arrayConvert(&\$c) [line [497](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter object

## converter helper used in PHP 4.3.0+

- **Used by** [parserSourceInlineTag::Convert\(\)](#) - in PHP 4.3.0+, this method is used to convert
- **Uses** [phpDocumentor\\_HighlightParser](#) - Parses the tokenized source

string function parserSourceInlineTag::Convert(&\$c) [line [481](#)]

**Function Parameters:**

- [Converter](#) &\$c the output converter object

## convert the tag

- **TODO** CS cleanup - rename to convert for camelCase rule
- **Uses** [parserSourceInlineTag::stringConvert\(\)](#) - in PHP 4.2.3-, this method is used to convert

- **Uses** [parserSourceInlineTag::arrayConvert\(\)](#) - in PHP 4.3.0+, this method is used to convert

*string* function parserSourceInlineTag::getString() [line [444](#)]  
**gets the source string**

*void* function parserSourceInlineTag::setSource(\$source, [\$class = false]) [line [459](#)]

**Function Parameters:**

- *string/array* **\$source** source code
- *string/bool* **\$class** class name if this is a method,  
if this is a method this will be true

boolean in php 4.3.0,

**sets the source tag's value**

*string* function parserSourceInlineTag::stringConvert(&\$c) [line [520](#)]

**Function Parameters:**

- [Converter](#) &**\$c** the output converter object

**converter helper used in PHP 4.2.3-**

- **Deprecated** in favor of PHP 4.3.0+ [arrayConvert\(\)](#)
- **Used by** [parserSourceInlineTag::Convert\(\)](#) - in PHP 4.2.3-, this method is used to convert
- **Uses** [Converter::unmangle\(\)](#) - remove the extraneous stuff from  
[http://www.php.net/highlight\\_string](http://www.php.net/highlight_string)

*int* function parserSourceInlineTag::Strlen() [line [434](#)]

**only used to determine blank lines. {@source} will not be blank, probably**

## Class parserToCInlineTag

[line 923]

## Represents {@toc} for table of contents generation in tutorials

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@toc}](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **See** [parserStringWithInlineTags](#)
- **Link** <http://www.phpdoc.org>
- **Link** <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

### parserTocInlineTag::\$inlinetype

*string = 'toc' [line 929]*

#### always 'toc'

Constructor **void** function parserTocInlineTag::parserTocInlineTag() [line 956]  
**constructor**

*mixed* function parserTocInlineTag::Convert(&\$c) [line 1015]

#### Function Parameters:

- [Converter](#) &\$c converter object

### converter method

array(  
'tagname' => string name of tag,  
'link' => [tutorialLink](#) to the tutorial,  
'id' => converter specific tutorial ID from  
      [Converter::getTutorialId\(\)](#)  
'title' => title of the tutorial) and returns the results as the table of contents

- **TODO** CS cleanup - rename to convert for camelCase rule
- **Uses** [Converter::getTutorialId\(\)](#) - retrieve the tutorial ID for
- **Uses** [Converter::formatTutorialTOC\(\)](#) - passes an array of format:

*void function parserTocInlineTag::setPath(\$path) [line [990](#)]*

**Function Parameters:**

- *string \$path* the path

### **set the path**

*void function parserTocInlineTag::setTOC(\$toc) [line [978](#)]*

**Function Parameters:**

- *array \$toc* format: array(  
    array(  
        'tag' => [parserXMLDocBookTag](#),  
        'id' => [parserIdInlineTag](#),  
        'title' => [title](#)  
    ),  
    ...  
)

### **set the TOC**

## Class parserTutorialInlineTag

*[line [278](#)]*

**Represents inline links to external tutorial documentation**

- **Package** phpDocumentor
- **Sub-Package** InlineTags
- **Tutorial** [inline {@tutorial}](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>

- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- See [parserStringWithInlineTags](#)
- Link <http://www.phpdoc.org>
- Link <http://pear.php.net/PhpDocumentor>
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

Constructor `void` function `parserTutorialInlineTag::parserTutorialInlineTag($link, $text)` [[line 286](#)]  
**Function Parameters:**

- `string` **\$link** stored in `$value`, see [parserBase::\\$value](#)
- `string` **\$text** see [\\$linktext](#)

## constructor

*mixed* function `parserTutorialInlineTag::Convert(&$c)` [[line 302](#)]  
**Function Parameters:**

- [Converter](#) &**\$c** converter used to change the abstract link into text for display

## convert part of the tag

- **TODO** CS cleanup - rename to convert for camelCase rule

# LinkClasses.inc

**Linking to element documentation is performed by the classes in this file.**

abstractLink descendants contain enough information to differentiate every documentable element, and so can be converted to a link string by [Converter::returnSee\(\)](#)

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: LinkClasses.inc 253641 2008-02-24 02:35:44Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2.0
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

# Class abstractLink

[line 61]

linking classes parent

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

**abstractLink::\$category**

*string = " [line 77]*

**abstractLink::\$fileAlias**

*string = " [line 70]*

**phpdoc alias \_phpdoc\_inc for phpdoc.inc**

**abstractLink::\$name**

*string = " [line 76]*

**abstractLink::\$package**

*string = " [line 78]*

**abstractLink::\$path**

*string = [line 66]*

**abstractLink::\$subpackage**

*string = " [line 79]*

**abstractLink::\$type**

*string = " [line 75]*

**element type linked to.**

can only be a documentable element

*void* function abstractLink::addLink(\$path, \$fileAlias, \$name, \$package, \$subpackage, [\$category = false])  
[line [94](#)]

**Function Parameters:**

- *string* **\$path** full path to file containing element
- *string* **\$fileAlias** page name, as configured by Parser::parse
- *string* **\$name** element name
- *string* **\$package** package element is in
- *string* **\$subpackage** subpackage element is in
- *string* **\$category** optional category that documentation is in

**sets up the link**

## Class classLink

[line [213](#)]

**class link**

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

**classLink::\$type**

*string* = 'class' [line [218](#)]

## Class constLink

[line 307]

### class constant link

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### constLink::\$type

*string = 'const'* [line 312]

## Class defineLink

[line 167]

### define link

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### defineLink::\$type

*string = 'define'* [line 172]

## Class functionLink

[line [144](#)]

### function link

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### functionLink::\$type

*string = 'function'* [line [149](#)]

## Class globalLink

[line [190](#)]

### global variable link

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*

- License [LGPL](#)

### globalLink::\$type

*string = 'global' [line [195](#)]*

## Class methodLink [line [236](#)]

### method link

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- License [LGPL](#)

### methodLink::\$class

*string = " [line [245](#)]*

### methodLink::\$type

*string = 'method' [line [241](#)]*

*void function methodLink::addLink(\$class, \$path, \$fileAlias, \$name, \$package, \$subpackage, [\$category = false]) [line [260](#)]*

#### Function Parameters:

- *string \$class* class name
- *string \$path* full path to file containing element
- *string \$fileAlias* page name, as configured by Parser::parse
- *string \$name* element name
- *string \$package* package element is in
- *string \$subpackage* subpackage element is in
- *string \$category* optional category that documentation is in

**sets up the link**

## Class pageLink [line [121](#)]

**procedural page link**

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

**pageLink::\$type**

*string = 'page' [line [126](#)]*

## Class tutorialLink [line [330](#)]

**tutorial link**

- **Package** phpDocumentor
- **Sub-Package** Links
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>

- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### **tutorialLink::\$section**

*string = " [line [336](#)]*

### **tutorialLink::\$title**

*string = false [line [337](#)]*

### **tutorialLink::\$type**

*string = 'tutorial' [line [335](#)]*

*void function tutorialLink::addLink(\$section, \$path, \$name, \$package, \$subpackage, [\$title = false], [\$category = false]) [line [353](#)]*

#### **Function Parameters:**

- *string \$section* section/subsection name
- *string \$path* full path to file containing element
- *string \$name* page name, as configured by Parser::parse
- *string \$package* package element is in
- *string \$subpackage* subpackage element is in
- *string \$title* title of tutorial
- *string \$category* optional category that documentation is in

### **sets up the link**

## **Class varLink**

*[line [284](#)]*

### **class variable link**

- **Package** phpDocumentor

- **Sub-Package** Links
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [GPL](#)

### **varLink::\$type**

*string = 'var' [line [289](#)]*

# ParserData.inc

## Parser Data Structures

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** ParserData
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: ParserData.inc 253814 2008-02-26 12:15:56Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.0rc1
- **License** [LGPL](#)

## Class parserBase [line 710]

**Base class for all elements**

- **Package** phpDocumentor
- **Sub-Package** ParserData
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Abstract Element**
- **Since** 1.0rc1
- **License** [LGPL](#)

### **parserBase::\$type**

*string = 'base' [line 717]*

**Type is used by many functions to skip the hassle of if**

`phpDocumentor_get_class($blah) == 'parserBlah'... always base`

### **parserBase::\$value**

*mixed = false [line 723]*

**set to different things by its descendants**

- **Abstract Element**

*string function parserBase::getType() [line 730]*

**gets the type**

*mixed function parserBase::getValue() [line 752]*

**gets the value**

*void function parserBase::setValue(\$value) [line 742]*

**Function Parameters:**

- *mixed \$value* set the value of this element

**sets the given value**

## Class parserData

[line 363]

Contains an in-memory representation of all documentable elements ([parserPage](#), [parserFunction](#), [parserDefine](#), [parserInclude](#), [parserClass](#), [parserMethod](#), [parserVar](#)) and their DocBlocks ([parserDocBlock](#)).

This class works in coordination with [phpDocumentor\\_IntermediateParser](#) to take output from Parser::handleEvent() and create indexes, links, and other assorted things (all documented in [phpDocumentor\\_IntermediateParser](#) and [Converter](#))

- **Package** phpDocumentor
- **Sub-Package** ParserData
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.0rc1
- **License** [LGPL](#)

### parserData::\$classElements

*array = array()* [line 395]

**array of parsed class elements**

### parserData::\$clean

*boolean = true* [line 419]

**used by**[phpDocumentor\\_IntermediateParser::handleDocBlock\(\)](#)

determine whether a docblock is a page-level docblock or not. \$clean is true as long

as only 0 or 1 docblock has been parsed, and no element other than parserPage has been parsed

#### parserData::\$docblock

*mixed = false [line 424]*

**DocBlock [parserDocBlock](#)for this page, or false if not set**

#### parserData::\$elements

*array = array() [line 375]*

**array of parsed elements**

#### parserData::\$links

*array = array() [line 411]*

**array of links descended from [abstractLink](#)**

- See [pageLink](#), [defineLink](#), [classLink](#), [functionLink](#), [methodLink](#), [varLink](#)

#### parserData::\$parent

*false|parserPage = false [line 370]*

**[parserPage](#)element that is this parserData's parent, or false if not set.**

#### parserData::\$privateclasselements

*array = array() [line 405]*

**array of parsed class elements with @access private**

#### parserData::\$privateelements

*array = array() [line 390]*

**array of parsed elements with @access private**

**parserData::\$tutorial**

*parserTutorial/false = false [line 400]*

**parserData::\$type**

*string = 'page' [line 438]*

### Type is used by many functions to skip the hassle of if

1     [phpDocumentor\\_get\\_class\(\\$blah\)](#) == 'parserBlah'  
always 'page', used in element indexing and conversion functions found in [Converter](#)

*void function parserData::addElement(&\$element) [line 449]*

#### **Function Parameters:**

- **[parserElement](#) &\$element** add a parsed element to the [\\$elements](#) array,  
also sets [\\$clean](#) to false

### add a new element to the tracking array

*string function parserData::addLink(&\$element, [\$classorpackage = "], [\$subpackage = "]) [line 553]*

#### **Function Parameters:**

- **[parserElement](#) &\$element** element to add a new link (descended from [abstractLink](#)) to the [\\$links](#) array
- **string \$classorpackage** classname for elements that are class-based (this may be deprecated in the future, as the classname should be contained within the element. if \$element is a page, this parameter is a package name
- **string \$subpackage** subpackage name for page elements

### adds a link

*void function parserData::addTutorial(\$t, &\$c) [line 499]*

#### **Function Parameters:**

- **[parserTutorial](#) \$t** a tutorial parser
- **[Converter](#) &\$c** the output converter

### adds a tutorial parser

*bool* function parserData::explicitDocBlock() [line 532]

**Tells this page that its DocBlock was not implicit**

*array* function parserData::getClasses(&\$c) [line 633]

**Function Parameters:**

- Converter &\$c output converter

**returns a list of all classes declared in a file**

*string* function parserData::getLink(&\$c, [\$text = false]) [line 615]

**Function Parameters:**

- Converter &\$c the output converter
- *bool* \$text a text flag

**returns a link**

*string* function parserData::getName() [line 650]

**Get the output-safe filename (. changed to \_)**

*tutorialLink* function parserData::getTutorial() [line 512]

**If this file has a tutorial associated with it, returns a link to the tutorial.**

*boolean* function parserData::hasClasses() [line 486]

**Does this package have classes?**

*bool* function parserData::hasExplicitDocBlock() [line 522]

**If the page-level DocBlock was present in the source, returns true**

*bool* function parserData::hasInterfaces() [line 476]

**Does this package have interfaces?**

*bool* function parserData::isClean() [line 674]

**checks if the element is "cleaned" already**

*void* function parserData::setDocBlock(&\$docblock) [*line 687*]

**Function Parameters:**

- [\*parserDocBlock\*](#) &\$docblock docblock element

**sets the docblock**

- See [\*parserDocBlock\*](#)

*void* function parserData::setParent(&\$parent) [*line 664*]

**Function Parameters:**

- [\*parserPage\*](#) &\$parent parent element of this parsed data

**sets the parent**

## Class parserPage

[*line 59*]

**Contains information about a PHP file, used to group procedural elements together.**

- **Package** phpDocumentor
- **Sub-Package** ParserData
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.0rc1

- License [LGPL](#)

**parserPage::\$category**

*string = 'default' [line 97]*

**parserPage::\$file**

*string = " [line 76]*

**filename.ext (no path)**

**parserPage::\$id**

*string = " [line 71]*

**not implemented in this version, will be used to link xml output pages**

**parserPage::\$modDate**

*string = " [line 115]*

**not implemented yet**

file modification date, will be used for makefiles

**parserPage::\$name**

*string = " [line 86]*

**phpdoc-safe name (only letters, numbers and \_)**

**parserPage::\$origName**

*string = " [line 93]*

**original phpdoc-safe name (only letters, numbers and \_)**

This fixes [ 1391432 ] Too many underscores in include links.

**parserPage::\$package**

*string = 'default' [line 101]*

**parserPage::\$packageOutput**

*mixed = false [line 132]*

## **Used to limit output, contains contents of --packageoutput commandline.**

Does not increase parsing time. Use --ignore for that

- **Var** either false or an array of packages
- See [phpDocumentor\\_IntermediateParser::\\$packageoutput](#), [Converter::\\$package\\_output](#)

### **parserPage::\$parserVersion**

*string = PHPDOCUMENTOR\_VER [line 109]*

### **parserPage::\$path**

*string = " [line 119]*

- **Var** full path this page represents

### **parserPage::\$source**

*array = array() [line 124]*

## **Tokenized source code of the file**

### **parserPage::\$sourceLocation**

*string = " [line 81]*

## **relative source location**

### **parserPage::\$subpackage**

*string = " [line 105]*

### **parserPage::\$type**

*string = 'page' [line 66]*

## **Type is used by many functions to skip the hassle of if**

1        [phpDocumentor\\_get\\_class\(\\$blah\) == 'parserBlah'](#)

*Constructor void function parserPage::parserPage() [line 139]*  
**sets package to default package**

- **Global Variable Used** string [\\$phpDocumentor\\_DefaultPackageName](#): default package name

*string/bool function parserPage::getFile() [line 188]*  
**gets the file name**

*string function parserPage::getName() [line 273]*  
**gets the name**

*array function parserPage::getPackageOutput() [line 247]*  
**gets the package output array**

- See [phpDocumentor\\_IntermediateParser::\\$packageoutput](#)

*bool function parserPage::getParseData() [line 333]*  
**Not implemented in this version**

*string function parserPage::getPath() [line 219]*  
**gets the path**

*string function parserPage::getSourceLocation(\$c, [\$pearize = false]) [line 307]*  
**Function Parameters:**

- [Converter \\$c](#) the output converter
- [bool \\$pearize](#) if this parameter is true,  
to the subdirectory of pear

it will truncate the source location

## **gets the source location**

- **TODO** determine if the str\_replace in the 'pear/' ELSE branch should be removed (see Documentation/tests/bug1574043.php). It does NOT exist in the similar function parserClass->getSourceLocation() in ParserElements.inc.

*string* function parserPage::getType() [line 150]

## **gets the tag type**

*void* function parserPage::setFile(\$file) [line 177]

### **Function Parameters:**

- *string* **\$file** the file name

## **Sets the name to display in documentation (can be an alias set with @name)**

*void* function parserPage::setName(\$name) [line 261]

### **Function Parameters:**

- *string* **\$name** phptoc-safe name (only \_, numbers and letters) set by Parser::parse()

## **sets the name**

- See [Parser::parse\(\)](#)

*void* function parserPage::setPackageOutput(\$packages) [line 236]

### **Function Parameters:**

- *array* **\$packages** array of packages to display in documentation (package1, package2,...)

## **loads the package output array**

- See [phpDocumentor\\_IntermediateParser::\\$packageoutput](#)

*void* function parserPage::setPath(\$path) [/line 203]

**Function Parameters:**

- *string* **\$path** full path to file

## **sets the path to the file**

*void* function parserPage::setSource(\$source) [/line 165]

**Function Parameters:**

- *string|array* **\$source** the token array/string

## **Sets the source code of the file for highlighting.**

PHP 4.3.0+ passes an array of tokenizer tokens by line number. PHP 4.2.3- passes a string to be passed to [http://www.php.net/highlight\\_string](http://www.php.net/highlight_string)

*void* function parserPage::setSourceLocation(\$source) [/line 288]

**Function Parameters:**

- *string* **\$source** path of this file relative to program root

## **sets the source location**

# Class parserStringWithInlineTags

[line 776]

Used to represent strings that contain inline tags, so that they can be properly parsed at link time

- **Package** phpDocumentor
- **Sub-Package** ParserData
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.0rc1
- **License** [LGPL](#)

## parserStringWithInlineTags::\$type

*string = '\_string' [line 784]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'...`  
always '\_string'

## parserStringWithInlineTags::\$value

*array = array() [line 795]*

### array of strings and [parserInlineTag](#)

Format: array(string1,string2,parserInlineTag1,string3,parserInlineTag2,...)

*void function parserStringWithInlineTags::add(\$stringOrInlineTag) [line 804]*

**Function Parameters:**

- *mixed \$stringOrInlineTag* either a string or a [parserInlineTag](#)

## equivalent to the . operator (\$a = \$b . \$c)

*string function parserStringWithInlineTags::Convert(&\$converter, [\$postprocess = true], [\$trim = true]) [line 924]*

**Function Parameters:**

- [Converter](#) &\$converter the output converter
- bool \$postprocess true if one needs to postprocess
- bool \$trim false if the output should not be trimmed

**Use to convert the string to a real string with all inline tags parsed and linked**

- See [Converter::returnSee\(\)](#)
- TODO CS cleanup - rename to convert for camelCase rule

*string* function parserStringWithInlineTags::getString([*\$trim* = true]) [line 896]

**Function Parameters:**

- bool \$trim whether to trim the string

**return the string unconverted (all inline tags are taken out - this should only be used in pre-parsing to see if any other text is in the string)**

- Uses [parserInlineTag::getString\(\)](#) - removes inline tag length, as it is indeterminate until conversion.

*bool* function parserStringWithInlineTags::hasInlineTag() [line 835]

**Determine whether the string contains any inline tags**

- Tutorial [phpDocumentor Inline tags](#)

*void* function parserStringWithInlineTags::setSource(\$source) [line 853]

**Function Parameters:**

- *string/array* **\$source** source code ready to be highlighted

**Pass source code to any {@source} tags contained within the string for later conversion.**

*integer* function parserStringWithInlineTags::trimmedStrlen() [line 868]  
**equivalent to trim(strlen(\$string))**

# ParserDocBlock.inc

## DocBlock Parser Classes

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** ParserDocBlock
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: ParserDocBlock.inc 287886 2009-08-30 05:31:05Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **See** [Parser](#), [WordParser](#)
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.0rc1
- **License** [LGPL](#)

## Class parserDesc

[line 47]

represents a short or long description in a DocBlock [parserDocBlock](#)

- **Package** phpDocumentor
- **Sub-Package** ParserDocBlock
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserDocBlock.inc 287886 2009-08-30 05:31:05Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

### **parserDesc::\$type**

*string = '\_desc' [line 54]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`  
always '\_desc'

*void function parserDesc::add(\$stringOrClass) [line 60]*

#### **Function Parameters:**

- *mixed \$stringOrClass* like [parserStringWithInlineTags::add\(\)](#), this can be a string or parserInlineTag, but it can also be a `parserStringWithInlineTags`, and the contents will be merged

*boolean function parserDesc::hasInheritDoc() [line 81]*

*boolean function parserDesc::hasSource() [line 92]*

*void function parserDesc::replaceInheritDoc(\$desc) [line 104]*

#### **Function Parameters:**

- [parserDesc \\$desc](#) parent parserDesc, used to retrieve the description

**replaces {@inheritDoc} with the contents of the parent DocBlock**

## Class parserDocBlock

*[line 129]*

Represents a docblock and its components [\\$desc](#), [\\$sdesc](#), [\\$tags](#), and also [\\$params](#) for functions

- **Package** phpDocumentor
- **Sub-Package** ParserDocBlock
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserDocBlock.inc 287886 2009-08-30 05:31:05Z ashnazg \$
- **Since** 1.0rc1

### **parserDocBlock::\$category**

*string = [line 219]*

### **parserDocBlock::\$desc**

*parserDesc = false [line 134]*

### **parserDocBlock::\$endlinenumber**

*false|integer = false [line 158]*

**Line number in the source on which this docblock ends**

- **Since** 1.2

### **parserDocBlock::\$explicitcategory**

*boolean = false [line 217]*

**fix for bug 708559**

- **Used by** [parserDocBlock::setExplicitCategory\(\)](#)

### **parserDocBlock::\$explicitpackage**

*boolean = false* [line 212]

## fix for bug 591396

- Used by [parserDocBlock::setExplicitPackage\(\)](#)

### parserDocBlock::\$funcglobals

*array = array()* [line 184]

#### array of global variable data.

Format: array(index of global variable in @global tag list -OR- global variable name  
=> array(datatype,parserStringWithInlineTags),...)

### parserDocBlock::\$hasaccess

*boolean = false* [line 227]

#### whether this DocBlock has an @access tag

### parserDocBlock::\$hasname

*boolean = false* [line 231]

#### whether this DocBlock has a @name tag

### parserDocBlock::\$linenumber

*false/integer = false* [line 152]

#### Line number in the source on which this docblock begins

- Since 1.2

### parserDocBlock::\$package

*string = 'default'* [line 221]

**parserDocBlock::\$packagedescrip**

*string = " [line 237]*

**description of package parsed from @package tag**

Unused in this version

**parserDocBlock::\$params**

*array = array() [line 176]*

**array of param data.**

Format: array(index of param in function parameter list -OR- parameter name => parserStringWithInlineTags,...)

**parserDocBlock::\$processed\_desc**

*array = false [line 138]*

- **Var** array of [parserDescs](#)

**parserDocBlock::\$processed\_sdesc**

*array = false [line 142]*

- **Var** array of [parserDescs](#)

**parserDocBlock::\$properties**

*mixed = array() [line 197]*

**array of [parserPropertyTag](#) [parserPropertyReadTag](#) [parserPropertyWriteTag](#) [parserMethodTag](#) magic tags**

**parserDocBlock::\$return**

*mixed = false [line 202]*

**This is either a [parserReturnTag](#) or false if no return tag is present**

**parserDocBlock::\$sdesc**

*parserDesc = false [line 146]*

**parserDocBlock::\$statics**

*array = array() [line 193]*

**array of static variable data.**

Format: array(index of static variable in @global tag list -OR- static variable name  
=> [parserStaticvarTag](#),...)

**parserDocBlock::\$subpackage**

*string = " [line 223]*

**parserDocBlock::\$subpackagedescrip**

*string = " [line 243]*

**description of subpackage parsed from @package tag**

Unused in this version

**parserDocBlock::\$tags**

*array = array() [line 163]*

**array of [parserTags](#)**

**parserDocBlock::\$unknown\_tags**

*array = array() [line 168]*

**array of unrecognized [parserTags](#)**

**parserDocBlock::\$var**

*mixed = false [line 207]*

**This is either a [parserVarTag](#) or false if no var tag is present**

Constructor void function parserDocBlock::parserDocBlock() [line 256]

**sets package to default**

- **Global Variable Used** string [\\$phpDocumentor\\_DefaultPackageName](#): default package name

*void* function parserDocBlock::addAccess(\$value) [line 916]

**Function Parameters:**

- *string* **\$value** should be either public or private

**add an @access tag to the [tags](#) array**

*void* function parserDocBlock::addExample(\$value, \$path) [line 641]

**Function Parameters:**

- *string* **\$value** contents of the tag
- *string* **\$path** path to the file containing this tag

**adds an @example tag**

*void* function parserDocBlock::addFileSource(\$path, \$source) [line 943]

**Function Parameters:**

- *string* **\$path** full path to the file
- *array* **\$source** tokenized source code, ordered by line number

**Adds a new @filesource tag to the DocBlock**

- **Tutorial** [@filesource](#)

*void* function parserDocBlock::addFuncGlobal(\$type, \$value) [line 857]

**Function Parameters:**

- *string* **\$type** global type
- *string* **\$value** description of how the global is used in the function

**adds a function declaration of @global to [\\$funcglobals array](#)**

*void* function parserDocBlock::addKeyword(\$keyword, \$value) [line 609]

**Function Parameters:**

- *string* **\$keyword** tag name
- [parserStringWithInlineTags](#) **\$value** the contents of the tag

- **Global Variable Used** array [\\$\\_phpDocumentor\\_setting](#): used to determine whether to add the @internal tag or not

*void* function parserDocBlock::addLink(\$link) [line 953]

**Function Parameters:**

- *string* **\$link**

**creates a [parserLinkTag](#) and adds it to the [\\$tags array](#)**

*void* function parserDocBlock::addName(\$value) [line 825]

**Function Parameters:**

- *string* **\$value** new name of element

**Adds a @name tag to the tag list**

*void* function parserDocBlock::addPackage(\$keyword, \$value) [line 728]

**Function Parameters:**

- *string* **\$keyword** tag name (either package or subpackage)

- *mixed \$value* either a string or a `parserStringWithInlineTags`. Strips all inline tags and use the text as the package

`void function parserDocBlock::addParam($paramVar, $paramType, $value) [line 520]`

**Function Parameters:**

- *string \$paramVar* if empty, param is indexed in the order received and set using `changeParam()`
- `parserStringWithInlineTags $value`
- `$paramType`

`void function parserDocBlock::addProperty($tagName, $propertyName, $propertyType, $value) [line 1033]`

**Function Parameters:**

- `$tagName`
- `$propertyName`
- `$propertyType`
- `$value`

### Adds a @property(-read or -write) or @method magic tag to the DocBlock

`void function parserDocBlock::addReturn($returnType, $value) [line 976]`

**Function Parameters:**

- *string \$returnType* the one-word name of the return type (mixed should be used if more than one type)
- `parserStringWithInlineTags $value`

### creates a `parserReturnTag` and adds it to the `$tags` array

`void function parserDocBlock::addSee($keyword, $value) [line 964]`

**Function Parameters:**

- *string \$value*
- *string \$keyword* either see or uses

### creates a `parserLinkTag` and adds it to the `$tags` array

*void function parserDocBlock::addStaticVar(\$staticvar, \$type, \$descrip) [line 844]*

**Function Parameters:**

- *string \$staticvar* if empty, staticvar is indexed in the order received and set using [changeStatic\(\)](#)
- *string \$type* data type
- [\*parserStringWithInlineTags\*](#) *\$descrip*

*void function parserDocBlock::addTag(\$tag) [line 587]*

**Function Parameters:**

- [\*parserTag\*](#) *\$tag* tag

## Used to insert DocBlock Template tags into a docblock

- **Global Variable Used** array [\*\\$phpDocumentor\\_setting\*](#): used to determine whether to add ignored tags, or not

*void function parserDocBlock::addUnknownTag(\$keyword, \$value) [line 651]*

**Function Parameters:**

- *string \$keyword* tag name
- *string \$value* tag value

## adds an unknown tag to the [\*\\$unknown\\_tags\*](#) array for use by custom converters

*void function parserDocBlock::addUses(\$seeel, \$description) [line 1024]*

**Function Parameters:**

- *string \$seeel* @see-style text, used for [\*Converter::getLink\(\)\*](#)
- [\*parserStringWithInlineTags\*](#) *\$description* description of how the used element is used

## Add a @uses tag to the DocBlock

- **Tutorial** [@uses](#)

*void* function parserDocBlock::addVar(\$varType, \$value) [line 993]

**Function Parameters:**

- *string* **\$varType** the one-word name of the variable type (mixed should be used if more than one type)
- [parserStringWithInlineTags](#) **\$value**

creates a [parserVarTag](#) and adds it to the [\\$tags](#) array

*void* function parserDocBlock::canSource() [line 320]

**Tells the DocBlock it can have a @filesource tag**

Only page-level DocBlocks may have a @filesource tag

*void* function parserDocBlock::cantSource() [line 330]

**Tells the DocBlock it can't have a @filesource tag**

Only page-level DocBlocks may have a @filesource tag

*void* function parserDocBlock::changeGlobal(\$index, \$name) [line 866]

**Function Parameters:**

- *integer* **\$index** index of parameter in the [\\$funcglobals](#) array
- *string* **\$name** name of the parameter to set in the \$funcglobals array

*void* function parserDocBlock::changeParam(\$index, \$name, \$type) [line 537]

**Function Parameters:**

- *integer* **\$index** index of parameter in the [\\$params](#) array
- *string* **\$name** name of the parameter to set in the \$params array
- *string|null* **\$type** type of the parameter

*void* function parserDocBlock::changeStatic(\$index, \$name) [line 876]

**Function Parameters:**

- *integer* **\$index** index of parameter in the [\\$statics](#) array
- *string* **\$name** name of the parameter to set in the \$statics array

*string* function parserDocBlock::getDesc(&\$converter) [line 498]

**Function Parameters:**

- [Converter](#) &**\$converter** takes [\\$desc](#) and converts it to a string and returns it if present, otherwise returns "

*integer* function parserDocBlock::getEndLineNumber() [line 294]

**Retrieve ending line number**

*boolean* function parserDocBlock::getExplicitCategory() [line 719]

**If the DocBlock has a @category tag, then this returns true**

*boolean* function parserDocBlock::getExplicitPackage() [line 698]

**If the DocBlock has a @package tag, then this returns true**

*mixed* function parserDocBlock::getKeyword(\$keyword) [line 1076]

**Function Parameters:**

- *string* **\$keyword**

*integer* function parserDocBlock::getLineNumber() [line 276]

**Retrieve starting line number**

*string* function parserDocBlock::getSDesc(&\$converter) [line 476]

**Function Parameters:**

- [Converter](#) &**\$converter** takes [\\$sdesc](#) and converts it to a string and returns it if present, otherwise returns "

*string* function parserDocBlock::getType() [line 1141]

*boolean* function parserDocBlock::hasInheritDoc() [line 453]

**Wrapper for [parserDesc::hasInheritDoc\(\)](#)**

array function parserDocBlock::listParams() [line 1091]  
array function parserDocBlock::listProperties() [line 1109]  
void function parserDocBlock::listTags(0) [line 1125]

**Function Parameters:**

- [Converter 0](#)

void function parserDocBlock::overridePackage(\$category, \$package, \$subpackage, \$elname, \$type) [line 666]

**Function Parameters:**

- *string \$category*
- *string \$package*
- *string \$subpackage*
- *string \$elname* element name
- *string \$type* element type (include, define, var, method, global, function, const)

**set the element's package to the passed values. Used in  
[phpDocumentor IntermediateParser](#) to align package of**  
elements inside a class or procedural page to the package of the class/procedural  
page

void function parserDocBlock::postProcess() [line 304]

**Parse out any html tags from doc comments, and make them into  
abstract structures**

- [Uses parserDescParser::parse\(\)](#)

void function parserDocBlock::replaceInheritDoc(\$desc) [line 465]

**Function Parameters:**

- [parserDesc \\$desc](#)

**Wrapper for[parserDesc::replaceInheritDoc\(\)](#)**

Also replaces {@inheritDoc} in the [\\$processed\\_desc](#)

*void function parserDocBlock::resetParams() [line 528]  
void function parserDocBlock::setDesc(\$desc) [line 438]*

**Function Parameters:**

- [parserDesc|parserDocBlock](#) **\$desc** sets [\\$desc](#)

*void function parserDocBlock::setEndLineNumber(\$number) [line 285]*

**Function Parameters:**

- *integer \$number*

**Sets the ending line number for the DocBlock**

*void function parserDocBlock::setExplicitCategory() [line 710]*

**Used if this docblock has a @category tag.**

phpDocumentor will guess category for DocBlocks that don't have a @category tag

- [Uses parserDocBlock::\\$explicitcategory](#)

*void function parserDocBlock::setExplicitPackage() [line 689]*

**Used if this docblock has a @package tag.**

phpDocumentor will guess package for DocBlocks that don't have a @package tag

- [Uses parserDocBlock::\\$explicitpackage](#)

*void function parserDocBlock::setLineNumber(\$number) [line 267]*

**Function Parameters:**

- *integer* **\$number**

### Sets the starting line number for the DocBlock

*void* function parserDocBlock::setShortDesc(**\$desc**) [line 396]

**Function Parameters:**

- [parserDesc](#)|[parserDocBlock](#)|[parserTag](#) **\$desc** sets [\\$sdesc](#)

### Set the short description of the DocBlock

Setting the short description is possible by passing in one of three possible parameters:

- another DocBlock's short description
- another DocBlock, the short description will be extracted
- a Zend Studio-compatible @desc tag

*void* function parserDocBlock::setSource(**\$source**, [**\$class** = false]) [line 426]

**Function Parameters:**

- *string|array* **\$source** tokenized highlight-ready source code
- *false|string* **\$class** name of class if this is a method source

### Passes to [parserStringWithInlineTags::setSource\(\)](#)

After passing, it calls [postProcess\(\)](#) to set up the new source

*void* function parserDocBlock::updateGlobals(**\$funcs**) [line 886]

**Function Parameters:**

- *array* **\$funcs**

replaces nameless global variables in the [\\$funcglobals](#) array with their names

*void* function parserDocBlock::updateModifiers(\$modifiers) [line 355]

**Function Parameters:**

- *array \$modifiers*

*void* function parserDocBlock::updateParams(\$params) [line 552]

**Function Parameters:**

- *array \$params* Format: array(parameter key => name[,1 => default value][,2 => type hint],...)

**replaces nameless parameters in the \$params array with their names**

add @param tags for params in the function with no entry

*void* function parserDocBlock::updateStatics(\$funcs) [line 901]

**Function Parameters:**

- *array \$funcs*

**replaces nameless static variables in the \$statics array with their names**

# ParserElements.inc

**Parser Elements, all classes representing documentable elements**

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2006 Gregory Beaver
- **See** [Parser](#), [WordParser](#)
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.1
- **License** [LGPL](#)

## Class parserClass

[line 682]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

#### parserClass::\$curfile

*string = false [line 725]*

- **Var** same as [parserElement::\\$path](#)

#### parserClass::\$extends

*mixed = false [line 695]*

- **Var** false or contents of extends clause in class declaration

#### parserClass::\$ignore

*boolean = false [line 720]*

**Used to determine whether a class should be ignored or not. Helps maintain integrity of parsing**

- See [Classes::getParentClass\(\)](#)

#### parserClass::\$parent

*mixed = false [line 714]*

**Format: array(file, parent) where parent class is found or false if no parent**

### **parserClass::\$sourceLocation**

*string = " [line 691]*

- See [parserPage::\\$sourceLocation](#)

### **parserClass::\$tutorial**

*tutorialLink|false = false [line 729]*

- **Var** either a link to the tutorial associated with this class, or false

### **parserClass::\$type**

*string = 'class' [line 688]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`

- **Var** always 'class'

### **parserClass::\$\_implements**

*array = array() [line 699]*

- **Var** a list of interfaces this class implements

*void function parserClass::addImplements(\$implements) [line 1379]*

**Function Parameters:**

- *string* **\$implements**

*void* function parserClass::addTutorial(*\$t, &\$c*) [line 755]

**Function Parameters:**

- [\*parserTutorial\*](#) **\$t**
- [\*Converter\*](#) **&\$c**

*array* function parserClass::getChildClassList(&*\$c*) [line 1308]

**Function Parameters:**

- [\*Converter\*](#) **&\$c** this function will not work before the Conversion stage of parsing

**returns a list of all child classes of this class**

- **Used by** [\*XMLDocBookpeardoc2Converter::generateChildClassList\(\)\*](#)
- **Used by** [\*Converter::generateChildClassList\(\)\*](#)

*mixed* function parserClass::getConflicts(&*\$c*) [line 776]

**Function Parameters:**

- [\*Converter\*](#) **&\$c**

**Returns all classes in other packages that have the same name as this class**

- **Used by** [\*Converter::getFormattedConflicts\(\)\*](#)

*array* function parserClass::getConstNames(&*\$c*) [line 1034]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

array function parserClass::getConsts(&\$c) [line 1009]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

boolean function parserClass::getExtends([\$raw = false]) [line 1396]

**Function Parameters:**

- \$raw

- See [parserClass::\\$extends](#)

array function parserClass::getImplements() [line 1387]

array function parserClass::getInheritedConsts(&\$c, [\$override = false], [\$consts = false]) [line 1225]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing
- boolean \$override determines whether overridden vars should be included in the list of inherited vars
- \$consts

array function parserClass::getInheritedMethods(&\$c, [\$override = false]) [line 1051]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing
- boolean \$override determines whether overridden methods should be included in the list of inherited methods

array function parserClass::getInheritedVars(&\$c, [\$override = true], [\$vars = false]) [line 1149]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

- *boolean \$override* determines whether overriden vars should be included in the list of inherited vars
- *\$vars*

*mixed function parserClass::getLink(\$c, [\$text = false], [\$returnobj = false]) [line 789]*

**Function Parameters:**

- *Converter \$c*
- *string \$text* text to display for the link or false for default text
- *\$returnobj*

### quick way to link to this element

*mixed function parserClass::getMethod(&\$c, \$name, [\$inherited = false]) [line 914]*

**Function Parameters:**

- *Converter &\$c* this function will not work before the Conversion stage of parsing
- *string \$name* method name in this class
- *boolean \$inherited* determines whether to search inherited methods as well

*array function parserClass::getMethodNames(&\$c) [line 943]*

**Function Parameters:**

- *Converter &\$c* this function will not work before the Conversion stage of parsing

*array function parserClass::getMethods(&\$c) [line 903]*

**Function Parameters:**

- *Converter &\$c* this function will not work before the Conversion stage of parsing

*array/false function parserClass::getModifiers() [line 736]*

### Get the PHP5+ modifiers for this class

(abstract/final/static/private/protected/public)

*mixed function parserClass::getParent(&\$c) [line 889]*

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

### retrieve object that represents the parent class

array function parserClass::getParentClassTree(&\$c) [line 1289]

#### Function Parameters:

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

- Usedby [CHMdefaultConverter::generateFormattedClassTree\(\)](#)
- Usedby [HTMLframesConverter::generateFormattedClassTree\(\)](#)
- Usedby [HTMLSmartyConverter::generateFormattedClassTree\(\)](#)
- Usedby [XMLDocBookConverter::generateFormattedClassTree\(\)](#)
- Usedby [XMLDocBookpeardoc2Converter::generateFormattedClassTree\(\)](#)

string function parserClass::getSourceLocation(\$c, [\$pearize = false]) [line 1337]

#### Function Parameters:

- [Converter](#) \$c
- boolean \$pearize

- See [parserClass::\\$sourceLocation](#)

parserTutorial function parserClass::getTutorial() [line 766]

### Get the associated tutorial for this class, if any

- Tutorial [phpDocumentor Tutorials](#)

*mixed* function parserClass::getVar(&\$c, \$name) [line 934]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing
- string \$name var name in this class

*array* function parserClass::getVarNames(&\$c) [line 1018]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

*array* function parserClass::getVars(&\$c) [line 1000]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing

*boolean* function parserClass::hasConst(&\$c, \$name) [line 991]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing
- string \$name class constant name

*boolean* function parserClass::hasMethod(&\$c, \$name, [\$inherited = false]) [line 961]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing
- string \$name method name
- boolean \$inherited determines whether to search inherited methods as well

*boolean* function parserClass::hasVar(&\$c, \$name) [line 981]

**Function Parameters:**

- [Converter](#) &\$c this function will not work before the Conversion stage of parsing
- string \$name var name

*boolean* function parserClass::isInterface() [line 870]

*void* function parserClass::setAccessModifiers(\$modifiers) [/line 879]

**Function Parameters:**

- *array* \$modifiers

**Use this method to set access modifiers for a class**

*void* function parserClass::setExtends(\$extends) [/line 1362]

**Function Parameters:**

- *string* \$extends

- See [parserClass::\\$extends](#)

*void* function parserClass::setInterface() [/line 862]

**Use this method to set the type of class to be an interface**

*void* function parserClass::setModifiers(\$m) [/line 746]

**Function Parameters:**

- *string* \$m

**Set the PHP5+ modifiers for this class**

(abstract/final/static/private/protected/public)

*void* function parserClass::setParent(\$p, \$f, &\$c) [/line 805]

**Function Parameters:**

- *string* \$p parent class name
- *string* \$f parent class file
- [Classes](#) &\$c [Classes](#) object currently calling setParent

- See [Classes::setClassParent\(\)](#)

*void* function parserClass::setParentNoClass(\$par) [line 854]

**Function Parameters:**

- *string* **\$par** parent class name (used by [Classes::setClassParent\(\)](#) if parent class not found

*void* function parserClass::setSourceLocation(\$sl) [line 1326]

**Function Parameters:**

- *string* **\$sl**
- See [parserClass::\\$sourceLocation](#)

## Class parserConst [line 1578]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.2.4

### parserConst::\$class

*string* = " [line 1586]

- **Var** class that contains this var

### **parserConst::\$type**

*string = 'const' [line 1584]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`

- **Var** always 'const'

Constructor `void` function `parserConst::parserConst($class)` [line 1591]

**Function Parameters:**

- *string \$class*

*string* function `parserConst::getClass()` [line 1600]

**Retrieve the class name**

*mixed* function `parserConst::getLink($c, [$text = false], [$returnobj = false])` [line 1611]

**Function Parameters:**

- *Converter \$c*
- *string \$text* text to display for the link or false for default text
- *\$returnobj*

**quick way to link to this element**

# Class parserDefine

[line 1983]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

## parserDefine::\$type

*string = 'define' [line 1989]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`

- **Var** always 'define'

*mixed function parserDefine::getConflicts(&\$c) [line 2011]*

**Function Parameters:**

- Converter &\$c

**Returns all defines in other packages that have the same name as this define**

- **Used by** Converter::getFormattedConflicts()

*mixed function parserDefine::getLink(\$c, [\$text = false], [\$returnobj = false]) [line 1997]*

**Function Parameters:**

- Converter \$c
- **string** \$text text to display for the link or false for default text

- **\$returnobj**

**quick way to link to this element**

## Class parserElement

*[line 49]*

**all elements except [parserPackagePage](#) descend from this abstract class**

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Abstract Element**
- **Since** 1.0rc1

### parserElement::\$conflicts

*mixed = false [line 63]*

- **Var** either false or an array of paths to files with conflicts

### parserElement::\$docblock

*mixed = false [line 54]*

- **Var** either false or a [parserDocBlock](#)

### parserElement::\$endlinenumber

*false|integer = 0* [line 82]

### **line number on file where this element stops**

- Since 1.2

#### **parserElement::\$file**

*string = "* [line 69]

### **location of this element (filename)**

#### **parserElement::\$linenumber**

*false|integer = false* [line 89]

### **Line number in the source on which this element appears**

- Since 1.2

#### **parserElement::\$name**

*mixed =* [line 58]

### **name of this element, or include type if element parserInclude**

#### **parserElement::\$path**

*string = "* [line 75]

### **full path location of this element (filename)**

*integer* function parserElement::getEndLineNumber() [line 136]

*string* function parserElement::getFile() [line 178]

*integer* function parserElement::getLineNumber() [line 128]

*string* function parserElement::getName() [line 169]

*string* function parserElement::getPackage() [line 142]

*string* function parserElement::getPath() [line 187]

*void* function parserElement::setDocBlock(\$docblock) [*line 94*]

**Function Parameters:**

- *parserDocBlock* \$docblock

*void* function parserElement::setEndLineNumber(\$l) [*line 120*]

**Function Parameters:**

- *integer* \$l

## Sets the ending line number of elements

*void* function parserElement::setFile(\$file) [*line 151*]

**Function Parameters:**

- *string* \$file

*void* function parserElement::setLineNumber(\$number) [*line 111*]

**Function Parameters:**

- *integer* \$number

## Set starting line number

*void* function parserElement::setName(\$name) [*line 102*]

**Function Parameters:**

- *string* \$name

*void* function parserElement::setPath(\$file) [*line 157*]

**Function Parameters:**

- *string* \$file

# Class parserFunction

[line 321]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

## parserFunction::\$globals

array = false [line 351]

### global declarations parsed from function definition

- **Var** Format: array(globalname1, globalname2,...)

## parserFunction::\$params

array = false [line 335]

### parameters parsed from function definition.

param name may be null, in which case, updateParams() must be called from the Converter

- **Var** Format: array(param name => default value parsed from function definition)
- **See** updateParams()

## parserFunction::\$returnsreference

*boolean = false [line 345]*

### **Function returns a reference to an element, instead of a value**

set to true if function is declared as:

1      **function & func(...)**

### **parserFunction::\$source**

*mixed = " [line 358]*

### **parserFunction::\$statics**

*array = false [line 356]*

### **static variable declarations parsed from function definition**

- **Var** Format: array(array('name' => staticvar1,'val' => " or default val of staticvar1),...)

### **parserFunction::\$type**

*string = 'function' [line 327]*

Type is used by many functions to skip the hassle of if  
**phpDocumentor\_get\_class(\$blah) == 'parserBlah'**

- **Var** always 'function'

*void function parserFunction::addGlobals(\$globals) [line 433]*

#### **Function Parameters:**

- **array \$globals** Format: array(globalname1, globalname2,...)

### **Add all "global \$var, \$var2" declarations to this function**

*void* function parserFunction::addParam(\$name, \$value, [\$has\_default = true], [\$typehint = null]) [line 366]

**Function Parameters:**

- *string* **\$name**
- *string* **\$value** default value parsed from function definition
- *boolean* **\$has\_default** indicates whether this parameter has a default value
- *null|string* **\$typehint** class type hint

*void* function parserFunction::addSource(\$source) [line 379]

**Function Parameters:**

- *string/array* **\$source**

## Set the source code. Always array in PHP 4.3.0+

*void* function parserFunction::addStatics(\$static, \$vals) [line 443]

**Function Parameters:**

- *array* **\$static** Format: array(varname1, varname2,...)
- *array* **\$vals** Format: array(default val of var 1, default val of var 2,...) if var 1 has no default, array(default val of var 2,...)

## Add all "static \$var, \$var2 = 6" declarations to this function

*mixed* function parserFunction::getConflicts(&\$c) [line 422]

**Function Parameters:**

- Converter **&\$c**

## Returns all functions in other packages that have the same name as this function

- Used by Converter::getFormattedConflicts()

*string* function parserFunction::getFunctionCall() [line 580]

### Get a human-friendly description of the function call

takes declaration like:

<sup>1</sup>

and returns: string &func( \$param1, [\$param2 = 6], [\$param3 = array('20',9 => "heroo")] )

*array* function parserFunction::getIntricateFunctionCall(\$converter, \$paramtags) [line 621]

#### Function Parameters:

- **\$converter**
- **\$paramtags**

**Like getFunctionCall(), but has no English or pre-determined formatting.**

Much more flexible.

- See [parserFunction::getFunctionCall\(\)](#)

*mixed* function parserFunction::getLink(\$c, [\$text = false], [\$returnobj = false]) [line 408]

#### Function Parameters:

- ***Converter* \$c**
- **string \$text** text to display for the link or false for default text
- **\$returnobj**

**quick way to link to this element**

*string* function parserFunction::getParam(\$name) [line 464]

#### Function Parameters:

- **string \$name**

*boolean* function parserFunction::getReturnsReference() [line 561]

*string|array* function parserFunction::getSource() [line 397]

*boolean* function parserFunction::hasSource() [line 388]

**Determine whether the source code has been requested via {@source}**

```
array function parserFunction::listGlobals() [line 515]
array function parserFunction::listParams() [line 480]
array function parserFunction::listStatics() [line 534]
void function parserFunction::setReturnsReference() [line 553]
    sets $returnsreference to true
```

## Class parserGlobal [line 217]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.1

### parserGlobal::\$datatype

*string = 'mixed' [line 229]*

#### Name of the global's data type

### parserGlobal::\$type

*string = 'global' [line 223]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`

- **Var** always 'global'

*mixed function parserGlobal::getConflicts(&\$c) [line 251]*

#### Function Parameters:

- [Converter](#) &\$c

**Returns all global variables in other packages that have the same name as this global variable**

- Used by [Converter::getFormattedConflicts\(\)](#)

*string* function parserGlobal::getDataType(&\$converter) [line 277]

**Function Parameters:**

- [Converter](#) &\$converter

**Retrieve converter-specific representation of the data type**

If the data type is a documented class name, then this function will return a Converter-specific link to that class's documentation, so users can click/browse to the documentation directly from the global variable declaration

*mixed* function parserGlobal::getLink(&\$c, [\$text = false], [\$returnobj = false]) [line 237]

**Function Parameters:**

- [Converter](#) &\$c
- *string* \$text text to display for the link or false for default text
- \$returnobj

**quick way to link to this element**

*void* function parserGlobal::setDataType(\$type) [line 262]

**Function Parameters:**

- *string* \$type

**Sets the name of the global variable's type**

## Class parserInclude

[line 201]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

### parserInclude::\$type

*string = 'include' [line 207]*

Type is used by many functions to skip the hassle of if  
phpDocumentor\_get\_class(\$blah) == 'parserBlah'

- **Var** always 'include'

## Class parserMethod

[line 1628]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

### **parserMethod::\$class**

*string = " [line 1640]*

- **Var** class that contains this method

### **parserMethod::\$isConstructor**

*boolean = false [line 1636]*

- **Var** whether this method is a constructor

### **parserMethod::\$isDestructor**

*boolean = false [line 1638]*

- **Var** whether this method is a destructor by PEAR standards

### **parserMethod::\$type**

*string = 'method' [line 1634]*

**Type is used by many functions to skip the hassle of if  
phpDocumentor\_get\_class(\$blah) == 'parserBlah'**

- **Var** always 'method'

### **parserMethod::\$\_modifiers**

*mixed* = array() [line 1641]

Constructor void function parserMethod::parserMethod(\$class) [line 1646]

**Function Parameters:**

- *string \$class*

*void* function parserMethod::addParam(\$name, \$value, [\$has\_default = true], [\$typehint = null]) [line 1657]

**Function Parameters:**

- *string \$name*
- *string \$value* default value parsed from function definition
- *boolean \$has\_default* indicates whether this parameter has a default value
- *null|string \$typehint* class type hint

*string* function parserMethod::getClass() [line 1690]

**Return name of the class that contains this method**

*string* function parserMethod::getFunctionCall() [line 1671]

**adds "constructor" to start of function call if      \$isConstructois true**

- See [parserFunction::getFunctionCall\(\)](#)

*mixed* function parserMethod::getImplements(&\$c) [line 1805]

**Function Parameters:**

- [Converter &\\$c](#)

- Used by [Converter::getFormattedMethodImplements\(\)](#)

*void* function parserMethod::getIntricateFunctionCall(\$converter, \$paramtags) [line 1678]

**Function Parameters:**

- **\$converter**
- **\$paramtags**

*mixed* function parserMethod::getLink(\$c, [\$text = false], [\$returnobj = false]) [line 1915]

**Function Parameters:**

- [Converter](#) **\$c**
- **string \$text** text to display for the link or false for default text
- **\$returnobj**

### quick way to link to this element

*string* function parserMethod::getModifiers() [line 1699]

**Return name of the class that contains this method**

*mixed* function parserMethod::getOverrides(&\$c) [line 1718]

**Function Parameters:**

- [Converter](#) **&\$c**

*array* function parserMethod::getOverridingMethods(&\$c) [line 1944]

**Function Parameters:**

- [Converter](#) **&\$c**
  - Used by [Converter::getFormattedDescMethods\(\)](#)

*array* function parserMethod::getOverridingMethodsForClass(&\$c, &\$class) [line 1956]

**Function Parameters:**

- [Converter](#) **&\$c**
- [parserClass](#) **&\$class**

*void* function parserMethod::setConstructor() [*line 1927*]

**Use this method to tell the parser that this method is the class constructor**

*void* function parserMethod::setDestructor() [*line 1935*]

**Use this method to tell the parser that this method is the class constructor**

*string* function parserMethod::setModifiers(\$m) [*line 1708*]

**Function Parameters:**

- \$m

**Return name of the class that contains this method**

## Class parserPackagePage [*line 2027*]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net) >
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

**parserPackagePage::\$package**

*string* = 'default' [*line 2035*]

**parserPackagePage::\$type**

*string* = 'packagepage' [*line 2033*]

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`

- **Var** always 'packagepage'

Constructor `void function parserPackagePage::parserPackagePage($package) [line 2040]`

**Function Parameters:**

- `string $package`

`void function parserPackagePage::Convert(&$c) [line 2048]`

**Function Parameters:**

- `Converter &$c`

## Class `parserTutorial` [line 2059]

- **Package** `phpDocumentor`
- **Sub-Package** `ParserElements`
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.2
- **Usedby** [XMLPackagePageParser::parse\(\)](#) - using Publisher::PublishEvent(), a new tutorial  
is created from the file parsed, and passed to the  
Intermediate Parser

### `parserTutorial::$children`

`array = false [line 2124]`

**links to the child tutorials, or false if none**

### `parserTutorial::$ini`

`array = false [line 2100]`

### **output from tutorialname.ext.ini**

an array generated by [phpDocumentor\\_parse\\_ini\\_file\(\)](#) containing an index 'Linked Tutorials' with an array of tutorial names in the order they should appear. This is used to generate a linked list of tutorials like [phpDocumentor tags](#)

### **parserTutorial::\$linked\_element**

*mixed = [line 2078]*

### **The documentable element this tutorial is linked to**

Can be a parserData, parserClass, or nothing for package/subpackage docs

### **parserTutorial::\$name**

*string = [line 2088]*

### **filename minus extension of this tutorial (used for @tutorial tag)**

### **parserTutorial::\$next**

*tutorialLink = false [line 2105]*

### **link to the next tutorial in a document series, or false if none**

- Used by [parserTutorial::setNext\(\)](#) - creates a link to the documentation for the next tutorial

### **parserTutorial::\$package**

*string = 'default' [line 2067]*

### **parserTutorial::\$parent**

*tutorialLink = false [line 2119]*

### **link to the parent tutorial in a document series, or false if none**

This is used to generate an "Up" or "Home" link like the php manual. The parent is defined as a tutorial that has a parenttutorialname.ext.ini file and is not contained by any other tutorial's tutorialname.ext.ini

- Used by [parserTutorial::setParent\(\)](#) - creates a link to the documentation for the parent tutorial

**parserTutorial::\$path**

*string = [line 2083]*

**path to the tutorial page**

**parserTutorial::\$prev**

*tutorialLink = false [line 2110]*

**link to the previous tutorial in a document series, or false if none**

- Used by [parserTutorial::setPrev\(\)](#) - creates a link to the documentation for the previous tutorial

**parserTutorial::\$tutorial\_type**

*string = [line 2072]*

**Either cls, pkg, or proc**

**parserTutorial::\$type**

*string = 'tutorial' [line 2065]*

**Type is used by many functions to skip the hassle of if  
phpDocumentor\_get\_class(\$blah) == 'parserBlah'**

- Var always 'tutorial'

**parserTutorial::\$\_xml**

*boolean = true [line 2090]*

Constructor `void function parserTutorial::parserTutorial($data, $info) [line 2136]`

**Function Parameters:**

- `parserXMLDocBookTag` **\$data** top-level tag (`<refentry>` for 1.2.0)
- `information` **\$info** about the tutorial file. Format:

```
array('tutename' => tutorial name,  
      'path' => relative path of tutorial to tutorials/ directory  
      'ini' => contents of the tutorial .ini file, if any)
```

`void function parserTutorial::Convert(&$c, [$postprocess = true]) [line 2174]`

**Function Parameters:**

- `Converter` **&\$c**
- `boolean` **\$postprocess** determines whether character data is postprocessed to be Converter-friendly or not.

`string|tutorialLink` function `parserTutorial::getLink(&$c, [$pure = false], [$section = "]) [line 2278]`

**Function Parameters:**

- `Converter` **&\$c**
- `boolean` **\$pure** if true, returns a `tutorialLink` instead of a string
- `string` **\$section** section name to link to

## Get a link to this tutorial, or to any subsection of this tutorial

`void function parserTutorial::getNext(&$c) [line 2243]`

**Function Parameters:**

- `Converter` **&\$c**

## Retrieve converter-specific link to the next tutorial's documentation

`void function parserTutorial::getParent(&$c) [line 2221]`

**Function Parameters:**

- [Converter](#) &\$c

### Retrieve converter-specific link to the parent tutorial's documentation

*void function parserTutorial::getPrev(&\$c) [line 2265]*

**Function Parameters:**

- [Converter](#) &\$c

### Retrieve converter-specific link to the previous tutorial's documentation

*void function parserTutorial::getTitle(&\$c, [\$subsection = "]) [line 2154]*

**Function Parameters:**

- [Converter](#) &\$c
- *string \$subsection* which subsection to retrieve the title from, if any

### Retrieve the title of the tutorial, or of any subsection

- **Uses** parserXMLDocBookTag::getSubSection() - retrieve the subsection to get a title from

*boolean function parserTutorial::isChildOf(\$parents) [line 2199]*

**Function Parameters:**

- *array \$parents* array of parserTutorials that have child tutorials

### Determine if this parserTutorial object is a child of another

WARNING: This method can enter an infinite loop when run on PHP v5.2.1... see [PHP Bug #40608](#) and [PEAR Bug #10289](#)

*void function parserTutorial::setNext(\$next, &\$c) [line 2232]*

**Function Parameters:**

- parserTutorial **\$next**
- Converter **&\$c**

- **Uses parserTutorial::\$next** - creates a link to the documentation for the next tutorial

*void function parserTutorial::setParent(\$parent, &\$c) [line 2184]*

**Function Parameters:**

- parserTutorial **\$parent**
- Converter **&\$c**

- **Uses parserTutorial::\$parent** - creates a link to the documentation for the parent tutorial

*void function parserTutorial::setPrev(\$prev, &\$c) [line 2254]*

**Function Parameters:**

- parserTutorial **\$prev**
- Converter **&\$c**

- **Uses parserTutorial::\$prev** - creates a link to the documentation for the previous tutorial

# Class parserVar [line 1415]

- **Package** phpDocumentor
- **Sub-Package** ParserElements
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: ParserElements.inc 248547 2007-12-19 02:16:49Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Since** 1.0rc1

## parserVar::\$class

*string = " [line 1423]*

- **Var** class that contains this var

## parserVar::\$type

*string = 'var' [line 1421]*

Type is used by many functions to skip the hassle of if  
`phpDocumentor_get_class($blah) == 'parserBlah'`

- **Var** always 'var'

## parserVar::\$\_modifiers

*array = [line 1425]*

Constructor `void function parserVar::parserVar($class) [line 1430]`

### **Function Parameters:**

- *string \$class*

*string function parserVar::getClass() [line 1439]*

### **Retrieve the class name**

*mixed* function parserVar::getLink(\$c, [\$text = false], [\$returnobj = false]) [line 1468]

**Function Parameters:**

- [Converter \\$c](#)
- *string* \$text text to display for the link or false for default text
- **\$returnobj**

## quick way to link to this element

*array* function parserVar::getModifiers() [line 1448]

**Return a list of access modifiers (static/private/etc.)**

*mixed* function parserVar::getOverrides(&\$c) [line 1481]

**Function Parameters:**

- [Converter &\\$c](#)

*array* function parserVar::getOverridingVars(&\$c) [line 1540]

**Function Parameters:**

- [Converter &\\$c](#)
  - Used by [Converter::getFormattedDescVars\(\)](#)

*array* function parserVar::getOverridingVarsForClass(&\$c, &\$class) [line 1552]

**Function Parameters:**

- [Converter &\\$c](#)
- [parserClass &\\$class](#)

*string* function parserVar::setModifiers(\$m) [line 1457]

**Function Parameters:**

- \$m

**Return name of the class that contains this method**

# Beautifier.php

## XML/Beautifier.php

Format XML files containing unknown entities (like all of peardoc)

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2004-2006 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Greg Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: Beautifier.php 212211 2006-04-30 22:18:14Z celflog \$
- **Copyright** 2004-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.3.0
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

# Tokenizer.php

## XML/Beautifier.php

Format XML files containing unknown entities (like all of peardoc)

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2004-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Greg Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: Tokenizer.php 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2004-2006 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 1.3.0
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

PHPDOC\_BEAUTIFIER\_CDATA = 100000 [*line 751*]

**do not remove, needed in plain renderer**

PHPDOC\_XMLTOKEN\_EVENT\_ATTRIBUTE = 4 [*line 701*]

**used when a <tag attr="attribute"> is found**

PHPDOC\_XMLTOKEN\_EVENT\_CDATA = 11 [line [736](#)]

**used when a <! is found**

PHPDOC\_XMLTOKEN\_EVENT\_CLOSETAG = 5 [line [706](#)]

**used when a close </tag> is found**

PHPDOC\_XMLTOKEN\_EVENT\_COMMENT = 7 [line [716](#)]

**used when a <!-- comment --> is found**

PHPDOC\_XMLTOKEN\_EVENT\_DEF = 10 [line [731](#)]

**used when a <! is found**

PHPDOC\_XMLTOKEN\_EVENT\_DOUBLEQUOTE = 9 [line [726](#)]

**used when a <!-- comment --> is found**

PHPDOC\_XMLTOKEN\_EVENT\_ENTITY = 6 [line [711](#)]

**used when an &entity; is found**

PHPDOC\_XMLTOKEN\_EVENT\_IN\_CDATA = 13 [line [746](#)]

**used when a <![CDATA[ section is found**

PHPDOC\_XMLTOKEN\_EVENT\_NOEVENTS = 1 [line [686](#)]

**starting state**

PHPDOC\_XMLTOKEN\_EVENT\_OPENTAG = 3 [line [696](#)]

**used when an open <tag> is found**

PHPDOC\_XMLTOKEN\_EVENT\_PI = 2 [line [691](#)]

**used when a processor instruction is found**

PHPDOC\_XMLTOKEN\_EVENT\_SINGLEQUOTE = 8 [line [721](#)]

**used when a <!-- comment --> is found**

PHPDOC\_XMLTOKEN\_EVENT\_XML = 12 [line [741](#)]

**used when a <?xml is found**

require\_once 'XML/Beautifier/Tokenizer.php' [line [44](#)]

## From the XML\_Beautifier package

STATE\_XMLTOKEN\_ATTRIBUTE = 104 [line [703](#)]  
**currently parsing an open <tag>**

STATE\_XMLTOKEN\_CDATA = 111 [line [738](#)]  
**currently parsing a <!**

STATE\_XMLTOKEN\_CLOSETAG = 105 [line [708](#)]  
**currently parsing a close </tag>**

STATE\_XMLTOKEN\_COMMENT = 107 [line [718](#)]  
**currently parsing a <!-- comment -->**

STATE\_XMLTOKEN\_DEF = 110 [line [733](#)]  
**currently parsing a <!**

STATE\_XMLTOKEN\_DOUBLEQUOTE = 109 [line [728](#)]  
**currently parsing a <!-- comment -->**

STATE\_XMLTOKEN\_ENTITY = 106 [line [713](#)]  
**currently parsing an &entity;**

STATE\_XMLTOKEN\_IN\_CDATA = 113 [line [748](#)]  
**currently parsing a <![CDATA[ ]]>**

STATE\_XMLTOKEN\_NOEVENTS = 101 [line [688](#)]  
**currently in starting state**

STATE\_XMLTOKEN\_OPENTAG = 103 [line [698](#)]  
**currently parsing an open <tag>**

STATE\_XMLTOKEN\_PI = 102 [line [693](#)]  
**currently in processor instruction**

STATE\_XMLTOKEN\_SINGLEQUOTE = 108 [line [723](#)]  
**currently parsing a <!-- comment -->**

STATE\_XMLTOKEN\_XML = 112 [line [743](#)]

**currently parsing a <?xml**

# HighlightParser.inc

## Source Code Highlighting

The classes in this file are responsible for the dynamic @example, @filesource and {@source} tags output. Using the phpDocumentor\_HighlightWordParser, the phpDocumentor\_HighlightParser retrieves PHP tokens one by one from the array generated by [phpDocumentorTWordParser](#) source retrieval functions and then highlights them individually.

It accomplishes this highlighting through the assistance of methods in the output Converter passed to its parse() method, and then returns the fully highlighted source as a string

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** [phpDocumentor](#)
- **Sub-Package** Parsers
- **Tutorial** [@example](#), [@filesource](#), [inline {@source}](#)
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: HighlightParser.inc 253641 2008-02-24 02:35:44Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2.0beta3
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)



# Parser.inc

## Base parser for all parsers

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** CVS: \$Id: Parser.inc 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2000-2006 Joshua Eichorn, Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **Since** 0.1
- **License** [LGPL](#)

PARSER\_EVENT\_ACCESS\_MODIFIER = 139 [line 277]

**used when parsing an access modifier**

PARSER\_EVENT\_ARRAY = 115 [line 160]

**used when an array definition is encountered in parsing**

PARSER\_EVENT\_CLASS = 111 [line 138]

**used when a class definition is encountered in parsing**

PARSER\_EVENT\_CLASS\_CONSTANT = 141 [line 287]

**used when a class implements interfaces**

PARSER\_EVENT\_CLASS\_MEMBER = 137 [line 267]

**used by the HighlightParser only, when ->var or ->function() is encountered in a method**

PARSER\_EVENT\_COMMENT = 105 [line 108]

**used when short comment // is encountered in parsing**

PARSER\_EVENT\_COMMENTBLOCK = 104 [line 103]

**used when long comment /x x/ where x is an asterisk is encountered in parsing**

PARSER\_EVENT\_DEFINE = 107 [line 118]

**used when a define statement is encountered in parsing**

PARSER\_EVENT\_DEFINE\_GLOBAL = 128 [line 222]

**used when parsing a global variable declaration**

PARSER\_EVENT\_DEFINE\_PARAMS = 108 [line 123]

**used when a define statement opening parenthesis is encountered in parsing**

PARSER\_EVENT\_DEFINE\_PARAMS\_PARENTHESIS = 120 [line 185]

**used when a define statement's opening parenthesis is encountered in parsing**

PARSER\_EVENT\_DESC = 126 [line 212]

**used when parsing the desc part of a docblock**

PARSER\_EVENT\_DOCBLOCK = 112 [line 145]

**used when a DocBlock is encountered in parsing**

PARSER\_EVENT\_DOCBLOCK\_TEMPLATE = 133 [line 247]

**used when encountering a /\*\*#@+ comment marking a new docblock template**

PARSER\_EVENT\_DOCKEYWORD = 113 [line 150]

**used when a @tag is encountered in DocBlock parsing**

PARSER\_EVENT\_DOCKEYWORD\_EMAIL = 114 [line 155]

**used when a <email@address> is encountered in parsing an @author tag**

PARSER\_EVENT\_END\_DOCBLOCK\_TEMPLATE = 134 [line 252]

**used when encountering a /\*\*#@-\* / comment (no space) marking the end of using a docblock template**

PARSER\_EVENT\_END\_STATEMENT = 121 [line 189]

PARSER\_EVENT\_EOFQUOTE = 122 [line 192]

**used when a <<< is encountered in parsing**

PARSER\_EVENT\_ESCAPE = 900 [line 40]

**used when a backslash is encountered in parsing a string or other escapable entity**

PARSER\_EVENT\_FUNCTION = 117 [line 170]

**used when a function definition is encountered in parsing**

PARSER\_EVENT\_FUNCTION\_PARAMS = 109 [line 128]

**used when a function statement opening parenthesis is encountered in parsing**

PARSER\_EVENT\_FUNCTION\_PARAM\_VAR = 144 [line 302]

**used when a \$param is encountered in a function definition**

PARSER\_EVENT\_FUNC\_GLOBAL = 130 [line 232]

**used when parsing a "global \$var1, \$var2;" declaration in a function**

PARSER\_EVENT\_GLOBAL\_VALUE = 129 [line 227]

**used when parsing the default value in a global variable declaration**

PARSER\_EVENT\_IMPLEMENTS = 140 [line 282]

**used when a class implements interfaces**

PARSER\_EVENT\_INCLUDE = 123 [line 197]

**used when an include/require/include\_once/include\_once statement is encountered in parsing**

PARSER\_EVENT\_INCLUDE\_PARAMS = 124 [line 202]

**used when an opening parenthesis of an include/require/include\_once/include\_once statement is encountered in parsing**

PARSER\_EVENT\_INCLUDE\_PARAMS\_PARENTHESIS = 125 [line 207]  
**used when an inner ( ) is encountered while parsing an include/require/include\_once/include\_once statement**

PARSER\_EVENT\_INLINE\_DOCKEYWORD = 119 [line 180]  
**used when an inline {@tag} is encountered in parsing a DocBlock**

PARSER\_EVENT\_LOGICBLOCK = 102 [line 93]  
**{ encountered in parsing a function or php code**

PARSER\_EVENT\_METHOD = 135 [line 257]  
**used by the HighlightParser only, when a method starts**

PARSER\_EVENT\_METHOD\_LOGICBLOCK = 136 [line 262]  
**used by the HighlightParser only, when a method body is parsed**

PARSER\_EVENT\_NOEVENTS = 103 [line 98]  
**used for the beginning of parsing, before first < ? php encountered**

PARSER\_EVENT\_OUTPHP = 118 [line 175]  
**used when a ? > (with no space) is encountered in parsing**

PARSER\_EVENT\_PHPCODE = 106 [line 113]  
**used when php code processor instruction (< ? php) is encountered in parsing**

PARSER\_EVENT\_QUOTE = 101 [line 88]  
**used when double quotation mark ("") encountered in parsing**

PARSER\_EVENT\_QUOTE\_VAR = 138 [line 272]  
**used by the HighlightParser only, when {\$var} is encountered in a string**

PARSER\_EVENT\_SINGLEQUOTE = 110 [line 133]  
**used when a single quote (' ) is encountered in parsing**

PARSER\_EVENT\_STATIC\_VAR = 131 [line 237]  
**used when parsing a "static \$var1, \$var2;" declaration in a function**

PARSER\_EVENT\_STATIC\_VAR\_VALUE = 132 [line 242]

**used when parsing the value in a "static \$var1 = x" declaration in a function**

PARSER\_EVENT\_TAGS = 127 [line 217]

**used when parsing the @tag block of a docblock**

PARSER\_EVENT\_VAR = 116 [line 165]

**used when a var statement is encountered in parsing a class definition**

PARSER\_EVENT\_VAR\_ARRAY = 142 [line 292]

**used when a variable value is an array**

PARSER\_EVENT\_VAR\_ARRAY\_COMMENT = 143 [line 297]

**used when a comment is found in a variable array value**

PHPDOCUMENTOR\_EVENT\_CLASS = 800 [line 45]

**Class published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_CONST = 806 [line 57]

**Class Constant published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_DEFINE = 805 [line 55]

**Constant (define) published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_DOCBLOCK = 801 [line 47]

**DocBlock published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_DOCBLOCK\_TEMPLATE = 814 [line 84]

**used when a docblock template is encountered in the source**

PHPDOCUMENTOR\_EVENT\_END\_DOCBLOCK\_TEMPLATE = 815 [line 86]

**used when a docblock template is encountered in the source**

PHPDOCUMENTOR\_EVENT\_END\_PAGE = 808 [line 67]

**used to inform phpDocumentor\_IntermediateParser that the current file has been completely parsed.**

Render then flushes all buffers for functions/classes/defines/includes on the current page

- See [phpDocumentor\\_IntermediateParser::HandleEvent\(\)](#)

PHPDOCUMENTOR\_EVENT\_FUNCTION = 802 [line 49]

**Function published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_GLOBAL = 813 [line 82]

**used when a global variable definition is encountered in the source**

PHPDOCUMENTOR\_EVENT\_INCLUDE = 810 [line 71]

**Include (include/require/include\_once/include\_once)** published to  
IntermediateParser with this event

PHPDOCUMENTOR\_EVENT\_MESSAGE = 807 [line 59]

- **Deprecated**

PHPDOCUMENTOR\_EVENT\_NEWFILE = 811 [line 78]

**use to inform ErrorTracker of a new file being parsed**

PHPDOCUMENTOR\_EVENT\_NEWLINENUM = 812 [line 80]

**use to inform ErrorTracker of the next line number being parsed**

PHPDOCUMENTOR\_EVENT\_NEWSTATE = 808 [line 61]

**use to inform IntermediateParser of a new element being parsed**

PHPDOCUMENTOR\_EVENT\_PACKAGEPAGE = 809 [line 69]

**Package-level page published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_PAGE = 804 [line 53]

**New File (page) published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_README\_INSTALL\_CHANGELOG = 812 [line 75]

## **Contents of README/INSTALL/CHANGELOG files published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_TUTORIAL = 811 [line 73]

**Tutorial published to IntermediateParser with this event**

PHPDOCUMENTOR\_EVENT\_VAR = 803 [line 51]

**Class Variable published to IntermediateParser with this event**

STATE\_ACCESS\_MODIFIER = 239 [line 279]

**currently parsing an access modifier**

STATE\_ARRAY = 215 [line 162]

**currently parsing an array**

STATE\_CLASS = 211 [line 140]

**currently parsing a class definition**

STATE\_CLASS\_CONSTANT = 241 [line 289]

**currently parsing a class constant**

STATE\_CLASS\_MEMBER = 237 [line 269]

**currently parsing a class member using the HighlightParser**

STATE\_COMMENT = 205 [line 110]

**currently parsing a short comment //**

STATE\_COMMENTBLOCK = 204 [line 105]

**currently parsing a long comment /x x/ where x is an asterisk**

STATE\_DEFINE = 207 [line 120]

**currently parsing a define statement**

STATE\_DEFINE\_PARAMS = 208 [line 125]

**currently parsing the stuff in ( ) of a define statement**

STATE\_DEFINE\_PARAMS\_PARENTHESIS = 220 [line 187]

**currently parsing an inner parenthetical statement of a define( )**

STATE\_DESC = 226 [line 214]  
**currently parsing the desc part of a docblock**

STATE\_DOCBLOCK = 212 [line 147]  
**currently parsing a DocBlock**

STATE\_DOCBLOCK\_TEMPLATE = 233 [line 249]  
**currently parsing the value in a "static \$var1 = x" declaration in a function**

STATE\_DOCKEYWORD = 213 [line 152]  
**currently parsing a @tag in a DocBlock**

STATE\_DOCKEYWORD\_EMAIL = 214 [line 157]  
**currently parsing an email in brackets in an @author tag of a DocBlock**

STATE\_END\_CLASS = 311 [line 142]  
**used to tell Render that a class has been completely parsed, and to flush buffers**

STATE\_END\_DOCBLOCK\_TEMPLATE = 234 [line 254]  
**currently parsing the value in a "static \$var1 = x" declaration in a function**

STATE\_EOFQUOTE = 222 [line 194]  
**currently parsing a string defined using Perl <<<**

STATE\_ESCAPE = 1000 [line 42]  
**used when a backslash is encountered in parsing a string or other escapable entity**

STATE\_FUNCTION = 217 [line 172]  
**currently parsing a Function or Method**

STATE\_FUNCTION\_PARAMS = 209 [line 130]  
**currently parsing the stuff in ( ) of a function definition**

STATE\_FUNCTION\_PARAM\_VAR = 244 [line 304]  
**currently parsing a \$param in a function definition**

STATE\_FUNC\_GLOBAL = 230 [line 234]  
**currently parsing a "global \$var1, \$var2;" declaration in a function**

STATE\_GLOBAL = 228 [line 224]  
**currently parsing a global variable declaration**

STATE\_GLOBAL\_VALUE = 229 [line 229]  
**currently parsing the default value in a global variable declaration**

STATE\_IMPLEMENTS = 240 [line 284]  
**currently parsing an implements clause**

STATE\_INCLUDE = 223 [line 199]  
**currently parsing an include/require/include\_once/include\_once**

STATE\_INCLUDE\_PARAMS = 224 [line 204]  
**currently parsing the stuff in ( ) of a define statement**

STATE\_INCLUDE\_PARAMS\_PARENTHESIS = 225 [line 209]  
**currently parsing an inner parenthetical statement of an  
include/includeonce/require/requireonce( )**

STATE\_INLINE\_DOCKEYWORD = 219 [line 182]  
**currently parsing an inline tag like { @link} in a DocBlock**

STATE\_LOGICBLOCK = 202 [line 95]  
**currently parsing a { } block**

STATE\_METHOD = 235 [line 259]  
**currently parsing a method using the HighlightParser**

STATE\_METHOD\_LOGICBLOCK = 236 [line 264]  
**currently parsing the method body using the HighlightParser**

STATE\_NOEVENTS = 203 [line 100]  
**out of < ? php tag**

STATE\_OUTPHP = 218 [line 177]  
**currently out of php code**

STATE\_PHPCODE = 206 [line 115]  
**currently parsing php code**

STATE\_QUOTE = 201 [line 90]  
**currently parsing a quote**

STATE\_QUOTE\_VAR = 238 [line 274]  
**currently parsing a {\$enclosed\_var} using the HighlightParser**

STATE\_SINGLEQUOTE = 210 [line 135]  
**currently parsing a string enclosed in single quotes ('')**

STATE\_STATIC\_VAR = 231 [line 239]  
**currently parsing a "static \$var1, \$var2;" declaration in a function**

STATE\_STATIC\_VAR\_VALUE = 232 [line 244]  
**currently parsing the value in a "static \$var1 = x" declaration in a function**

STATE\_TAGS = 227 [line 219]  
**currently parsing the @tag block of a docblock**

STATE\_VAR = 216 [line 167]  
**currently parsing a Class variable**

STATE\_VAR\_ARRAY = 242 [line 294]  
**currently parsing a variable value is an array**

STATE\_VAR\_ARRAY\_COMMENT = 243 [line 299]  
**currently parsing a comment in a variable array value**

T\_ABSTRACT = 'foo' [line 312]  
T\_CONST = 'foo' [line 310]  
T\_DOC\_COMMENT = T\_ML\_COMMENT [line 325]  
T\_FINAL = 'foo' [line 316]  
T\_IMPLEMENTS = 'foo' [line 317]  
T\_INTERFACE = 'foo' [line 308]  
T\_ML\_COMMENT = T\_COMMENT [line 321]  
T\_PRIVATE = 'foo' [line 313]  
T\_PROTECTED = 'foo' [line 315]

T\_PUBLIC = 'foo' [*line 314*]

**phpDocumentorTParser.inc**  
**tokenizer extension-based parser for PHP code**  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: phpDocumentorTParser.inc 286921 2009-08-08 05:01:24Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2
- **License** [LGPL](#)

# TutorialHighlightParser.inc

## Source Code Highlighting

The classes in this file are responsible for the dynamic @example, and <programlisting role="tutorial"> tags output. Using the WordParser, the phpDocumentor\_TutorialHighlightParser retrieves PHP tokens one by one from the array generated by [WordParser](#) source retrieval functions and then highlights them individually.

It accomplishes this highlighting through the assistance of methods in the output Converter passed to its parse() method, and then returns the fully highlighted source as a string

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2003-2007 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Tutorial** [@example](#), [@filesource](#)
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: TutorialHighlightParser.inc 246148 2007-11-14 01:57:04Z ashnazg \$
- **Copyright** 2003-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - PHPCS needs to ignore CVS Id length
- **Since** 1.3.0
- **License** [LGPL](#)

STATE\_TUTORIAL\_ATTRIBUTE = 104 [line 568]  
**currently parsing an open <tag>**

STATE\_TUTORIAL\_CLOSETAG = 105 [line 578]  
**currently parsing a close </tag>**

STATE\_TUTORIAL\_COMMENT = 107 [line 598]  
**currently parsing a <!-- comment -->**

STATE\_TUTORIAL\_DOUBLEQUOTE = 109 [line 618]  
**currently parsing a <!-- comment -->**

STATE\_TUTORIAL\_ENTITY = 106 [line 588]  
**currently parsing an &entity;**

STATE\_TUTORIAL\_ITAG = 102 [line 548]  
**currently parsing an {@inline tag}**

STATE\_TUTORIAL\_NOEVENTS = 101 [line 538]  
**currently in starting state**

STATE\_TUTORIAL\_OPENTAG = 103 [line 558]  
**currently parsing an open <tag>**

STATE\_TUTORIAL\_SINGLEQUOTE = 108 [line 608]  
**currently parsing a <!-- comment -->**

TUTORIAL\_EVENT\_ATTRIBUTE = 4 [line 563]  
**used when a <tag attr="attribute"> is found**

TUTORIAL\_EVENT\_CLOSETAG = 5 [line 573]  
**used when a close </tag> is found**

TUTORIAL\_EVENT\_COMMENT = 7 [line 593]  
**used when a <!-- comment --> is found**

TUTORIAL\_EVENT\_DOUBLEQUOTE = 9 [line 613]  
**used when a <!-- comment --> is found**

TUTORIAL\_EVENT\_ENTITY = 6 [line 583]  
**used when an &entity; is found**

TUTORIAL\_EVENT\_ITAG = 2 [line 543]  
**used when an {@inline tag} is found**

TUTORIAL\_EVENT\_NOEVENTS = 1 [line 533]  
**starting state**

TUTORIAL\_EVENT\_OPENTAG = 3 [line 553]  
**used when an open <tag> is found**

TUTORIAL\_EVENT\_SINGLEQUOTE = 8 [line 603]  
**used when a <!-- comment --> is found**

# XMLpackagePageParser.inc

## Parser for XML DocBook-based phpDocumentor tutorials

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2007 Gregory Beaver

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Tutorial** [phpDocumentor Tutorials](#)
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** CVS: \$Id: XMLpackagePageParser.inc 246143 2007-11-14 01:31:24Z ashnazg \$
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - PHPCS needs to ignore CVS Id length
- **Since** 1.2
- **License** [GPL](#)

PHPDOCUMENTOR\_PDP\_EVENT\_ATTRIBUTES = 603 [line 69]  
**when tag attributes name="value" are found**

PHPDOCUMENTOR\_PDP\_EVENT\_CDATA = 602 [line 61]  
**when <![CDATA[ ]]> is found**

PHPDOCUMENTOR\_PDP\_EVENT\_ENTITY = 604 [line 77]

**when tag attributes name="value" are found**

PHPDOCUMENTOR\_PDP\_EVENT\_PROGRAMLISTING = 600 [line 45]

**when <programlisting> is found**

PHPDOCUMENTOR\_PDP\_EVENT\_TAG = 601 [line 53]

**when a DocBook <tag> is found**

PHPDOCUMENTOR\_PDP\_STATE\_ATTRIBUTES = 703 [line 73]

**when tag attributes name="value" are found**

PHPDOCUMENTOR\_PDP\_STATE\_CDATA = 702 [line 65]

**when <![CDATA[ ]]> is found**

PHPDOCUMENTOR\_PDP\_STATE\_ENTITY = 704 [line 81]

**when tag attributes name="value" are found**

PHPDOCUMENTOR\_PDP\_STATE\_PROGRAMLISTING = 700 [line 49]

**when <programlisting> is found**

PHPDOCUMENTOR\_PDP\_STATE\_TAG = 701 [line 57]

**when a DocBook <tag> is found**

## Class Parser

[line 338]

**PHP Parser for PHP 4.2.3-**

This parser is slower than the tokenizer-based parser, and is deprecated.

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>

- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** \$Id: Parser.inc 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2000-2007 Kellin, Joshua Eichorn
- **Deprecated** in favor of [phpDocumentorTParser](#)

Constructor `void function Parser::Parser() [line 487]`

### **Set up invariant parsing variables**

`void function Parser::categoryTagHandler($word) [line 2046]`

#### **Function Parameters:**

- `string $word`

### **handles @category**

Tag Handlers

- **Tutorial** [@category](#)

`mixed function Parser::checkEventPop($word, $pevent) [line 2612]`

#### **Function Parameters:**

- `$word`
- `$pevent`

**this function checks whether parameter \$word is a token for popping the current event off of the Event Stack.**

`mixed function Parser::checkEventPush($word, $pevent) [line 2590]`

#### **Function Parameters:**

- `$word`
- `$pevent`

**this function checks whether parameter \$word is a token for pushing a new event onto the Event Stack.**

*void function Parser::configWordParser(\$e) [line 2957]*

**Function Parameters:**

- *\$value \$e* integer an event number

**tell the parser's WordParser \$wp to set up tokens to parse words by.**

tokens are word separators. In English, a space or punctuation are examples of tokens. In PHP, a token can be a ;, a parenthesis, or even the word "function"

- See [WordParser](#)

*void function Parser::defaultTagHandler(\$word) [line 1989]*

**Function Parameters:**

- *string \$word*

**Handles all standard tags that only have a description**

Tag Handlers

*void function Parser::endTag() [line 1971]*

**Called to clean up at the end of parsing a @tag in a docblock**

*void function Parser::exampleTagHandler(\$word) [line 2031]*

**Function Parameters:**

- *string \$word*

**handles @example**

Tag Handlers

- Tutorial [@example](#)

*void* function Parser::getParserEventName(\$value) [line 2968]

**Function Parameters:**

- *\$value* **\$value** integer an event number

**Debugging function, takes an event number and attempts to return its name**

*void* function Parser::globalTagHandler(\$word) [line 2062]

**Function Parameters:**

- *string* **\$word**

**handles @global**

Tag Handlers

- Tutorial [@global](#)

*void* function Parser::invalidTagHandler(\$word) [line 2003]

**Function Parameters:**

- *string* **\$word**

**Handles tags like '@filesource' that only work in PHP 4.3.0+**

Tag Handlers

*void* function Parser::packageTagHandler(\$word) [line 2015]

**Function Parameters:**

- *string* \$word

**handles @package**

Tag Handlers

- **Tutorial** [@package](#)

*void* function Parser::paramTagHandler(\$word) [line 2192]

**Function Parameters:**

- *string* \$word

**handles @param**

Tag Handlers

- **Tutorial** [@param](#)

*bool* function Parser::parse(&\$parse\_data, \$path, [\$base = 0], [\$packages = false], \$parse\_data) [line 509]

**Function Parameters:**

- *string* \$parse\_data
- *string* \$path
- *int* \$base number of directories to drop off the bottom when creating names using path
- &\$parse\_data
- \$packages

## Parse a new file

- **Static Variable Used** integer \$endrecur: used for recursion limiting if a handler for an event is not found

*void function Parser::propertyTagHandler(\$word) [line 2276]*

**Function Parameters:**

- *string \$word*

**Handles @property(-read or -write) and @method magic tag  
Tag Handlers**

*void function Parser::returnTagHandler(\$word) [line 2228]*

**Function Parameters:**

- *string \$word*

**handles @return  
Tag Handlers**

- **Tutorial** [@return](#)

*void function Parser::setupStates() [line 2628]*

**setup the parser tokens, and the pushEvent/popEvent arrays**

- See [Publisher::\\$tokens](#), [Publisher::\\$pushEvent](#), [Publisher::\\$popEvent](#)

*void function Parser::staticvarTagHandler(\$word) [line 2134]*

**Function Parameters:**

- *string \$word*

**handles @staticvar**

Tag Handlers

- [Tutorial @staticvar](#)

*void function Parser::usesTagHandler(\$word) [line 2170]*

**Function Parameters:**

- *string \$word*

**handles @uses**

Tag Handlers

- [Tutorial @uses](#)

*void function Parser::varTagHandler(\$word) [line 2252]*

**Function Parameters:**

- *string \$word*

**handles @var**  
Tag Handlers

- Tutorial [@var](#)

## Class parserDescParser

*[line 298]*

**Parses a DocBlock description to retrieve abstract representations of**  
`<pre>,<code>,<p>,<ul>,<ol>,<li>,<b>,<i>`

- **Package** phpDocumentor
- **Sub-Package** Parsers
- Tutorial [phpDocumentor Tutorial](#)
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: Parser.inc 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2000-2007 Kellin, Joshua Eichorn
- **Since** 1.2

Constructor `void` function `parserDescParser::parserDescParser()` *[line 349]*

**sets \$wp to be a** [ObjectWordParser](#)

\$wp is the word parser that retrieves tokens

`boolean` function `parserDescParser::doSimpleList($word)` *[line 1188]*

**Function Parameters:**

- `string` **\$word** line that may contain a simple list

**Return a simple list, if found**

This helper function extracts a simple list beginning with any of 'o', '- '#'','+','0','1','0.','1.' and starts parsing it.

*void* function parserDescParser::getParserEventName(\$value) [line 1462]

**Function Parameters:**

- **\$value**

*void* function parserDescParser::parse(&\$parse\_data, [\$sdesc = false], [\$ind\_type = 'parserDesc']) [line 363]

**Function Parameters:**

- **array &\$parse\_data** array of strings or [parserInlineTags](#)
- **boolean \$sdesc** true if the description is a short description. (only 1 paragraph allowed in short desc)
- **string \$ind\_type** name of the class to instantiate for each paragraph. parserDesc for desc/sdesc, parserStringWithInlineTags for tag data

## Parse a long or short description for tags

- **Static Variable Used** integer \$endrecur: used for recursion limiting if a handler for an event is not found
- **Used by** [parserDocBlock::postProcess\(\)](#)

*void* function parserDescParser::setupStates(\$sdesc) [line 1231]

**Function Parameters:**

- **boolean \$sdesc** determines whether to allow paragraph parsing

## setup the parser tokens, and the pushEvent/popEvent arrays

- **Global Variable Used** boolean 0: used to determine whether to slow things down or not by eliminating whitespace from comments
- **See** [Publisher::\\$tokens](#), [Publisher::\\$pushEvent](#), [Publisher::\\$popEvent](#)

# Class phpDocumentorTParser

[line 56]

## Tokenizer-based parser for PHP source code

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

### phpDocumentorTParser::\$eventHandlers

```
mixed = array(  
    PARSER_EVENT_ARRAY          => 'handleArray',  
    PARSER_EVENT_VAR_ARRAY       => 'handleArray',  
    PARSER_EVENT_VAR_ARRAY_COMMENT => 'handleVarArrayComment',  
    PARSER_EVENT_CLASS           => 'handleClass',  
    PARSER_EVENT_COMMENT         => 'handleComment',  
    PARSER_EVENT_DOCBLOCK_TEMPLATE => 'handleDocBlockTemplate',  
    PARSER_EVENT_END_DOCBLOCK_TEMPLATE => 'handleEndDocBlockTemplate',  
    PARSER_EVENT_LOGICBLOCK      => 'handleLogicBlock',  
    PARSER_EVENT_NOEVENTS        => 'defaultHandler',  
    PARSER_EVENT_OUTPHP          => 'defaultHandler',  
    PARSER_EVENT_DEFINE          => 'handleDefine',  
    PARSER_EVENT_DEFINE_PARAMS   => 'handleDefineParams',  
    PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS => 'handleDefineParamsParenthesis',  
    PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS => 'handleIncludeParamsParenthesis',  
    PARSER_EVENT_DOCBLOCK         => 'handleDocBlock',  
    PARSER_EVENT_TAGS            => 'handleTags',  
    PARSER_EVENT_DESC             => 'handleDesc',  
    PARSER_EVENT_DOCKEYWORD      => 'handleTag',  
    PARSER_EVENT_DOCKEYWORD_EMAIL => 'handleDockeywordEmail',  
    PARSER_EVENT_EOFQUOTE         => 'handleHereDoc',  
    PARSER_EVENT_FUNCTION         => 'handleFunction',  
    PARSER_EVENT_FUNCTION_PARAMS  => 'handleFunctionParams',
```

```

PARSER_EVENT_FUNCTION_PARAM_VAR      => 'handleFunctionParams',
PARSER_EVENT_FUNC_GLOBAL           => 'handleFuncGlobal',
PARSER_EVENT_DEFINE_GLOBAL         => 'handleGlobal',
PARSER_EVENT_GLOBAL_VALUE          => 'handleGlobalValue',
PARSER_EVENT_INLINE_DOCKEYWORD    => 'handleInlineDockeyword',
PARSER_EVENT_INCLUDE              => 'handleInclude',
PARSER_EVENT_INCLUDE_PARAMS       => 'handleIncludeParams',
PARSER_EVENT_QUOTE               => 'handleQuote',
PARSER_EVENT_PHPCODE              => 'handlePhpCode',
PARSER_EVENT_SINGLEQUOTE          => 'handleSingleQuote',
PARSER_EVENT_STATIC_VAR           => 'handleStaticVar',
PARSER_EVENT_STATIC_VAR_VALUE     => 'handleStaticValue',
PARSER_EVENT_VAR                 => 'handleVar',
PARSER_EVENT_ACCESS_MODIFIER     => 'handleAccessModifier',
PARSER_EVENT_IMPLEMENTS          => 'handleImplements',
PARSER_EVENT_CLASS_CONSTANT       => 'handleClassConstant',
) [line 147]

```

### **phpDocumentorTParser::\$inlineTagHandlers**

```

mixed = array(
'*'  => 'handleDefaultInlineTag',
'link' => 'handleLinkInlineTag',
) [line 188]

```

### **phpDocumentorTParser::\$source\_location**

```
string = [line 146]
```

## **relative path of the parsed file from the base parse directory**

Constructor void function phpDocumentorTParser::phpDocumentorTParser() [line 197]

### **Constructor**

bool function phpDocumentorTParser::parse(&\$parse\_data, \$path, [\$base = 0], [\$packages = false]) [line 225]  
**Function Parameters:**

- **string &\$parse\_data** the parse data
- **string \$path** the path
- **int \$base** number of directories to drop off the bottom  
names using path when creating
- **bool \$packages ???**

## **Parse a new file**

- **Static Variable Used** int \$endrecur: used for recursion limiting if a handler for an event is not found

## Class phpDocumentor\_HighlightParser [line 226]

Highlights source code using [parse\(\)](#)

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [celog@php.net](mailto:celog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change class name to PhpDocumentor\_\*
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2.0beta3
- **Usedby** [parserExampleInlineTag::arrayConvert\(\)](#) - Parses the tokenized source
- **Usedby** [parserSourceInlineTag::arrayConvert\(\)](#) - Parses the tokenized source
- **Usedby** [parserExampleTag::ConvertSource\(\)](#) - highlights source code
- **Usedby** [parserFileSourceTag::ConvertSource\(\)](#) - highlights source code
- **License** [LGPL](#)

Constructor **void** function phpDocumentor\_HighlightParser::phpDocumentor\_HighlightParser() [line 2322]

**Initialize the \$tokenpushEvent, \$wordpushEvent arrays**

**void** function phpDocumentor\_HighlightParser::configWordParser(&\$data) [line 2231]

**Function Parameters:**

- **array** **&\$data** all tokens separated by line number

**Give the word parser necessary data to begin a new parse**

- **Usedby** [phpDocumentor\\_HighlightParser::parse\(\)](#) - pass \$parse\_data to prepare retrieval of

`void function phpDocumentor_HighlightParser::newLineNum() [line 346]`  
**wraps the current line (via the converter) and resets it to empty**

- **Uses** Converter::SourceLine() - encloses \$\_line in a converter-specific format

`bool function phpDocumentor_HighlightParser::parse(&$parse_data, &$converter, [$inlinesourceparse = false], [$class = false], [$linenum = false], [$filesourcepath = false]) [line 406]`

**Function Parameters:**

- **array &\$parse\_data** the parse data
- **[Converter](#) &\$converter** the converter object
- **bool \$inlinesourceparse** whether this data is from an inline {@source} tag
- **string/false \$class** if a string, it is the name of the class whose method we are parsing containing a {@source} tag
- **false/integer \$linenum** starting line number from {@source linenum}
- **false/string \$filesourcepath** full path to file with @filesource tag, if this is a @filesource parse

### Parse a new file

The parse() method is a do...while() loop that retrieves tokens one by one from the \$\_event\_stack, and uses the token event array set up by the class constructor to call event handlers.

The event handlers each process the tokens passed to them, and use the \_addoutput() method to append the processed tokens to the \$\_line variable. The word parser calls [newLineNum\(\)](#) every time a line is reached.

In addition, the event handlers use special linking functions \_link() and its cousins (\_classlink(), etc.) to create in-code hyperlinks to the documentation for source code elements that are in the source code.

- **Static Variable Used** int \$endrecur: used for recursion limiting if a handler for event is not found
- **TODO** CS cleanup - rename tokenizer\_ext constant to uppercase
- **Uses** [phpDocumentor\\_HighlightParser::setupStates\(\)](#) - initialize parser state variables
- **Uses** [phpDocumentor\\_HighlightParser::configWordParser\(\)](#) - pass \$parse\_data to prepare retrieval of tokens

void function phpDocumentor\_HighlightParser::setLineNum(\$num) [[line 365](#)]

**Function Parameters:**

- *int \$num* line number

### Start the parsing at a certain line number

void function phpDocumentor\_HighlightParser::setupStates(\$inlinesourceparse, \$class) [[line 2248](#)]

**Function Parameters:**

- *bool \$inlinesourceparse* true if we are highlighting an inline {@source} tag's output
- *false/string \$class* name of class we are going to start from

### Initialize all parser state variables

- **Used by** [phpDocumentor\\_HighlightParser::parse\(\)](#) - initialize parser state variables
- **Uses** \$\_wp - sets to a new [phpDocumentor\\_HighlightWordParser](#)

## Class phpDocumentor\_HighlightWordParser

[[line 69](#)]

### Retrieve tokens from an array of tokens organized by line numbers

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change class name to PhpDocumentor\_\*
- **Since** 1.2.0beta3
- **License** [LGPL](#)

*void* function phpDocumentor\_HighlightWordParser::backupPos(\$last\_token, [\$is\_pos = false]) [line [172](#)]

**Function Parameters:**

- *array/string* **\$last\_token** token, or output from [nextToken\(\)](#)
- *bool* **\$is\_pos** if true, backupPos interprets \$last\_token to be the position in the internal token array of the last token

**back the word parser to the previous token as defined by \$last\_token**

*array/string* function phpDocumentor\_HighlightWordParser::getWord() [line [147](#)]

**Retrieve the next token**

*array* function phpDocumentor\_HighlightWordParser::nextToken() [line [127](#)]

**Retrieve the position of the next token that will be parsed in the internal token array**

*void* function phpDocumentor\_HighlightWordParser::setup(&\$input, &\$parser) [line [85](#)]

**Function Parameters:**

- *array* **&\$input** the input
- [phpDocumentor\\_HighlightParser](#) **&\$parser** the parser

**Initialize the parser object**

# Class phpDocumentor\_peardoc2\_XML\_Beautifier

[line 49]

This is just like XML\_Beautifier, but uses  
[phpDocumentor XML Beautifier Tokenizer](#)

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Since** 1.3.0

*mixed* function phpDocumentor\_peardoc2\_XML\_Beautifier::formatFile(\$file, [\$newFile = null], [\$renderer = "Plain"]) [line 63]

**Function Parameters:**

- **string \$file** filename
- **mixed \$newFile** filename for beautified XML file (if none is given, the XML string will be returned.) if you want overwrite the original file, use XML\_BEAUTIFIER\_OVERWRITE
- **string \$renderer** Renderer to use, default is the plain xml renderer

## format a file or URL

- **Throws** PEAR\_Error
- **Access** public
- **Uses** \_loadRenderer() - to load the desired renderer

*string* function phpDocumentor\_peardoc2\_XML\_Beautifier::formatString(\$string, [\$renderer = "Plain"]) [line 109]

**Function Parameters:**

- **string \$string** XML
- **\$renderer**

## format an XML string

- **Throws** PEAR\_Error
- **Access** public

## Class phpDocumentor\_TutorialHighlightParser

[line 69]

Highlights source code using [parse\(\)](#)

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2003-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **License** [LGPL](#)

Constructor void function phpDocumentor\_TutorialHighlightParser::phpDocumentor\_TutorialHighlightParser()  
[line 423]

**Initialize the \$tokenpushEvent, \$wordpushEvent arrays**

mixed function phpDocumentor\_TutorialHighlightParser::checkEventPop(\$word, \$pevent) [line 380]

**Function Parameters:**

- *string/array* **\$word** token value
- *integer* **\$pevent** parser event from [Parser.inc](#)

**This function checks whether parameter \$word is a token for popping the current event off of the Event Stack.**

- **TODO** CS cleanup - PHP-CS needs to recognize docblock template tags

*mixed* function `phpDocumentor_TutorialHighlightParser::checkEventPush($word, $pevent)` [*line 358*]

**Function Parameters:**

- *string/array* **\$word** token value
- *integer* **\$pevent** parser event from [Parser.inc](#)

**This function checks whether parameter \$word is a token for pushing a new event onto the Event Stack.**

- **TODO** CS cleanup - PHP-CS needs to recognize docblock template tags

*void* function `phpDocumentor_TutorialHighlightParser::configWordParser($e)` [*line 341*]

**Function Parameters:**

- *integer* **\$e** an event number

**Tell the parser's WordParser \$wp to set up tokens to parse words by.**

Tokens are word separators. In English, a space or punctuation are examples of tokens. In PHP, a token can be a ;, a parenthesis, or even the word "function"

- See [WordParser](#)
- Used by [phpDocumentor\\_TutorialHighlightParser::parse\(\)](#) - pass \$parse\_data to prepare retrieval of tokens

*string/int* function `phpDocumentor_TutorialHighlightParser::getParserEventName($value)` [line 508]

**Function Parameters:**

- *int \$value* the event number

**searches for a parser event name based on its number**

*void* function `phpDocumentor_TutorialHighlightParser::newLineNum()` [line 115]

**advances output to a new line**

- **Uses** `Converter::SourceLine()` - encloses `$_line` in a converter-specific format

*bool* function `phpDocumentor_TutorialHighlightParser::parse($parse_data, &$converter, [$filesourcepath = false], [$linenum = false])` [line 166]

**Function Parameters:**

- *string \$parse\_data* blah
- *[Converter](#) &\$converter* blah
- *false/string \$filesourcepath* full path to file with @filesource tag, if this is a @filesource parse
- *false/integer \$linenum* starting line number from {@source linenum}

### Parse a new file

The `parse()` method is a do...while() loop that retrieves tokens one by one from the `$_event_stack`, and uses the token event array set up by the class constructor to call event handlers.

The event handlers each process the tokens passed to them, and use the `_addoutput()` method to append the processed tokens to the `$_line` variable. The word parser calls [newLineNum\(\)](#) every time a line is reached.

In addition, the event handlers use special linking functions `_link()` and its cousins (`_classlink()`, etc.) to create in-code hyperlinks to the documentation for source code elements that are in the source code.

- **Static Variable Used** integer \$endrecur: used for recursion limiting if a handler for an event is not found
- **TODO** CS cleanup - unable to get function signature below 85char wide
- **Uses** [phpDocumentor\\_TutorialHighlightParser::setupStates\(\)](#) - initialize parser state variables
- **Uses** [phpDocumentor\\_TutorialHighlightParser::configWordParser\(\)](#) - pass \$parse\_data to prepare retrieval of tokens

*void* function phpDocumentor\_TutorialHighlightParser::setLineNum(\$num) [*line 130*]

**Function Parameters:**

- *int \$num* the line number

## Start the parsing at a certain line number

*void* function phpDocumentor\_TutorialHighlightParser::setupStates(\$parsedata) [*line 403*]

**Function Parameters:**

- *bool/string \$parsedata* true if we are highlighting an inline {@source} tag's output, or the name of class we are going to start from

## Initialize all parser state variables

- **Used by** [phpDocumentor\\_TutorialHighlightParser::parse\(\)](#) - initialize parser state variables
- **Uses** \$\_wp - sets to a new [phpDocumentor\\_HighlightWordParser](#)

# Class phpDocumentor\_XML\_Beautifier\_Tokenizer

[*line 50*]

## Highlights source code using parse()

- **Package** phpDocumentor
- **Sub-Package** Parsers

### **phpDocumentor\_XML\_Beautifier\_Tokenizer::\$eventHandlers**

```
array = array(
    PHPDOC_XMLTOKEN_EVENT_NOEVENTS => 'normalHandler',
    PHPDOC_XMLTOKEN_EVENT_XML => 'parseXMLHandler',
    PHPDOC_XMLTOKEN_EVENT_PI => 'parsePiHandler',
    PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE => 'attrHandler',
    PHPDOC_XMLTOKEN_EVENT_OPENTAG => 'tagHandler',
    PHPDOC_XMLTOKEN_EVENT_IN_CDATA => 'realcdataHandler',
    PHPDOC_XMLTOKEN_EVENT_DEF => 'defHandler',
    PHPDOC_XMLTOKEN_EVENT_CLOSETAG => 'closetagHandler',
    PHPDOC_XMLTOKEN_EVENT_ENTITY => 'entityHandler',
    PHPDOC_XMLTOKEN_EVENT_COMMENT => 'commentHandler',
    PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE => 'stringHandler',
    PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE => 'stringHandler',
    PHPDOC_XMLTOKEN_EVENT_CDATA => 'parseCdataHandler',
)
) [line 64]
```

Constructor *void* function

`phpDocumentor_XML_Beautifier_Tokenizer::phpDocumentor_XML_Beautifier_Tokenizer()` [line 570]

### **Initialize the \$tokenpushEvent, \$wordpushEvent arrays**

*mixed* function `phpDocumentor_XML_Beautifier_Tokenizer::checkEventPop($word, $pevent)` [line 535]

#### **Function Parameters:**

- **\$word**
- **\$pevent**

**this function checks whether parameter \$word is a token for popping the current event off of the Event Stack.**

*mixed* function `phpDocumentor_XML_Beautifier_Tokenizer::checkEventPush($word, $pevent)` [line 513]

#### **Function Parameters:**

- **\$word**
- **\$pevent**

**this function checks whether parameter \$word is a token for pushing a new event onto the Event Stack.**

*void* function `phpDocumentor_XML_Beautifier_Tokenizer::configWordParser($e)` [line [504](#)]

**Function Parameters:**

- **\$value** `$e` integer an event number

**tell the parser's WordParser \$wp to set up tokens to parse words by.**

tokens are word separators. In English, a space or punctuation are examples of tokens. In PHP, a token can be a ;, a parenthesis, or even the word "function"

- See [WordParser](#)
- Usedby [phpDocumentor XML\\_Beautifier\\_Tokenizer::parseString\(\)](#) - pass \$parse\_data to prepare retrieval of tokens

*void* function `phpDocumentor_XML_Beautifier_Tokenizer::getParserEventName($value)` [line [661](#)]

**Function Parameters:**

- **\$value**

*void* function `phpDocumentor_XML_Beautifier_Tokenizer::incdataHandler($parser, $cdata)` [line [408](#)]

**Function Parameters:**

- **object XML \$parser** parser object
- **string \$cdata** CDATA

**Handler for real character data**

- **Access** protected

*bool* function `phpDocumentor_XML_Beautifier_Tokenizer::parseString($parse_data, 1, 2, 3)` [line [108](#)]

**Function Parameters:**

- *Converter* 1
- *false|string* 2 full path to file with @filesource tag, if this is a @filesource parse
- *false/integer* 3 starting line number from {@source linenum}
- *string \$parse\_data*

**Parse a new file**

The parse() method is a do...while() loop that retrieves tokens one by one from the \$\_event\_stack, and uses the token event array set up by the class constructor to call event handlers.

The event handlers each process the tokens passed to them, and use the \_addoutput() method to append the processed tokens to the \$\_line variable. The word parser calls newLineNum() every time a line is reached.

In addition, the event handlers use special linking functions \_link() and its cousins (\_classlink(), etc.) to create in-code hyperlinks to the documentation for source code elements that are in the source code.

- **Static Variable Used** integer \$endrecur: used for recursion limiting if a handler for an event is not found
- **Uses** [phpDocumentor\\_XML\\_Beautifier\\_Tokenizer::setupStates\(\)](#) - initialize parser state variables
- **Uses** [phpDocumentor\\_XML\\_Beautifier\\_Tokenizer::configWordParser\(\)](#) - pass \$parse\_data to prepare retrieval of tokens

void function phpDocumentor\_XML\_Beautifier\_Tokenizer::setupStates(\$parsedata, 1) [[line 553](#)]

**Function Parameters:**

- *false/string* 1 name of class we are going to start from
- *boolean \$parsedata* true if we are highlighting an inline {@source} tag's output

**Initialize all parser state variables**

- **Used by** [phpDocumentor\\_XML\\_Beautifier\\_Tokenizer::parseString\(\)](#) - initialize parser state

- variables
- **Uses** `$_wp` - sets to a new [phpDocumentor\\_HighlightWordParser](#)

## Class ppageParser [line 3026]

### Global package page parser

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Tutorial** [phpDocumentor Tutorials](#)
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** \$Id: Parser.inc 238276 2007-06-22 14:58:30Z ashnazg \$
- **Copyright** 2000-2007 Kellin, Joshua Eichorn
- **Deprecated** in favor of tutorials

#### ppageParser::\$package

*string = false [line 3029]*

#### ppageParser::\$subpackage

*string = " [line 3031]*

Constructor **void** function ppageParser::ppageParser() [line 3035]

#### set up invariant Parser variables

**void** function ppageParser::defaultHandler(\$word, \$pevent) [line 3149]

#### Function Parameters:

- *string* **\$word** token
- *integer* **\$pevent** parser event

#### Handles all non-inline tags

*void* function ppageParser::handleInlineDockeyword(\$word, \$pevent) [*line 3165*]

**Function Parameters:**

- *string* **\$word** token
- *integer* **\$pevent** parser event

**handler for INLINE\_DOCKEYWORD.**

this handler recognizes {@inline tags} like link, and parses them, replacing them directly in the text flow with their output.

*mixed* function ppageParser::parse(&\$parse\_data, \$xml, [\$package = 'default'], [\$subpackage = ""], [\$tutorial = ""], [\$category = 'default'], [\$path = ""], \$parse\_data) [*line 3071*]

**Function Parameters:**

- *string* **\$parse\_data**
- *string* **\$package**
- *int* **\$subpackage**
- *&\$parse\_data*
- *\$xml*
- *\$tutorial*
- *\$category*
- *\$path*

**Parse a new file**

*void* function ppageParser::setupStates() [*line 3047*]

**set up invariant Parser variables**

## Class XMLPackagePageParser

[*line 98*]

**Used to parse XML DocBook-based tutorials**

- **Package** phpDocumentor

- **Sub-Package** Parsers
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2
- **License** [GPL](#)

### **XMLPackagePageParser::\$context**

*array = [line 125]*

- **Var** the tag stack

### **XMLPackagePageParser::\$eventHandlers**

```
array = array(
    PHPDOCUMENTOR_PDP_EVENT_TAG => 'handleTag',
    PHPDOCUMENTOR_PDP_EVENT_ATTRIBUTES => 'handleAttributes',
    PHPDOCUMENTOR_PDP_EVENT_CDATA => 'handleCData',
    PARSER_EVENT_NOEVENTS => 'defaultHandler',
    PARSER_EVENT_COMMENTBLOCK => 'ignoreHandler',
    PARSER_EVENT_OUTPHP => 'ignoreHandler',
    PARSER_EVENT_QUOTE => 'handleQuote',
    PHPDOCUMENTOR_PDP_EVENT_ENTITY => 'handleEntity',
)
) [line 103]
```

### **XMLPackagePageParser::\$pars**

*array = array() [line 117]*

### **XMLPackagePageParser::\$refsect1id**

*mixed = false [line 119]*

### **XMLPackagePageParser::\$refsect2id**

*mixed = false [line 120]*

### **XMLPackagePageParser::\$refsect3id**

*mixed = false [line 121]*

Constructor **void** function XMLPackagePageParser::XMLPackagePageParser() [line 138]

#### **Set up the wordparser**

```
1   function XMLPackagePageParser()
2   {
```

```
3     $this-> wp =new ObjectWordParser(true);  
4 }
```

- Uses [ObjectWordParser](#)

*mixed* function XMLPackagePageParser::getParserEventName(\$value) [line 615]

**Function Parameters:**

- *mixed* **\$value** a value

#### debugging function

```
1  function getParserEventName ($value)  
2 {  
3     $lookup= array(  
4         PARSER_EVENT_NOEVENTS  
5             => "PARSER_EVENT_NOEVENTS"  
6         PHPDOCUMENTOR_PDP_EVENT_TAG  
7             => "PHPDOCUMENTOR_PDP_EVENT_TAG"  
8         PHPDOCUMENTOR_PDP_EVENT_ATTRIBUTES  
9             => "PHPDOCUMENTOR_PDP_EVENT_ATTRIBUTES"  
10        PHPDOCUMENTOR_PDP_EVENT_CDATA  
11            => "PHPDOCUMENTOR_PDP_EVENT_CDATA"  
12        PHPDOCUMENTOR_PDP_EVENT_LIST  
13            => "PHPDOCUMENTOR_PDP_EVENT_LIST"  
14        PARSER_EVENT_QUOTE  
15            => "PARSER_EVENT_QUOTE"  
16        PHPDOCUMENTOR_PDP_EVENT_ENTITY  
17            => "PHPDOCUMENTOR_PDP_EVENT_ENTITY"  
18        PHPDOCUMENTOR_PDP_EVENT_COMMENT  
19            => "PHPDOCUMENTOR_PDP_EVENT_COMMENT"  
20        PHPDOCUMENTOR_PDP_EVENT_PI  
21            => "PHPDOCUMENTOR_PDP_EVENT_PI"  
22    );  
23    if (isset( $lookup[$value])) {  
24        return$lookup[$value];  
25    } else {  
26        return$value;  
27    }  
28 }
```

- **Static**

*bool* function XMLPackagePageParser::parse(\$parse\_data, \$tutorial) [/line 155]

**Function Parameters:**

- *string* **\$parse\_data** the parse data
- *array* **\$tutorial** for format, see [lo::getTutorials\(\)](#)

## Parse a new file

- **Static Variable Used** integer \$endrecur: used for recursion limiting if a handler for an event is not found
- **Uses** [parserTutorial](#) - using Publisher::PublishEvent(), a new tutorial is created from the file parsed, and passed to the Intermediate Parser

*void* function XMLPackagePageParser::setupStates() [/line 524]

**setup the parser tokens, and the pushEvent/popEvent arrays**

- See [Publisher::\\$tokens](#), [Publisher::\\$pushEvent](#), [Publisher::\\$popEvent](#)

# find\_phpdoc.php

**Utility file to locate phpDocumentor for a non-PEAR installation**

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2007 Gregory Beaver

## LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** setup
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: find\_phpdoc.php 246330 2007-11-17 03:09:41Z ashnazg \$
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change subpackage to Setup
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2
- **Filesource** [Source Code for this file](#)
- **License** [LGPL](#)

include " [/line [46](#)]

## Dummy value

# PackagePageElements.inc

## Data structures used in parsing XML DocBook-based tutorials

Conversion of DocBook-based tutorials is performed using special [Converter](#) class methods. By default, these methods simply retrieve simple rules for replacement of tags and slight re-ordering from the options.ini file present for every template.

In future versions, there may be utilization of xslt or other more powerful protocols. However, for most situations, the power of these classes will be more than sufficient to handle very complex documentation.

Note that an entire tutorial is contained in a single parserXMLDocBookTag, matching the document model for DocBook. The top-level tag, <refentry>, contains every other tag and all text.

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2008 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Tutorial
- **Tutorial** [phpDocumentor Tutorials](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** CVS: \$Id: PackagePageElements.inc 253643 2008-02-24 04:27:54Z ashnazg \$
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 1.2.0

- License [LGPL](#)

## Class parserCData [line 72]

Represents <![CDATA[ ]]> sections.

These sections are interpreted as plain text

- Package phpDocumentor
- Sub-Package Tutorial
- Tutorial [phpDocumentor Tutorials](#)
- Author Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- Version Release: @VER@
- Copyright 2002-2008 Gregory Beaver
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- TODO CS cleanup - change package to PhpDocumentor
- TODO CS cleanup - change classname to PhpDocumentor\_\*
- License [LGPL](#)

string function parserCData::Convert(&\$c, [\$postprocess = true]) [line 84]

**Function Parameters:**

- [Converter](#) &\$c the output converter
- bool \$postprocess if postprocessing is needed

**calls the output conversion**

- TODO CS cleanup - rename to convert for camelCase rule
- Uses Converter::getCData() - convert contents to text

# Class parserEntity

[line 479]

a standard entity like &rdquo;

This class is designed to represent all DocBook entities.

- **Package** phpDocumentor
- **Sub-Package** Tutorial
- **Tutorial** [phpDocumentor Tutorials](#)
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.2
- **License** [GPL](#)

Constructor **void** function parserEntity::parserEntity(**\$name**) [line 486]

**Function Parameters:**

- **string** **\$name** entity name

**sets up the entity**

**string** function parserEntity::Convert(&**\$c**, [**\$postprocess** = true]) [line 501]

**Function Parameters:**

- [\*\*Converter\*\*](#) &**\$c** the output converter
- **bool** **\$postprocess** if postprocessing is needed

**calls the output conversion**

- **TODO** CS cleanup - rename to convert for camelCase rule
- **Uses** [Converter::TranslateEntity\(\)](#) - convert contents to text

## Class parserXMLDocBookTag [line 122]

### a standard XML DocBook Tag

This class is designed to represent all DocBook tags. It is intelligent enough to understand the <title> tag, and also the <refname> tag for as title for <refentry>

- **Package** phpDocumentor
- **Sub-Package** Tutorial
- **Tutorial** [phpDocumentor Tutorials](#)
- **Author** Gregory Beaver < [celflog@php.net](mailto:celflog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2008 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **TODO** CS cleanup - rename to parserXmlDocBookTag for camelCase rule
- **Since** 1.2
- **License** [GPL](#)

### parserXMLDocBookTag::\$attributes

array = array() [line 130]

### Attributes from the XML tag

Format: array(attrname => attrvalue, attrname => attrvalue,...)

### parserXMLDocBookTag::\$name

*string* [line 135]

## Name of the tag

Constructor *void* function parserXMLDocBookTag::parserXMLDocBookTag(*\$name*) [line 165]

### Function Parameters:

- *string* **\$name** tag name

## sets up the tag

- **TODO** CS cleanup - rename to parserXmlDocBookTag for camelCase rule

*mixed* function parserXMLDocBookTag::add(*\$el*) [line 443]

### Function Parameters:

- [parserEntity](#)|[parserCData](#)|[parserXMLDocBookTag](#)|*string* **\$el** nested tag, entity, or text

## Add contents to this tag.

There are four kinds of data in a DocBook tutorial:

1. **tags** - normal tags like <refentry>
  2. **entities** - normal entities like &rdquo;
  3. **<![CDATA[** - character data that should not be interpreted, like <programlisting> contents
  4. **text** - normal non-markup text
- All four kinds of data are added here

*void* function parserXMLDocBookTag::addAttribute(*\$name*, *\$value*) [line 271]

### Function Parameters:

- *string* **\$name** attribute name
- *string*|[parserId](#)|[inlineTag](#) **\$value** value of attribute

**Add an xml tag attribute name="value" pair**  
if the attribute is id, value must be a [parserIdInlineTag](#)

*void* function parserXMLDocBookTag::addCData(\$word) [line 256]  
**Function Parameters:**

- *string* \$word word to add

### **add a word to CData**

*string* function parserXMLDocBookTag::Convert(&\$c, [\$postprocess = true]) [line 180]  
**Function Parameters:**

- [Converter](#) &\$c the output converter
- *bool* \$postprocess if postprocessing is needed

### **calls the output conversion**

- **Uses [Converter::TranslateTag\(\)](#)** - Calls this to enclose the contents of the DocBook tag based on the values in template options.ini file

*void* function parserXMLDocBookTag::endCData() [line 211]

### **Adds \$\_cdata to \$value**

*string* function parserXMLDocBookTag::getId(&\$c) [line 323]

**Function Parameters:**

- [Converter](#) &\$c the output converter

**Return converter-specific formatting of ID.**  
Passes \$c to [parserIdInlineTag::Convert\(\)](#)

*bool|string* function parserXMLDocBookTag::getSubsection(&\$c, \$subsection) [line 401]

**Function Parameters:**

- [Converter](#) &\$c the output converter
- *string* \$subsection converter-specific subsection

### Retrieve the contents of a subsection

This method uses the \$\_id members of nested docbook tags to retrieve the section defined by \$subsection

*string* function parserXMLDocBookTag::getTitle(&\$c) [line 347]

**Function Parameters:**

- [Converter](#) &\$c the output converter

### Retrieve Converter-specific formatting of the title of this element

*int* function parserXMLDocBookTag::getTOC([\$state = false]) [line 228]

**Function Parameters:**

- *false/integer* \$state either an index of the {@toc} tag in \$this->value  
false, if the next index value of \$this->value is needed or

### Retrieve either the table of contents index, or the location that the TOC will go

- See [parserXMLDocBookTag::setTOC\(\)](#)

*boolean* function parserXMLDocBookTag::hasTitle() [line 335]

### Determine whether the docbook element has a title

*void* function parserXMLDocBookTag::setId(\$id) [line 309]

**Function Parameters:**

- [\*parserIdInlineTag\*](#) **\$id** the id value

**If the id attribute is present, this method will set its id**

*void function parserXMLDocBookTag::setTitle(\$title) [/line 297]*

**Function Parameters:**

- [\*parserXMLDocBookTag\*](#) **\$title** the title element

**Set the title of a DocBook tag section.**

For most DocBook tags, the title is represented with a <title></title> tag pair. The <refentry> top-level tag is a little different. Instead of using <title></title>, phpDocumentor uses the contents of the <refname> tag in the <refnamediv> tag

*void function parserXMLDocBookTag::setTOC(\$state, \$val) [/line 244]*

**Function Parameters:**

- *integer \$state* index of the TOC in \$this->value
- [\*parserTocInlineTag\*](#) **\$val** tag value

**sets the TOC value**

*void function parserXMLDocBookTag::startCData() [/line 201]*

**Begin a new CData section**

- See [\*parserXMLDocBookTag::addCData\(\)\*](#)

**phpDocumentorTWordParser.inc**  
**tokenizer extension-based lexer for PHP code**  
phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2002-2006 Gregory Beaver

LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** Parsers
- **Author** Gregory Beaver < [celfog@php.net](mailto:celfog@php.net)>
- **Version** CVS: \$Id: phpDocumentorTWordParser.inc 287887 2009-08-30 06:10:55Z ashnazg \$
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - PHPCS needs to ignore CVS Id length
- **Since** 1.2
- **License** [LGPL](#)

# WordParser.inc

## a generic lexer

phpDocumentor :: automatic documentation generator

PHP versions 4 and 5

Copyright (c) 2000-2007 Joshua Eichorn

### LICENSE:

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

- **Package** phpDocumentor
- **Sub-Package** WordParsers
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** CVS: \$Id: WordParser.inc 246145 2007-11-14 01:37:03Z ashnazg \$
- **Copyright** 2000-2007 Joshua Eichorn
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **Since** 0.1
- **License** [LGPL](#)

## Class ObjectWordParser *[line 130]*

Like WordParser but designed to handle an array with strings and

## [parserInlineTags](#)

- **Package** phpDocumentor
- **Sub-Package** WordParsers
- **Author** Greg Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2000-2007 Joshua Eichorn
- **Since** 1.2
- **Usedby** [XMLPackagePageParser::XMLPackagePageParser\(\)](#)

Constructor `void function ObjectWordParser::ObjectWordParser([\$casesensitive = false])` [line 138]

**Function Parameters:**

- **\$casesensitive**

`void function ObjectWordParser::getWord()` [line 166]

`boolean function ObjectWordParser::nextIsObjectOrNonNL()` [line 281]

**Determine if the next word is an inline tag**

`void function ObjectWordParser::setup(&\$input)` [line 149]

**Function Parameters:**

- **array &\\$input** [parserStringWithInlineTags::\\$value](#) style-array, with alternating text and inline tags

**Set the word parser to go.**

## Class phpDocumentorTWordParser

[line 59]

**Like WordParser, but expects an array of tokens from the tokenizer instead of a string.**

- **Package** phpDocumentor
- **Sub-Package** WordParsers
- **Author** Gregory Beaver < [cellog@php.net](mailto:cellog@php.net)>
- **Version** Release: @VER@
- **Copyright** 2002-2007 Gregory Beaver
- **Link** <http://pear.php.net/PhpDocumentor>
- **Link** <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **TODO** CS cleanup - change classname to PhpDocumentor\_\*
- **Since** 1.2
- **License** [LGPL](#)

*void* function phpDocumentorTWordParser::addFileSource(\$word) [line 278]

**Function Parameters:**

- **array \$word** full file source code

**Wrapper for[addSource\(\)](#) used to retrieve the entire source code organized by line number in setup()**

*void* function phpDocumentorTWordParser::addSource(\$word, [\$file = false]) [line 300]

**Function Parameters:**

- **array/string \$word** token to add
- **bool \$file** true if this should be added to \$file\_source

**Generate source token arrays organized by line number**

This code will split up tokens that contain "\n" and add them to the source code as separate tokens on different lines.

- **Uses \_set\_sars()**

*int/void* function phpDocumentorTWordParser::backupPos() [line 361]

## **backs the parser up to the previous position**

*string* function `phpDocumentorTWordParser::concatTokens($a)` [line 193]

### **Function Parameters:**

- `array $a` array of tokens

## **Utility function to convert a series of tokens into a string**

- **Static**

*void* function `phpDocumentorTWordParser::findGlobal($tokens)` [line 347]

### **Function Parameters:**

- `array $tokens` tokens that represent the global variable definition

## **Tell the `phpDocumentorTWordParser` to return the entire global variable if it is found.**

- **Uses** `$_global_search`

*array* function `phpDocumentorTWordParser::getFileSource()` [line 148]

## **gets the source code tokens**

- **Used by** [parserExampleTag::parserExampleTag\(\)](#) - uses to parse an example and retrieve all tokens by line number

*array* function `phpDocumentorTWordParser::getSource()` [line 135]  
**loads up next set of source code**

*string/array* function `phpDocumentorTWordParser::getWord()` [line 218]  
**Retrieve a token for the phpDocumentorTParser**

*void* function `phpDocumentorTWordParser::setup(&$input)` [line 117]  
**Function Parameters:**

- *string &\$input* source code

Uses [http://www.php.net/token\\_get](http://www.php.net/token_get) to tokenize the source code.

*bool* function `phpDocumentorTWordParser::tokenEquals($a, $b)` [line 178]  
**Function Parameters:**

- *mixed \$a* first token
- *mixed \$b* second token

**Utility function to determine whether two tokens from the tokenizer are equal**

- **Static**

## Class WordParser [line 56]

**Retrieves tokens from source code for use by the Parser**

- **Package** phpDocumentor
- **Sub-Package** WordParsers
- **Author** Joshua Eichorn < [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)>
- **Version** Release: @VER@
- **Copyright** 2000-2007 Joshua Eichorn
- See [Parser](#)
- Link <http://pear.php.net/PhpDocumentor>
- Link <http://www.phpdoc.org>
- **TODO** CS cleanup - change package to PhpDocumentor
- **License** [LGPL](#)

*void* function WordParser::backupPos(\$word) [line 346]

**Function Parameters:**

- *string* **\$word** token to back up to

**Backup to the previous token so that it can be retrieved again in a new context.**

Occasionally, a word will be passed to an event handler that should be handled by another event handler. This method allows that to happen.

*string* function WordParser::getBlock(\$start, \$len) [line 305]

**Function Parameters:**

- *integer* **\$start** starting position
- *integer* **\$len** length of block to retrieve

**Unused**

```
1  function getBlock($start $len)
2  {
3      return substr($this>    data,$start $len);
4  }
```

*int* function WordParser::getPos() [line 290]

**Returns the current pointer position, or 1 character after the end of the word**

*string* function WordParser::getSource() [line 149]

## Retrieve source code for the last function/method

*string/false* function WordParser::getWord() [line 206]

### Retrieve a token from the token list

The [Parser](#) class relies upon this method to retrieve the next token. The \$wordseparators array is a collection of strings that delineate tokens for the current parser state. \$wordseparators is set by the parser with a call to [Parser::configWordParser\(\)](#) every time a new parser state is reached.

For example, while parsing the source code for a class, the word

```
1 var  
is a token, and  
1 global
```

is not, but inside a function, the reverse is true. The parser state PARSER\_STATE\_CLASS has a token list that includes whitespace, code delimiters like ; and {}, and comment/DocBlock indicators

If the whitespace option has been turned off using [setWhitespace\(\)](#), then no whitespace is returned with tokens

{@ is of course the string containing the PHP code to be parsed, and \$pos is the cursor, or current location within the parsed data. }}}

*void* function WordParser::setPos(\$pos) [line 330]

### Function Parameters:

- *integer \$pos* the position

## Set the internal cursor within the source code

*void* function WordParser::setSeparator(&\$seps) [line 318]

### Function Parameters:

- *array &\$seps* array of strings that separate tokens

## Sets the list of possible separator tokens

- **Uses \$wordseparators**

*void* function WordParser::setup(&\$input) [/line 132]

**Function Parameters:**

- *string &\$input* source code

## Initialize the WordParser

*void* function WordParser::setWhitespace([&\$val = false]) [/line 360]

**Function Parameters:**

- *boolean \$val* flag to return or strip whitespace

## set parser to return or strip whitespace

# Appendices

# Appendix A - Class Trees

## Package phpDocumentor

### abstractLink

- [abstractLink](#)
  - [classLink](#)
  - [defineLink](#)
  - [functionLink](#)
  - [globalLink](#)
  - [methodLink](#)
    - [constLink](#)
    - [varLink](#)
  - [pageLink](#)
  - [tutorialLink](#)

### bug\_772441

- [bug\\_772441](#)

### Classes

- [Classes](#)

### ErrorTracker

- [ErrorTracker](#)

## EventStack

- [EventStack](#)

## Io

- [Io](#)

## parserBase

- [parserBase](#)
    - [parserElement](#)
      - [parserClass](#)
      - [parserConst](#)
      - [parserDefine](#)
      - [parserFunction](#)
        - [parserMethod](#)
    - [parserGlobal](#)
    - [parserInclude](#)
    - [parserVar](#)
  - [parserInlineTag](#)
    - [parserIdInlineTag](#)
    - [parserInheritdocInlineTag](#)
    - [parserLinkInlineTag](#)
      - [parserTutorialInlineTag](#)
  - [parserSourceInlineTag](#)
    - [parserExampleInlineTag](#)
  - [parserTocInlineTag](#)
- [parserStringWithInlineTags](#)
  - [parserB](#)
  - [parserBr](#)
  - [parserCData](#)
  - [parserCode](#)
  - [parserDesc](#)
  - [parserDescVar](#)
  - [parserI](#)
  - [parserKbd](#)

- [parserList](#)
- [parserPackagePage](#)
  - [parserTutorial](#)
- [parserPre](#)
- [parserSamp](#)
- [parserTag](#)
  - [parserAccessTag](#)
  - [parserFileSourceTag](#)
    - [parserExampleTag](#)
- [parserLinkTag](#)
  - [parserLicenseTag](#)
  - [parserSeeTag](#)
    - [parserTutorialTag](#)
    - [parserUsesTag](#)
      - [parserUsedByTag](#)
- [parserNameTag](#)
- [parserReturnTag](#)
  - [parserPropertyTag](#)
    - [parserMethodTag](#)
    - [parserPropertyReadTag](#)
    - [parserPropertyWriteTag](#)
  - [parserVarTag](#)
    - [parserParamTag](#)
      - [parserStaticvarTag](#)
- [parserXMLDocBookTag](#)

## parserData

- [parserData](#)

## parserDocBlock

- [parserDocBlock](#)

## parserEntity

- [parserEntity](#)

## parserPage

- [parserPage](#)

## phpDocumentor\_IntermediateParser

- [phpDocumentor\\_IntermediateParser](#)

## phpDocumentor\_peardoc2\_XML\_Beautifier

- XML\_Beautifier
  - [phpDocumentor\\_peardoc2\\_XML\\_Beautifier](#)

## phpDocumentor\_setup

- [phpDocumentor\\_setup](#)

## phpDocumentor\_XML\_Beautifier\_Tokenizer

- XML\_Beautifier\_Tokenizer
  - [phpDocumentor\\_XML\\_Beautifier\\_Tokenizer](#)

## ProceduralPages

- [ProceduralPages](#)

## Publisher

- [Publisher](#)
  - [Parser](#)
    - [parserDescParser](#)
    - [PDFParser](#)
    - [phpDocumentorTParser](#)
      - [phpDocumentor\\_HighlightParser](#)
    - [phpDocumentor\\_TutorialHighlightParser](#)
    - [ppageParser](#)
    - [XMLPackagePageParser](#)

## RecordWarning

- [RecordWarning](#)
  - [RecordError](#)

## tests\_HighlightParserTests

- [tests\\_HighlightParserTests](#)

## tests\_IntermediateParserTests

- [tests\\_IntermediateParserTests](#)

## tests\_ParserClassTests

- [tests\\_ParserClassTests](#)

## tests\_ParserPageTests

- [tests\\_ParserPageTests](#)

## tests\_phpDocumentorSetupTests

- [tests\\_phpDocumentorSetupTests](#)

## tests\_phpDocumentorTParserTests

- [tests\\_phpDocumentorTParserTests](#)

## WordParser

- [WordParser](#)
  - [ObjectWordParser](#)
  - [phpDocumentorTWordParser](#)
    - [phpDocumentor\\_HighlightWordParser](#)

## Package HTML\_TreeMenu

### HTML\_TreeMenu

- [HTML\\_TreeMenu](#)

### HTML\_TreeMenu\_Presentation

- [HTML\\_TreeMenu\\_Presentation](#)
  - [HTML\\_TreeMenu\\_DHTML](#)
  - [HTML\\_TreeMenu\\_Listbox](#)

### HTML\_TreeNode

- [HTML\\_TreeNode](#)
  - [DirNode](#)

## Package tests

### a

- [a](#)

a

- [a](#)

b553607\_Parser

- [b553607 Parser](#)

brokenlinkstovars

- [brokenlinkstovars](#)

bug540341

- [bug540341](#)

bug557861

- [bug557861](#)

bug\_489398

- [bug\\_489398](#)

bug\_556894\_base

- [bug\\_556894\\_base](#)
  - [bug\\_556894\\_sub1](#)
  - [bug\\_556894\\_sub2](#)

## childofpriv

- priv
  - [childofpriv](#)

## ClubBase

- PEAR
  - [ClubBase](#)

## ctest

- [ctest](#)

## few

- [few](#)

## functionincomment

- [functionincomment](#)

## iConverter

- [iConverter](#)
  - [iHTMLConverter](#)

## iiparserBase

- [iiparserBase](#)

## iNewRender

- [iNewRender](#)

## iParser

- [iParser](#)

## iparserElement

- [iparserElement](#)

## mama

- [mama](#)
  - [baby](#)

## metoo

- [metoo](#)

## multipl

- [multipl](#)

## notseen

- [notseen](#)

## parent\_b587733

- [parent\\_b587733](#)
  - [kiddie\\_b587733](#)

## priv1

- [priv1](#)

## RecordWarning

- [RecordWarning](#)

## summary\_form

- [summary\\_form](#)

## test

- [test](#)

## test2

- [test2](#)

## testarraybug

- [testarraybug](#)

## testClass

- [testClass](#)

## testme

- [testme](#)

## tests\_HighlightParserGetInlineTagsTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_HighlightParserGetInlineTagsTests](#)

## tests\_IntermediateParserAddPrivatePageTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_IntermediateParserAddPrivatePageTests](#)

## tests\_ParserClassGetSourceLocationTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_ParserClassGetSourceLocationTests](#)

## tests\_ParserPageGetSourceLocationTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_ParserPageGetSourceLocationTests](#)

## tests\_phpDocumentorSetupCleanConverterNamePieceTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_phpDocumentorSetupCleanConverterNamePieceTests](#)

## tests\_phpDocumentorSetupDecideOnOrOffTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_phpDocumentorSetupDecideOnOrOffTests](#)

## tests\_phpDocumentorTParserGetInlineTagsTests

- PHPUnit\_Framework\_TestCase
  - [tests\\_phpDocumentorTParserGetInlineTagsTests](#)

## test\_541886

- [test\\_541886](#)

Package org-phpdoc

Package Converters

## Converter

- [Converter](#)
  - [CHMdefaultConverter](#)
  - [HTMLframesConverter](#)
  - [HTMLSmartyConverter](#)
  - [PDFdefaultConverter](#)
  - [XMDocBookConverter](#)
  - [XMDocBookpeardoc2Converter](#)

## PDFParser

- [Parser \(Different package\)](#)
  - [PDFParser](#)

## phpdocpdf

- [Cezpdf \(Different package\)](#)
  - [phpdocpdf](#)

## Package XML\_Beautifier

## PHPDoc\_XML\_Beautifier\_Renderer\_Plain

- XML\_Beautifier\_Renderer
  - [PHPDoc\\_XML\\_Beautifier\\_Renderer\\_Plain](#)

## Package Cpdf

## Cpdf

- [Cpdf](#)
  - [Cezpdf](#)
    - [phpdocpdf](#)

## Package Smarty

### Config\_File

- [Config\\_File](#)

### Smarty

- [Smarty](#)
  - [Smarty\\_Compiler](#)

# Appendix B -

## README/CHANGELOG/INSTALL

# README

\$Id: README 242918 2007-09-26 02:57:11Z ashnazg \$

#####
phpDocumentor

#####
Installation

#####

Installation for phpDocumentor is as simple as installing PHP and a web server.  
If you need help installing a webserver or php, refer to the documentation that  
comes with the webserver or with php. phpDocumentor also runs from the  
command-line, and only requires that PHP CLI version be installed.

All users with PEAR should either install this version directly from the  
package.xml file included in the distribution, or install via  
"pear install PhpDocumentor."

The command-line interface "phpdoc" will be in the path, you can use it  
immediately on both windows and unix. The web interface is in a subdirectory of  
your document root, be sure to "pear config-set data\_dir /path/to/my/htdocs"  
prior to installation.

To use phpDocumentor as a command-line tool in \*nix-based systems, simply run  
the phpdoc script. In windows, run  
"c:\php\php.exe phpdoc" where C:\php is the path to the cli version of php.  
The phpdoc command-line interface will not run with the ISAPI module of PHP.  
To see the command line options, use phpdoc -h or read at the bottom of this  
README.

There is a BASH shell script that you can put inside your project that can save  
time writing command-line options of phpdoc. Simply copy makedoc.sh to your  
project path and edit it. When you wish to rebuild the documentation, run it again.  
Note that .ini scripts are far easier to edit and are supported via "phpdoc -c file.ini"

To use phpDocumentor's web interface, install the files into a subdirectory of  
your document root (read the webserver's documentation for more information if you  
don't understand how to install into a subdirectory of your document root).  
Then, browse to the web page as you would to any other web page. If you encounter  
problems, make sure that the web server has write access permissions to the output  
directory that you choose. Again, your web server documentation has detailed  
information on how to do this. The command-line interface will not have problems  
with access.

If you need more detailed help, check out INSTALL.

#####
Release Notes

#####

Release notes for the newest version are in Release-@VER@.

@VER@ is out, a @STABILITY@ release.

To learn how to write PHPDoc-style documentation, run phpDocumentor on itself, with ./phpdoc -c makedocs. Before you do, modify users/makedocs.ini to contain the proper path values.

If you find a bug please post it at:

[http://pear.php.net/bugs/search.php?cmd=display&package\\_name%5B%5D=PhpDocumentor](http://pear.php.net/bugs/search.php?cmd=display&package_name%5B%5D=PhpDocumentor)

#####
General Notes
#####

#####
phpDocumentor is hosted at [phpdoc.org](http://phpdoc.org), with downloads and mailing list through sourceforge.net, and primary support at PEAR.

webpage: <http://phpdoc.org/>  
documentation: <http://phpdoc.org/manual.php>  
mirror: <http://phpdoc.sourceforge.net>  
projectpage: <http://www.phpdoc.org>  
PEAR: <http://pear.php.net/package/PhpDocumentor>

For a list of people who have helped work on this project please read the Authors file.

phpDocumentor @VER@ is released under the LGPL version 2.1, text is in LICENSE.

#####
Feature Notes
#####

#####
phpDocumentor is a JavaDoc-like automatic documentation generator for PHP, written in PHP. It is the most versatile tool for documenting PHP.

For users new to phpDocumentor, phpDocumentor uses an event-driven parser and intermediary data structures that allow it to perform at a level other automatic documentors for PHP cannot achieve:

- parsing any php file, with multiple classes and functions in the same file
- fully compliant with even the most esoteric php syntax  
  (\$string = <<< EOF, for example)
- ability to generate multiple output formats
- extremely fast parsing
- error/warnings by line number and file to allow for debugging of documentation tags
- multiple output templates to easily customize the look of generated documentation
- extensive documentation of the package, including a detailed specification for tags and templates
- LGPL License

New since version 1.3.0:

- PHP 5 Parsing
- hundreds of bugfixes from version 1.2.3

If something is missing from this list make sure sure to file a feature request at PEAR:

[http://pear.php.net/bugs/search.php?cmd=display&package\\_name%5B%5D=PhpDocumentor](http://pear.php.net/bugs/search.php?cmd=display&package_name%5B%5D=PhpDocumentor)

If you want to help code that new feature and get it just right, please let us know.  
Any help is greatly appreciated, just contact the  
phpDocumentor team for information on getting started.

phpDocumentor needs php 4.1.0 or greater, but you'll see the best performance  
in the newest version. The recommended version for phpDocumentor 1.3.0 and up  
is PHP 5.1.3 or newer.

```
#####
Command Line notes
#####
```

Read the documentation at <http://www.phpdoc.org/docs> for the most up-to-date  
command-line information.

You can also generate documentation from the source using:

```
phpdoc -c makedocs
```

Run this command from the installation directory, and make sure you have full  
write and directory creation permissions or it will not work.

Generated documentation is accessible via Documentation/new/index.html

```
#####
```

If you run phpdoc and get :  
bash: ./phpdoc: No such file or directory

Then you haven't installed the cgi version of php.

Go to your php src dir and try  
make clean  
.configure  
make  
make install

phpdoc should work now.

If you're using php 4.2.0 or higher you will want to use the cli version  
instead of the cgi. Checkout [php.net](http://php.net) for details on these changes

```
#####
Web Interface notes
#####
Put phpdoc.php together with the *.inc files someplace on your webserver.
NEVER USE THE WEB INTERFACE ON A PRODUCTION WEB SERVER. Allowing your server
to write files to disk is a serious security risk, and phpDocumentor is not
designed to work on insecure systems. Setup php on a development machine
that has a firewall or no internet connection, and run phpDocumentor from there.
```

Make sure your webserver can write to wherever you specify as a target or you  
will get lots of errors.

```
#####
Thanks
#####
Thanks to Sam Blum for his assistance with @example and other enhancements.
Thanks to William K. Hardeman for his HTML:Smarty:HandS design.
```

Thanks to Marco von Ballmoos for transforming the HTML:frames converter and templates into a magnificent work of art.

Thanks to Andrew Eddie for docbuilder and the CHM Converter enhancements.

Thanks to Darren Cook for suggesting exciting new features, and making sure that they worked when implemented.

Thanks to Marko Kaening, Dan Convisser for policing the source and finding bugs that would have otherwise gone undetected.

Thanks to Juan Pablo Morales for the web interface.

Thanks to whoever sent me the patch to make phpDocumentor work better in NT... I have your diff and patched the program but i seem to have lost your name... if you send it to me i'll add it to the Authors file.

Thanks to Florian Clever for the newest set of win32 patches... they seem to have fixed the last of the problems.

Thanks to EVERYONE who has provided ideas and input, without you, phpDocumentor would be nothing.

#####
#####  
If you have any questions please try our mailing list at  
[phpdocu-general@sourceforge.net](mailto:phpdocu-general@sourceforge.net).

joshua eichorn [jeichorn@phpdoc.org](mailto:jeichorn@phpdoc.org)  
gregory beaver [cellog@php.net](mailto:cellog@php.net)  
chuck burgess [ashnazg@php.net](mailto:ashnazg@php.net)

vim: set expandtab:

## ChangeLog

\$Id: ChangeLog 229303 2007-02-07 20:06:44Z ashnazg \$

See package.xml for changelogs for all releases.

Release details are under the <package><changelog> section, in the various <release> tags. Specific changelog info for each release is under each <release><notes> tag.

vim: set expandtab:

## INSTALL

\$Id: INSTALL 256227 2008-03-28 02:27:12Z ashnazg \$

phpDocumentor Installation instructions

---

phpDocumentor is very easy to install. The hardest part is installing PHP and possibly a Web Server, if you plan to use the web interface phpdoc.php.

To install phpDocumentor, use a compression program to decompress the contents to a directory, preserving internal directory structure of the archive. If you wish to use the web interface, you must unzip the archive into a subdirectory accessible by the web server.

**NOTE:**

There have been problems with the non GNU tar not being able to extracts some of the long directory paths properly. The simplest solution for BSD users is to download the zip version of phpDocumentor.

**PEAR installation**

---

This is by far the easiest way to install phpDocumentor. Included in this release is a package.xml file.

If you plan to use the web interface, you must run this command prior to installation:

```
pear config-set data_dir /path/to/public_html/pear
```

The line above assumes that /path/to/public\_html is the location of index.html if you browse to <http://localhost/index.html>

To install or upgrade phpDocumentor, simply run

```
pear upgrade PhpDocumentor
```

To use the web interface, browse to <http://localhost/pear/PhpDocumentor/index.html>

The cli interface is installed in the pear bin\_dir, you can find this directory by running:  
pear config-show

**Command-line interface installation and usage instructions**

---

To use phpDocumentor as a command-line interface, you need to install PHP version 4.1.0 or greater.

\*IMPORTANT\* any version of PHP below 4.1.0 will not work with version 1.3.0 or later of phpDocumentor!!! To use the advanced tokenizer-based parser, you must have php 4.3.0

**Windows**

---

You need the cli version of PHP (php-cli.exe or cli/php.exe in 4.3.0+). Either run phpDocumentor from the directory that PHP resides in, or put php.exe in your DOS PATH environment variable. The simplest usage of phpDocumentor is:

```
C:\>php.exe "C:\Path\To\phpdoc" -t targetdir -o HTML:default:default -d parsedir
```

where targetdir is the directory you wish phpDocumentor to create output, and parsedir is the directory you wish to parse.

**Unix**

---

Make sure that the cgi/cli version of PHP is in your path. The simplest usage of phpDocumentor is:

```
phpdoc -t targetdir -o HTML:default:default -d parsedir
```

You can also use makedoc.sh. Simply copy it to your project path and edit it. When you wish to (re)build your project's documentation, simply run it.

#### Web interface installation and usage instructions

---

To use phpDocumentor as a web interface, you need to install a Web server and configure it, as well as install PHP version 4.1.0 or greater. Consult the documentation of your webserver and of PHP for installation support.

#### Windows and Unix

---

Open a web browser and browse to index.html at the location that you set up. Instructions are in the web interface.

vim: set expandtab :

## FAQ

---

### ##### phpDocumentor Frequently Asked Questions

---

#### ##### Introduction

---

Before we start with questions and their answers, make sure to read the documentation \*thoroughly\* found at <http://www.phpdoc.org/manual.php>. The complexity of phpDocumentor can be a bit impenetrable until you understand the logic behind it.

If you have read the documentation thoroughly, and have read this FAQ and you are still lost, please go directly to the bug database at [http://sourceforge.net/tracker/?group\\_id=11194&atid=111194](http://sourceforge.net/tracker/?group_id=11194&atid=111194) and read through the open bugs. If you don't find the solution to your problem (or proof that it at least is not your fault), then go to the help forum and post a help message at [http://sourceforge.net/forum/forum.php?forum\\_id=35065](http://sourceforge.net/forum/forum.php?forum_id=35065).

---

#### ##### All Questions (Table of Contents)

---

#### ##### Installation

Q: There is no package.xml in the release, where is it?

#### ##### Documentation Errors

Q: I keep getting an illegal package tag error for the first DocBlock in a file, isn't this a page-level DocBlock?

Q: I keep getting a warning that there is no @package tag in file xxx.php, but there is a @package tag in the class definition, doesn't this apply to the file?

#### ##### Feature Requests

Q: Can you implement the @example tag inline so that I can put PHP source code examples directly in DocBlocks?

#### ##### Crash/Ssegfaults

Q: phpDocumentor crashes if I use \_\_FUNCTION\_\_!!!

```
#####
# Installation
#####
```

Q: There is no package.xml in the release, where is it?

A: this problem occurs when one does the faulty steps of:

```
$ tar xvf PhpDocumentor-1.3.1.tgz
$ pear install package.xml
```

instead, the user should simply run:

```
$ pear install PhpDocumentor-1.3.1.tgz
```

or, if the zlib extension is not enabled:

```
$ gunzip PhpDocumentor-1.3.1.tgz
$ pear install PhpDocumentor-1.3.1.tar
```

```
#####
# Documentation Errors
#####
```

Q: I keep getting an illegal package tag error for the first DocBlock in a file,  
isn't this a page-level DocBlock?

---

```
---[[UPDATE]]
---VERSION 1.2.2 has a different page-level docblock recognition algorithm
---Now, the first docblock in a file is a page-level docblock if it contains
---a @package tag.
```

---

A: Please read the documentation very carefully. A page-level DocBlock does  
occur at the start of a file, but the first DocBlock is not always a  
page-level DocBlock! Why is this? Very often, you will want to document  
the entire page, or describe it (this page contains functions for blah), and  
also document the first item in a page separately. An example:

```
<?php
/*
 * This file contains all foobar functions and defines
 * @package mypackage
 */
/*
 * controls parsing of bar information
 */
define('parse_bar',true);
?>
```

The page has its own information, and the define has its own information.  
An example of what not to do:

```
<?php
/*
 * This file contains all foobar functions and defines
 * @package mypackage
*/
```

```
*/  
define('parse_bar',true);  
?>
```

Here, the DocBlock belongs to the define and not to the page! phpDocumentor can successfully parse this DocBlock, but it will apply the comment to the documentation of define('parse\_bar',true); and not to the page. Therefore, it warns you that your @package tag will be ignored because defines may not contain a @package tag.

Q: I keep getting a warning that there is no @package tag in file xxx.php, but there is a @package tag in the class definition, doesn't this apply to the file?

A: No. This example does not have a page-level DocBlock:

```
<?php  
/**  
 * This class is in package mypackage  
 * @package mypackage  
 */  
class in_mypackage {...}  
?>
```

phpDocumentor therefore assumes the page-level package is the same as the class package, mypackage. This is fine in most cases, but if multiple classes are in the same file with different packages, as in:

```
<?php  
/**  
 * This class is in package mypackage  
 * @package mypackage  
 */  
class in_mypackage {...}  
/**  
 * This class is in package anotherpackage  
 * @package anotherpackage  
 */  
class in_anotherpackage {...}  
?>
```

There is no way to determine which package the page should belong to, and phpDocumentor will automatically put it in the default package. This can cause incredible headaches debugging, and so we have added a warning in the 1.2.0 series that informs you if the package is inferred by phpDocumentor. To fix this warning, simply place a page-level DocBlock with a @package tag like:

```
<?php  
/**  
 * This file contains two packages  
 * @package filepackage  
 */  
/**  
 * This class is in package mypackage  
 * @package mypackage  
 */  
class in_mypackage {...}  
/**
```

```
* This class is in package anotherpackage
* @package anotherpackage
*/
class in_anotherpackage {...}
?>
#####
Feature Requests
#####

```

Q: Can you implement the @example tag inline so that I can put PHP source code examples directly in DocBlocks?

A: This is implemented using the HTML <code></code> tags as in:

```
/*
 * Short description
 *
 * Start of long description, here is a code example:
 * <code>
 * $my_phpcode = "easy to explain";
 * </code>
 * More text
 * <code>
 * define('morecode',true);
 * </code>
 */
#####
Crash/Segfaults
#####

```

Q: phpDocumentor crashes if I use \_\_FUNCTION\_\_!!!

A: This is caused by a known bug in all PHP versions prior to 4.3.2. It was fixed in PHP 4.3.2RC1, you must upgrade to PHP 4.3.2 if you use \_\_FUNCTION\_\_ in code that you wish to document (sorry!) or apply the bugfix patch to the tokenizer extension and recompile php (see the php.internals archive at php.net for help)

# Appendix C - Source Code

# Package phpDocumentor

# File Source for Converter.inc

Documentation for this file is available at [Converter.inc](#)

```
1  <?php
2  /**
3  * Base class for all Converters
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2001-2006 Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @package    Converters
29 * @author     Greg Beaver <cellog@php.net>
30 * @copyright  2001-2006 Gregory Beaver
31 * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
32 * @version    CVS: $Id: Converter.inc 287891 2009-08-30 08:08:00Z ashnazg $
33 * @filesource
34 * @link       http://www.phpdoc.org
35 * @link       http://pear.php.net/PhpDocumentor
36 * @see        parserDocBlock, parserInclude, parserPage, parserClass
37 * @see        parserDefine, parserFunction, parserMethod, parserVar
38 * @since      1.0rc1
39 */
40 /**
41 * Smarty template files
42 */
43 include_once("phpDocumentor/Smarty-2.6.0/libs/Smarty.class.php");
44 /**
45 * Base class for all output converters.
46 *
47 * The Converter marks the final stage in phpDocumentor. phpDocumentor works
48 * in this order:
49 *
50 * <pre>Parsing => Intermediate Parsing organization => Conversion to
51 * output</pre>
52 *
53 * A Converter takes output from the {@link phpDocumentor_IntermediateParser} and
54 * converts it to output. With version 1.2, phpDocumentor includes a variety
55 * of output converters:
56 * <ul>
57 *   <li> {@link HTMLframesConverter}</li>
58 *   <li> {@link HTMLSmartyConverter}</li>
59 *   <li> {@link PDFdefaultConverter}</li>
60 *   <li> {@link CHMdefaultConverter}</li>
61 *   <li> {@link CSVdia2codeConverter}</li>
62 *   <li> {@link XMLDocBookConverter}</li>
63 * </ul>
64 * {@internal
65 * The converter takes output directly from {@link phpDocumentor_IntermediateParser}
66 * and using {@link walk()} or {@link walk_everything} (depending on the value of
67 * {@link $sort_absolutely_everything}) it "walks" over an array of phpDocumentor
```

```

elements.}}}
67     *
68     * @package Converters
69     * @abstract
70     * @author Greg Beaver <cellog@php.net>
71     * @since 1.0rc1
72     * @version $Id: Converter.inc 287891 2009-08-30 08:08:00Z ashnazg $
73     */
74 class Converter
75 {
76     /**
77      * This converter knows about the new root tree processing
78      * In order to fix PEAR Bug #6389
79      * @var boolean
80      */
81     var $processSpecialRoots = false;
82     /**
83      * output format of this converter
84      *
85      * in Child converters, this will match the first part of the -o command-line
86      * as in -o HTML:frames:default "HTML"
87      * @tutorial phpDocumentor.howto.pkg#using.command-line.output
88      * @var string
89      */
90     var $outputformat = 'Generic';
91     /**
92      * package name currently being converted
93      * @var string
94      */
95     var $package = 'default';
96     /**
97      * subpackage name currently being converted
98      * @var string
99      */
100    var $subpackage = '';
101   /**
102    * set to a classname if currently parsing a class, false if not
103    * @var string/false
104    */
105    var $class = false;
106    /**#@+
107     * @access private
108     */
109    /**
110     * the workhorse of linking.
111     *
112     * This array is an array of link objects of format:
113     * [package][subpackage][eltype][elname] = descendant of {@link abstractLink}
114     * eltype can be page/function/define/class/method/var
115     * if eltype is method or var, the array format is:
116     * [package][subpackage][eltype][class][elname]
117     * @var array
118     * @see functionLink, pageLink, classLink, defineLink, methodLink, varLink, globalLink
119     */
120    var $links = array();
121    /**
122     * the workhorse of linking, with allowance for support of multiple
123     * elements in different files.
124     *
125     * This array is an array of link objects of format:
126     * [package][subpackage][eltype][file][elname] = descendant of {@link abstractLink}
127     * eltype can be function/define/class/method/var
128     * if eltype is method or var, the array format is:
129     * [package][subpackage][eltype][file][class][elname]
130     * @var array
131     * @see functionLink, pageLink, classLink, defineLink, methodLink, varLink, globalLink
132     */
133    var $linkswithfile = array();
134    /**#@-*/
135    /**
136     * set to value of -po commandline
137     * @tutorial phpDocumentor.howto.pkg#using.command-line.packageoutput
138     * @var mixed
139     */
140    var $package_output;
141    /**
142     * name of current page being converted
143     * @var string

```

```

146      */
147  var $page;
148
149 /**
150  * path of current page being converted
151  * @var string
152  */
153 var $path;
154
155 /**
156  * template for the procedural page currently being processed
157  * @var Smarty
158  */
159 var $page_data;
160
161 /**
162  * template for the class currently being processed
163  * @var Smarty
164  */
165 var $class_data;
166
167 /**
168  * current procedural page being processed
169  * @var parserPage
170  */
171 var $curpage;
172 /**
173  * alphabetical index of all elements sorted by package, subpackage, page,
174  * and class.
175  * @var array Format: array(package => array(subpackage => array('page' | 'class' =>
array(path|classname => array(element, element,...))))
176  * @uses $sort_absolutely_everything if true, then $package_elements is used,
177  * otherwise, the {@link ParserData::$classelements} and
178  * {@link ParserData::$pageelements} variables are used
179  */
180 var $package_elements = array();
181 /**
182  * alphabetical index of all elements
183  *
184  * @var array Format: array(first letter of element name => array( {@link parserElement}
or {@link parserPage},...))
185  * @see formatIndex(), HTMLframesConverter::formatIndex()
186  */
187 var $elements = array();
188 /**
189  * alphabetized index of procedural pages by package
190  *
191  * @see $leftindex
192  * @var array Format: array(package => array(subpackage => array( {@link pageLink}
1,{@link pageLink} 2,...)
193  */
194 var $page_elements = array();
195 /**
196  * alphabetized index of defines by package
197  *
198  * @see $leftindex
199  * @var array Format: array(package => array(subpackage => array( {@link defineLink}
1,{@link defineLink} 2,...)
200  */
201 var $define_elements = array();
202 /**
203  * alphabetized index of classes by package
204  *
205  * @see $leftindex
206  * @var array Format: array(package => array(subpackage => array( {@link classLink}
1,{@link classLink} 2,...)
207  */
208 var $class_elements = array();
209 /**
210  * alphabetized index of global variables by package
211  *
212  * @see $leftindex
213  * @var array Format: array(package => array(subpackage => array( {@link globalLink}
1,{@link globalLink} 2,...)
214  */
215 var $global_elements = array();
216 /**
217  * alphabetized index of functions by package
218  *
219  * @see $leftindex

```

```

220      * @var array Format: array(package => array(subpackage => array(           {@link functionLink}
1,{@link functionLink} 2,...)
221      */
222      var $function_elements = array();
223      /**
224      * alphabetical index of all elements, indexed by package/subpackage
225      *
226      * @var array Format: array(first letter of element name => array(   {@link parserElement}
or {@link parserPage},...))
227      * @see formatPkgIndex(), HTMLframesConverter::formatPkgIndex()
228      */
229      var $pkg_elements = array();
230
231      /**
232      * alphabetical index of all elements on a page by package/subpackage
233      *
234      * The page itself has a link under ###main
235      * @var array Format: array(package => array(subpackage => array(path =>
array({@link abstractLink} descendant 1, ...)))
236      * @see formatLeftIndex()
237      */
238      var $page_contents = array();
239
240      /**
241      * This determines whether the {@link $page_contents} array should be sorted by element
type as well as alphabetically by name
242      * @see sortPageContentsByElementType()
243      * @var boolean
244      */
245      var $sort_page_contents_by_type = false;
246      /**
247      * This is used if the content must be passed in the order it should be read, i.e. by
package, procedural then classes
248      *
249      * This fixes bug 637921, and is used by {@link PDFdefaultConverter}
250      */
251      var $sort_absolutely_everything = false;
252      /**
253      * alphabetical index of all methods and vars in a class by package/subpackage
254      *
255      * The class itself has a link under ###main
256      * @var array
257      * Format:<pre>
258      * array(package =>
259      *       array(subpackage =>
260      *             array(path =>
261      *                   array(class =>
262      *                         array({@link abstractLink} descendant 1, ...
263      *                           )
264      *                         )
265      *                   )
266      *                 )</pre>
267      * @see formatLeftIndex()
268      */
269      var $class_contents = array();
270      /**
271      * controls processing of elements marked private with @access private
272      *
273      * defaults to false. Set with command-line --parseprivate or -pp
274      * @var bool
275      */
276      var $parseprivate;
277      /**
278      * controls display of progress information while parsing.
279      *
280      * defaults to false. Set to true for cron jobs or other situations where no visual output
is necessary
281      * @var bool
282      */
283      var $quietmode;
284
285      /**
286      * directory that output is sent to. -t command-line sets this.
287      * @tutorial phpDocumentor.howto.pkg#using.command-line.target
288      */
289      var $targetDir = '';
290
291      /**
292      * Directory that the template is in, relative to phpDocumentor root directory
293      * @var string

```

```

294 /**
295  var $templateDir = '';
296
297 /**
298  * Directory that the smarty templates are in
299  * @var string
300  */
301  var $smarty_dir = '';
302
303 /**
304  * Name of the template, from last part of -o
305  * @tutorial phpDocumentor.howto.pkg#using.command-line.output
306  * @var string
307  */
308  var $templateName = '';
309
310 /**
311  * full path of the current file being converted
312  */
313  var $curfile;
314
315 /**
316  * All class information, organized by path, and by package
317  * @var Classes
318  */
319  var $classes;
320
321 /**
322  * Flag used to help converters determine whether to do special source highlighting
323  * @var boolean
324  */
325  var $highlightingSource = false;
326
327 /**
328  * Hierarchy of packages
329  *
330  * Every package that contains classes may have parent or child classes
331  * in other packages. In other words, this code is legal:
332  *
333  * <code>
334  * /**
335  *   * @package one
336  *   *
337  * class one {}
338  *
339  * /**
340  *   * @package two
341  *   *
342  * class two extends one {}
343  * </code>
344  *
345  * In this case, package one is a parent of package two
346  * @var array
347  * @see phpDocumentor_IntermediateParser::$package_parents
348  */
349  var $package_parents;
350
351 /**
352  * Packages associated with categories
353  *
354  * Used by the XML:DocBook/peardoc2 converter, and available to others, to
355  * group many packages into categories
356  * @see phpDocumentor_IntermediateParser::$packagecategories
357  * @var array
358  */
359  var $packagecategories;
360
361 /**
362  * All packages encountered in parsing
363  * @var array
364  * @see phpDocumentor_IntermediateParser::$all_packages
365  */
366  var $all_packages;
367
368 /**
369  * A list of files that have had source code generated
370  * @var array
371  */
372  var $sourcePaths = array();
373

```

```

374     /**
375      * Controls which of the one-element-only indexes are generated.
376      *
377      * Generation of these indexes for large packages is time-consuming. This is an
378      * optimization feature. An
379      * example of how to use this is in {@link HTMLframesConverter:::$leftindex}, and in {@link
380      * HTMLframesConverter:::formatLeftIndex()}.
381      * These indexes are intended for use as navigational aids through documentation, but can
382      * be used for anything by converters.
383      * @see $class_elements, $page_elements, $function_elements, $define_elements,
384      * $global_elements
385      * @see formatLeftIndex()
386      * @var array
387      */
388      var $leftindex = array('classes' => true, 'pages' => true, 'functions' => true,
389      'defines' => true, 'globals' => true);
390
391      /**
392      * @access private
393      * @var string
394      * @see phpDocumentor_IntermediateParser::$title
395      */
396      var $title = 'Generated Documentation';
397
398      /**
399      * Options for each template, parsed from the options.ini file in the template base
400      * directory
401      * @tutorial phpDocumentor/tutorials.pkg#conversion.ppage
402      * @var array
403      */
404      var $template_options;
405
406      /**
407      * Tutorials and Extended Documentation parsed from a tutorials/package[/subpackage]
408      * directory
409      * @tutorial tutorials.pkg
410      * @access private
411      */
412      var $tutorials = array();
413
414      /**
415      * tree-format structure of tutorials and their child tutorials, if any
416      * @var array
417      * @access private
418      */
419      var $tutorial_tree = false;
420
421      /**
422      * list of tutorials that have already been processed. Used by @link _setupTutorialTree()
423      * @var array
424      * @access private
425      */
426      var $processed_tutorials;
427
428      /**
429      * List of all @todo tags and a link to the element with the @todo
430      * Format: array(package => array(link to element, array(todo {@link
431      * parserTag},...),...))
432      * @tutorial tags.todo.pkg
433      * @var array
434      */
435      var $todoList = array();
436
437      /**
438      * Directory where compiled templates go - will be deleted on exit
439      * @var string
440      * @access private
441      */
442      var $_compiledDir = array();
443
444      /**
445      * Initialize Converter data structures
446      * @param array {@link $all_packages} value
447      * @param array {@link $package_parents} value
448      * @param Classes {@link $classes} value
449      * @param ProceduralPages {@link $proceduralpages} value
450      * @param array {@link $package_output} value

```

```

446 * @param boolean {@link $parseprivate} value
447 * @param boolean {@link $quietmode} value
448 * @param string {@link $targetDir} value
449 * @param string {@link $templateDir} value
450 * @param string {@link $title} value
451 */
452 function Converter(& $allp, & $packp, & $classes, & $procpages, $po, $pp, $qm,
453 $targetDir, $template, $title)
454 {
455     $this-> all_packages = $allp;
456     $this-> package_parents = $packp;
457     $this-> package = $GLOBALS['phpDocumentor_DefaultPackageName'];
458     $this-> proceduralpages = & $procpages;
459     $this-> package_output = $po;
460     if (is_array($po))
461     {
462         $a = $po[0];
463         $this-> all_packages = array_flip($po);
464         $this-> all_packages[$a] = 1;
465     }
466     $this-> parseprivate = $pp;
467     $this-> quietmode = $qm;
468     $this-> classes = & $classes;
469     $this-> roots = $classes-> getRoots($this-> processSpecialRoots);
470     $this-> title = $title;
471     $this-> setTemplateDir($template);
472     $this-> setTargetdir($targetDir);
473 }
474 /**
475 * Called by IntermediateParser after creation
476 * @access private
477 */
478 function setTutorials($tutorials)
479 {
480     $this-> tutorials = $tutorials;
481 }
482
483 /**
484 * @param pkg/cls/proc the tutorial type to search for
485 * @param tutorial name
486 * @param string package name
487 * @param string subpackage name, if any
488 * @return false/parserTutorial if the tutorial exists, return it
489 */
490 function hasTutorial($type, $name, $package, $subpackage = '')
491 {
492     if (isset($this-> tutorials[$package][$subpackage][$type][$name . '...' . $type]))
493         return $this-> tutorials[$package][$subpackage][$type][$name . '...' . $type];
494     return false;
495 }
496
497 /**
498 * Called by {@link walk()} while converting, when the last class element
499 * has been parsed.
500 *
501 * A Converter can use this method in any way it pleases. HTMLframesConverter
502 * uses it to complete the template for the class and to output its
503 * documentation
504 * @see HTMLframesConverter::endClass()
505 * @abstract
506 */
507 function endClass()
508 {
509 }
510
511 /**
512 * Called by {@link walk()} while converting, when the last procedural page
513 * element has been parsed.
514 *
515 * A Converter can use this method in any way it pleases. HTMLframesConverter
516 * uses it to complete the template for the procedural page and to output its
517 * documentation
518 * @see HTMLframesConverter::endClass()
519 * @abstract
520 */
521 function endPage()
522 {
523 }
524

```

```

525 /**
526 * Called by {@link walk()} while converting.
527 *
528 * This method is intended to be the place that {@link $pkg_elements} is
529 * formatted for output.
530 * @see HTMLframesConverter::formatPkgIndex()
531 * @abstract
532 */
533 function formatPkgIndex()
534 {
535 }
536
537 /**
538 * Called by {@link walk()} while converting.
539 *
540 * This method is intended to be the place that {@link $elements} is
541 * formatted for output.
542 * @see HTMLframesConverter::formatIndex()
543 * @abstract
544 */
545 function formatIndex()
546 {
547 }
548
549 /**
550 * Called by {@link walk()} while converting.
551 *
552 * This method is intended to be the place that any of
553 * {@link $class_elements}, {@link $function_elements}, {@link $page_elements},
554 * {@link $define_elements}, and {@link $global_elements} is formatted for
555 * output, depending on the value of {@link $leftindex}
556 * @see HTMLframesConverter::formatLeftIndex()
557 * @abstract
558 */
559 function formatLeftIndex()
560 {
561 }
562
563 /**
564 * Called by {@link parserSourceInlineTag::stringConvert()} to allow
565 * converters to format the source code the way they'd like.
566 *
567 * default returns it unchanged (html with xhtml tags)
568 * @param string output from highlight_string() - use this function to
569 * reformat the returned data for Converter-specific output
570 * @return string
571 * @deprecated in favor of tokenizer-based highlighting. This will be
572 * removed for 2.0
573 */
574 function unmangle($sourcecode)
575 {
576     return $sourcecode;
577 }
578
579 /**
580 * Initialize highlight caching
581 */
582 function startHighlight()
583 {
584     $this-> _highlightCache = array(false, false);
585     $this-> _appendHighlight = '';
586 }
587
588 function getHighlightState()
589 {
590     return $this-> _highlightCache;
591 }
592
593 function _setHighlightCache($type, $token)
594 {
595     $test = ($this-> _highlightCache[0] === $type &&
596 == $token);
597     if (!$test) {
598         $this-> _appendHighlight = $this-> flushHighlightCache();
599     } else {
600         $this-> _appendHighlight = '';
601     }
602     $this-> _highlightCache = array($type, $token);
603     return $test;
604 }

```

```

604
605 	/***
606 	* Return the close text for the current token
607 	* @return string
608 	*/
609 	function flushHighlightCache()
610 	{
611 		$hc = $this-> _highlightCache;
612 		$this-> _highlightCache = array(false, false);
613 		if ($hc[0]) {
614 			if (!isset($this-> template_options[$hc[0]][ '/' . $hc[1]])) {
615 				return '';
616 			}
617 			return $this-> template_options[$hc[0]][ '/' . $hc[1]];
618 		}
619 		return '';
620 	}
621
622 	/**
623 	* Used to allow converters to format the source code the way they'd like.
624 	*
625 	* default returns it unchanged. Mainly used by the {@link HighlightParser}
626 	* {@internal}
627 	* The method takes information from options.ini, the template options
628 	* file, specifically the [highlightSourceTokens] and [highlightSource]
629 	* sections, and uses them to enclose tokens.
630 	*
631 	* {@source }
632 	* @param integer token value from {@link PHP MANUAL#tokenizer tokenizer constants}
633 	* @param string contents of token
634 	* @param boolean whether the contents are preformatted or need modification
635 	* @return string
636 	*/
637 	function highlightSource($token, $word, $preformatted = false)
638 	{
639 		if ($token !== false)
640 		{
641 			if (!$preformatted) $word = $this-> postProcess($word);
642 			if (isset($this-> template_options['highlightSourceTokens'][$token_name($token)]))
643 			{
644 				if ($this-> _setHighlightCache('highlightSourceTokens', $token_name($token))) {
645 					return $word;
646 				}
647 				$e = $this-> _appendHighlight;
648 				return $e . $this-
649 > template_options['highlightSourceTokens'][$token_name($token)] . $word;
650 			} else
651 			{
652 				$this-> _setHighlightCache(false, false);
653 				$e = $this-> _appendHighlight;
654 				return $e . $word;
655 			}
656 		} else
657 		{
658 			if (isset($this-> template_options['highlightSource'][$word]))
659 			{
660 				$newword = ($preformatted ? $word : $this-> postProcess($word));
661 				if ($this-> _setHighlightCache('highlightSource', $word)) {
662 					return $newword;
663 				}
664 				$e = $this-> _appendHighlight;
665 				return $e . $this-> template_options['highlightSource'][$word] . $newword;
666 			} else
667 			{
668 				$this-> _setHighlightCache(false, false);
669 				$e = $this-> _appendHighlight;
670 				return $e . ($preformatted ? $word : $this-> postProcess($word));
671 			}
672 		}
673 	}
674 	/**
675 	* Used to allow converters to format the source code of DocBlocks the way
676 	* they'd like.
677 	*
678 	* default returns it unchanged. Mainly used by the {@link HighlightParser}
679 	* {@internal}
680 	* The method takes information from options.ini, the template options
681 	* file, specifically the [highlightDocBlockSourceTokens] section, and uses
682 	* it to enclose tokens.

```

```

683 *
684 * {@source } }
685 * @param string name of docblock token type
686 * @param string contents of token
687 * @param boolean whether the contents are preformatted or need modification
688 * @return string
689 */
690 function highlightDocBlockSource($token, $word, $preformatted = false)
691 {
692     if (empty($word)) {
693         $this-> _setHighlightCache(false, false);
694         $e = $this-> _appendHighlight;
695         return $e . $word;
696     }
697     if (isset($this-> template_options['highlightDocBlockSourceTokens'][$token]))
698     {
699         if (!$preformatted) $word = $this-> postProcess($word);
700         if ($this-> _setHighlightCache('highlightDocBlockSourceTokens', $token)) {
701             return $word;
702         }
703         $e = $this-> _appendHighlight;
704         return $e . $this-> template_options['highlightDocBlockSourceTokens'][$token] .
$word;
705     } else {
706         $this-> _setHighlightCache(false, false);
707         $e = $this-> _appendHighlight;
708         return $e . ($preformatted ? $word : $this-> postProcess($word));
709     }
710 }
711 /**
712 * Used to allow converters to format the source code of Tutorial XML the way
713 * they'd like.
714 *
715 * default returns it unchanged. Mainly used by the {@link HighlightParser}
716 * {@internal
717 * The method takes information from options.ini, the template options
718 * file, specifically the [highlightDocBlockSourceTokens] section, and uses
719 * it to enclose tokens.
720 *
721 * {@source } }
722 * @param string name of docblock token type
723 * @param string contents of token
724 * @param boolean whether the contents are preformatted or need modification
725 * @return string
726 */
727 function highlightTutorialSource($token, $word, $preformatted = false)
728 {
729     if (empty($word)) {
730         $this-> _setHighlightCache(false, false);
731         $e = $this-> _appendHighlight;
732         return $e . $word;
733     }
734     if (isset($this-> template_options['highlightTutorialSourceTokens'][$token]))
735     {
736         if (!$preformatted) $word = $this-> postProcess($word);
737         if ($this-> _setHighlightCache('highlightTutorialSourceTokens', $token)) {
738             return $word;
739         }
740         $e = $this-> _appendHighlight;
741         return $e . $this-> template_options['highlightTutorialSourceTokens'][$token] .
$word;
742     } else {
743         $this-> _setHighlightCache(false, false);
744         $e = $this-> _appendHighlight;
745         return $e . ($preformatted ? $word : $this-> postProcess($word));
746     }
747 }
748 /**
749 * Called by {@link parserReturnTag::Convert()} to allow converters to
750 * change type names to desired formatting
751 *
752 * Used by {@link XMLDocBookConverter::type_adjust()} to change true and
753 * false to the peardoc2 values
754 * @param string
755 * @return string
756 */
757 function type_adjust($typename)
758 {
759 }

```

```

761     return $typename;
762 }
763
764 /**
765  * Used to convert the {@example} inline tag in a docblock.
766 *
767  * By default, this just wraps ProgramExample
768  * @see XMLDocBookpeardoc2Converter::exampleProgramExample
769  * @param string
770  * @param boolean true if this is to highlight a tutorial <programlisting>
771  * @return string
772 */
773 function exampleProgramExample($example, $tutorial = false, $inlinesourceparse =
774 null/*false*/,
775                                     $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
776 null/*false*/)
777 {
778     return $this-> ProgramExample($example, $tutorial, $inlinesourceparse, $class,
779                                     $linenum, $filesourcepath);
780 }
781
782 /**
783  * Used to convert the <<code>> tag in a docblock
784  * @param string
785  * @param boolean true if this is to highlight a tutorial <programlisting>
786  * @return string
787 */
788 function ProgramExample($example, $tutorial = false, $inlinesourceparse = null/*false*/,
789                         $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
790 null/*false*/)
791 {
792     $this-> highlightingSource = true;
793     if (tokenizer_ext)
794     {
795         $e = $example;
796         if (!is_array($example))
797         {
798             $obj = new phpDocumentorTWordParser;
799             $obj-> setup($example);
800             $e = $obj-> getFileSource();
801             $bOpenTagFound = false;
802             foreach ($e as $ke => $ee)
803             {
804                 foreach ($ee as $kee => $eee)
805                 {
806                     if ((int) $e[$ke][$kee][0] == T_OPEN_TAG)
807                     {
808                         $bOpenTagFound = true;
809                     }
810                 }
811             }
812             if (!$bOpenTagFound)
813             {
814                 $example = "<?php\n" . $example;
815                 $obj-> setup($example);
816                 $e = $obj-> getFileSource();
817                 unset($e[0]);
818                 $e = array_values($e);
819             }
820             unset($obj);
821         }
822         $saveclass = $this-> class;
823         $parser = new phpDocumentor_HighlightParser;
824         if (!isset($inlinesourceparse))
825         {
826             $example = $parser-> parse($e, $this, true); // force php mode
827         } else
828         {
829             if (isset($filesourcepath))
830             {
831                 $example = $parser-> parse($e, $this, $inlinesourceparse, $class,
832                                     $linenum, $filesourcepath);
833             } elseif (isset($linenum))
834             {
835                 $example = $parser-> parse($e, $this, $inlinesourceparse, $class,
836                                     $linenum);
837             } elseif (isset($class))
838             {
839                 $example = $parser-> parse($e, $this, $inlinesourceparse, $class);
840             } else
841             {

```

```

835             $example = $parser-> parse($e, $this, $inlinesourceparse);
836         }
837     }
838     $this-> class = $saveclass;
839 }
840 {
841     $example = $this-> postProcess($example);
842 }
843 $this-> highlightingSource = false;
844
845 if ($tutorial)
846 {
847     return $example;
848 }
849
850 if (!isset($this-> template_options['desctranslate'])) return $example;
851 if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
852 $example = $this-> template_options['desctranslate'][['code']] . $example;
853 if (!isset($this-> template_options['desctranslate'][['/code']])) return $example;
854 return $example . $this-> template_options['desctranslate'][['/code']];
855 }
856
857 /**
858 * @param string
859 */
860 function TutorialExample($example)
861 {
862     $this-> highlightingSource = true;
863     $parse = new phpDocumentor_TutorialHighlightParser;
864     $x = $parse-> parse($example, $this);
865     $this-> highlightingSource = false;
866     return $x;
867 }
868
869 /**
870 * Used to convert the contents of <><li>> in a docblock
871 * @param string
872 * @return string
873 */
874 function ListItem($item)
875 {
876     if (!isset($this-> template_options['desctranslate'])) return $item;
877     if (!isset($this-> template_options['desctranslate'][['li']])) return $item;
878     $item = $this-> template_options['desctranslate'][['li']] . $item;
879     if (!isset($this-> template_options['desctranslate'][['/li']])) return $item;
880     return $item . $this-> template_options['desctranslate'][['/li']];
881 }
882
883 /**
884 * Used to convert the contents of <><ol>> or <><ul>> in a docblock
885 * @param string
886 * @return string
887 */
888 function EncloseList($list,$ordered)
889 {
890     $listname = ($ordered ? 'ol' : 'ul');
891     if (!isset($this-> template_options['desctranslate'])) return $list;
892     if (!isset($this-> template_options['desctranslate'][$listname])) return $list;
893     $list = $this-> template_options['desctranslate'][$listname] . $list;
894     if (!isset($this-> template_options['desctranslate'][['/'.$listname]])) return $list;
895     return $list . $this-> template_options['desctranslate'][['/'.$listname]];
896 }
897
898 /**
899 * Used to convert the contents of <><pre>> in a docblock
900 * @param string
901 * @return string
902 */
903 function PreserveWhiteSpace($string)
904 {
905     if (!isset($this-> template_options['desctranslate'])) return $string;
906     if (!isset($this-> template_options['desctranslate'][['pre']])) return $string;
907     $string = $this-> template_options['desctranslate'][['pre']] . $string;
908     if (!isset($this-> template_options['desctranslate'][['/pre']])) return $string;
909     return $string . $this-> template_options['desctranslate'][['/pre']];
910 }
911
912 /**
913 * Used to enclose a paragraph in a docblock
914 * @param string

```

```

915     * @return string
916     */
917     function EncloseParagraph($para)
918     {
919         if (!isset($this-> template_options['desctranslate'])) return $para;
920         if (!isset($this-> template_options['desctranslate'][['p']])) return $para;
921         $para = $this-> template_options['desctranslate'][['p']] . $para;
922         if (!isset($this-> template_options['desctranslate'][['/p']])) return $para;
923         return $para . $this-> template_options['desctranslate'][['/p']];
924     }
925
926 /**
927 * Used to convert the contents of <>b></> in a docblock
928 * @param string
929 * @return string
930 */
931     function Bolden($para)
932     {
933         if (!isset($this-> template_options['desctranslate'])) return $para;
934         if (!isset($this-> template_options['desctranslate'][['b']])) return $para;
935         $para = $this-> template_options['desctranslate'][['b']] . $para;
936         if (!isset($this-> template_options['desctranslate'][['/b']])) return $para;
937         return $para . $this-> template_options['desctranslate'][['/b']];
938     }
939
940 /**
941 * Used to convert the contents of <>i></> in a docblock
942 * @param string
943 * @return string
944 */
945     function Italicize($para)
946     {
947         if (!isset($this-> template_options['desctranslate'])) return $para;
948         if (!isset($this-> template_options['desctranslate'][['i']])) return $para;
949         $para = $this-> template_options['desctranslate'][['i']] . $para;
950         if (!isset($this-> template_options['desctranslate'][['/i']])) return $para;
951         return $para . $this-> template_options['desctranslate'][['/i']];
952     }
953
954 /**
955 * Used to convert the contents of <>var></> in a docblock
956 * @param string
957 * @return string
958 */
959     function Varize($para)
960     {
961         if (!isset($this-> template_options['desctranslate'])) return $para;
962         if (!isset($this-> template_options['desctranslate'][['var']])) return $para;
963         $para = $this-> template_options['desctranslate'][['var']] . $para;
964         if (!isset($this-> template_options['desctranslate'][['/var']])) return $para;
965         return $para . $this-> template_options['desctranslate'][['/var']];
966     }
967
968 /**
969 * Used to convert the contents of <>kbd></> in a docblock
970 * @param string
971 * @return string
972 */
973     function Kbdize($para)
974     {
975         if (!isset($this-> template_options['desctranslate'])) return $para;
976         if (!isset($this-> template_options['desctranslate'][['kbd']])) return $para;
977         $para = $this-> template_options['desctranslate'][['kbd']] . $para;
978         if (!isset($this-> template_options['desctranslate'][['/kbd']])) return $para;
979         return $para . $this-> template_options['desctranslate'][['/kbd']];
980     }
981
982 /**
983 * Used to convert the contents of <>samp></> in a docblock
984 * @param string
985 * @return string
986 */
987     function Sampize($para)
988     {
989         if (!isset($this-> template_options['desctranslate'])) return $para;
990         if (!isset($this-> template_options['desctranslate'][['samp']])) return $para;
991         $para = $this-> template_options['desctranslate'][['samp']] . $para;
992         if (!isset($this-> template_options['desctranslate'][['/samp']])) return $para;
993         return $para . $this-> template_options['desctranslate'][['/samp']];
994     }

```

```

995
996 /**
997 * Used to convert <>br></> in a docblock
998 * @param string
999 * @return string
1000 */
1001 function Br($para)
1002 {
1003     if (!isset($this-> template_options['desctranslate'])) return $para;
1004     if (!isset($this-> template_options['desctranslate']['br'])) return $para;
1005     $para = $this-> template_options['desctranslate']['br'] . $para;
1006     return $para;
1007 }
1008 /**
1009 * This version does nothing
1010 *
1011 * Perform necessary post-processing of string data. For example, the HTML
1012 * Converters should escape < and > to become &lt; and &gt;
1013 * @return string
1014 */
1015 function postProcess($text)
1016 {
1017     return $text;
1018 }
1019
1020 /**
1021 * Creates a table of contents for a {@toc} inline tag in a tutorial
1022 *
1023 * This function should return a formatted table of contents. By default, it
1024 * does nothing, it is up to the converter to format the TOC
1025 * @abstract
1026 * @return string table of contents formatted for use in the current output format
1027 * @param array format: array(array('tagname' => section, 'link' => returnsee link,
1028 'id' => anchor name, 'title' => from title tag),...)
1029 */
1030 function formatTutorialTOC($toc)
1031 {
1032     return '';
1033 }
1034
1035 /**
1036 * Write out the formatted source code for a php file
1037 *
1038 * This function provides the primary functionality for the
1039 * {@tutorial tags.filesource.pkg} tag.
1040 * @param string full path to the file
1041 * @param string fully highlighted/linked source code of the file
1042 * @abstract
1043 */
1044 function writeSource($filepath, $source)
1045 {
1046     debug($source);
1047     return;
1048 }
1049
1050 /**
1051 * Write out the formatted source code for an example php file
1052 *
1053 * This function provides the primary functionality for the
1054 * {@tutorial tags.example.pkg} tag.
1055 * @param string example title
1056 * @param string example filename (no path)
1057 * @param string fully highlighted/linked source code of the file
1058 * @abstract
1059 */
1060 function writeExample($title, $path, $source)
1061 {
1062     return;
1063 }
1064
1065 /** Translate the path info into a unique file name for the highlighted
1066 * source code.
1067 * @param string $pathinfo
1068 * @return string
1069 */
1070 function getFileSourceName($path)
1071 {
1072     global $phpDocumentor_options;
1073     $pathinfo = $this-> proceduralpages-> getPathInfo($path, $this);

```

```

1074     $pathinfo['source_loc'] =
1075     str_replace($_phpDocumentor_options['Program_Root'].'/', '', $pathinfo['source_loc']);
1076     $pathinfo['source_loc'] = str_replace('/', '_', $pathinfo['source_loc']);
1077     return
1078   "  fsource_{$pathinfo['package']}_{$pathinfo['subpackage']}_{$pathinfo['source_loc']}";
1079
1080   /** Return the fixed path to the source-code file folder.
1081   * @param string $base Path is relative to this folder
1082   * @return string
1083   */
1084   function getFileSourcePath($base)
1085   {
1086     if (substr($base, strlen($base) - 1) != PATH_DELIMITER) {
1087       $base .= PATH_DELIMITER;
1088     }
1089     return $base . '__filesource';
1090
1091   /** Return the path to the current
1092   * @param string $pathinfo
1093   * @return string
1094   */
1095   function getCurrentPageURL()
1096   {
1097     return "{$srcdir}" . PATH_DELIMITER . $this-> page_dir;
1098   }
1099
1100 /**
1101  * @return string an output-format dependent link to phpxref-style highlighted
1102  * source code
1103  * @abstract
1104  */
1105   function getSourceLink($path)
1106   {
1107     return '';
1108   }
1109
1110 /**
1111  * @return string Link to the current page being parsed.
1112  * Should return {@link $curname} and a converter-specific extension.
1113  * @abstract
1114  */
1115   function getCurrentPageLink()
1116   {
1117   }
1118
1119 /**
1120  * Return a line of highlighted source code with formatted line number
1121  *
1122  * If the $path is a full path, then an anchor to the line number will be
1123  * added as well
1124  * @param integer line number
1125  * @param string highlighted source code line
1126  * @param false|string full path to @filesource file this line is a part of,
1127  * if this is a single line from a complete file.
1128  * @return string formatted source code line with line number
1129  */
1130   function sourceLine($linenumber, $line, $path = false)
1131   {
1132     if ($path)
1133     {
1134       return $this-> getSourceAnchor($path, $linenumber) .
1135       $this-> Br(sprintf('%-
6u', $linenumber).str_replace("\n" , ' ', $line));
1136     } else
1137     {
1138       return $this-> Br(sprintf('%-
6u', $linenumber).str_replace("\n" , ' ', $line));
1139     }
1140   }
1141
1142 /**
1143  * Determine whether an element's file has generated source code, used for
1144  * linking to line numbers of source.
1145  *
1146  * Wrapper for {@link $sourcePaths} in this version
1147  *
1148  * {@internal since file paths get stored with most/all slashes
1149  * set to forward slash '/', we need to doublecheck that

```

```

1150 * we're not given a backslashed path to search for...
1151 * if we are, it's likely that it was originally stored
1152 * with a forward slash. Further, I'm not convinced it's safe
1153 * to just check the {@link PHPDOCUMENTOR_WINDOWS} flag, so I'm checking
1154 * specifically for backslashes instead.}}
1155 *
1156 * @param string full path to the source code file
1157 * @return boolean
1158 */
1159 function hasSourceCode($path)
1160 {
1161     return isset($this-> sourcePaths[$path]);
1162     if (strpos($path, '\\') > - 1) {
1163         $modifiedPath = str_replace('\\', '/', $path);
1164         return isset($this-> sourcePaths[$modifiedPath]);
1165     } else {
1166         return isset($this-> sourcePaths[$path]);
1167     }
1168 }
1169 /**
1170 * Mark a file as having had source code highlighted
1171 * @param string full path of source file
1172 */
1173 function setSourcePaths($path)
1174 {
1175     $this-> sourcePaths[$path] = true;
1176 }
1177 /**
1178 * Used to translate an XML DocBook entity like &rdquo; from a tutorial by
1179 * reading the options.ini file for the template.
1180 * @param string entity name
1181 */
1182 function TranslateEntity($name)
1183 {
1184     if (!isset($this-> template_options['ppage']))
1185     {
1186         if (!$this-> template_options['preservedocbooktags'])
1187             return '';
1188         else
1189             return '&' . $name . ';';
1190     }
1191     if (isset($this-> template_options['ppage'][('&' . $name . ';' )]))
1192     {
1193         return $this-> template_options['ppage'][('&' . $name . ';' )];
1194     } else
1195     {
1196         if (!$this-> template_options['preservedocbooktags'])
1197             return '';
1198         else
1199             return '&' . $name . ';';
1200     }
1201 }
1202 /**
1203 * Used to translate an XML DocBook tag from a tutorial by reading the
1204 * options.ini file for the template.
1205 * @param string tag name
1206 * @param string any attributes Format: array(name => value)
1207 * @param string the tag contents, if any
1208 * @param string the tag contents, if any, unpost-processed
1209 * @return string
1210 */
1211 function TranslateTag($name,$attr,$cdata,$unconvertedcdata)
1212 {
1213     if (!isset($this-> template_options['ppage']))
1214     {
1215         if (!$this-> template_options['preservedocbooktags'])
1216             return $cdata;
1217         else
1218             return '<' . $name . $this-
1219             > AttrToString($name,$attr,true). '>' . $cdata. '</' . $name . '>' . "\n";
1220     }
1221     // make sure this template transforms the tag into something
1222     if (isset($this-> template_options['ppage'][$name]))
1223     {
1224         // test for global attribute transforms like $attr$role = class, changing
1225         // all roles="" attributes to class="" in html, for example
1226         foreach($attr as $att => $val)
1227     }

```

```

1229
1230     {
1231         if (isset($this-> template_options['$attr$'].$att)))
1232     {
1233         $new = '';
1234         if (!isset($this-> template_options['$attr$'].$att]['close']))
1235         {
1236             $new .= '<' . $this-> template_options['$attr$'].$att]['open'];
1237             if (isset($this-> template_options['$attr$'].$att]['cdata!']))
1238             {
1239                 if (isset($this-> template_options['$attr$'].$att]['separateall']))
1240                 $new .= $this-> template_options['$attr$'].$att]['separator'];
1241                 else
1242                     $new .= ' ';
1243                 $new .= $this-> template_options['$attr$'].$att]['$'].$att];
1244                 $new .= $this-> template_options['$attr$'].$att]['separator'];
1245                 if ($this-> template_options['$attr$'].$att]['quotevalues']) $val
1246                 .= "'"; ;
1247             }
1248             $new .= $val.'>';
1249         }
1250         $new .= '</' . $this-
1251     > template_options['$attr$'].$att]['open']. '>';
1252     }
1253     else
1254     {
1255         $new .= $this-> template_options['$attr$'].$att]['open'];
1256     }
1257     unset($attr[$att]);
1258     $cdata = $new . $cdata;
1259 }
1260
1261     if (!isset($this-> template_options['ppage'][ '/' . $name]))
1262     { // if the close tag isn't specified, we put opening and closing tags around it,
1263     with translated attributes
1264         if (isset($this-> template_options['ppage'][$name] . '/'))
1265             $cdata = '<' . $this-> template_options['ppage'][$name].$this-
1266     > AttrToString($name,$attr). '/>' . $cdata;
1267         else
1268             $cdata = '<' . $this-> template_options['ppage'][$name].$this-
1269     > AttrToString($name,$attr). '>' . $cdata .
1270             '</' . $this-> template_options['ppage'][$name]. '>' ;
1271     }
1272     else
1273     { // if close tag is specified, use the open and close as literal
1274         if ($name == 'programlisting' && isset($attr['role']) &&
1275             ($attr['role'] == 'php' || $attr['role'] == 'tutorial' || $attr['role']
1276 == 'html'))
1277             { // highlight PHP source
1278                 var_dump($unconvertedcdata, $cdata);exit;
1279                 if ($attr['role'] == 'php') {
1280                     $cdata = $this-> ProgramExample($unconvertedcdata, true);
1281                 } elseif ($attr['role'] == 'tutorial') {
1282                     $cdata = $this-> TutorialExample($unconvertedcdata);
1283                 } elseif ($attr['role'] == 'html') {
1284                     $cdata = $unconvertedcdata;
1285                 }
1286             }
1287             { // normal case below
1288                 $cdata = $this-> template_options['ppage'][$name].$this-
1289     > AttrToString($name,$attr). $cdata . $this-> template_options['ppage'][ '/' . $name];
1290             }
1291             return $cdata;
1292     }
1293     else
1294     {
1295         if ($this-> template_options['preservedocbooktags'])
1296         {
1297             return '<' . $name.$this-> AttrToString($name,$attr,true). '>' . $cdata .
1298             '</' . $name. '>' . "\n";
1299         }
1300         {
1301             return $cdata;
1302         }
1303     }
1304 }
1305
1306 /**
1307 * Convert the attribute of a Tutorial docbook tag's attribute list

```

```

1301 * to a string based on the template options.ini
1302 * @param string tag name
1303 * @param attribute array
1304 * @param boolean if true, returns attrname="value"...
1305 * @return string
1306 */
1307 function AttrToString($tag,$attr,$unmodified = false)
1308 {
1309     $ret = '';
1310     if ($unmodified)
1311     {
1312         $ret = ' ';
1313         foreach($attr as $n =>    $v)
1314         {
1315             $ret .= $n.' = "' . $v.'"';
1316         }
1317         return $ret;
1318     }
1319 // no_attr tells us to ignore all attributes
1320 if (isset($this-> template_options['no_attr'])) return $ret;
1321 // tagname! tells us to ignore all attributes for this tag
1322 if (isset($this-> template_options['ppage'][$tag.'!'])) return $ret;
1323 if (count($attr)) $ret = ' ';
1324 // pass 1, check to see if any attributes add together
1325 $same = array();
1326 foreach($attr as $n =>    $v)
1327 {
1328     if (isset($this-> template_options['ppage'][$tag.'->' . $n]))
1329     {
1330         $same[$this-> template_options['ppage'][$tag.'->' . $n]][] = $n;
1331     }
1332 }
1333 foreach($attr as $n =>    $v)
1334 {
1335     if (isset($this-> template_options['ppage'][$tag.'->' . $n]))
1336     {
1337         if (count($same[$this-> template_options['ppage'][$tag.'->' . $n]]) == 1)
1338         {
1339             // only 1 attribute translated for this one
1340             // this is useful for equivalent value names
1341             if (isset($this-> template_options['ppage'][$tag.'->' . $n.'+'.$v])) $v
1342 = $this-> template_options['ppage'][$tag.'->' . $n.'+'.$v];
1343         } else
1344             { // more than 1 attribute combines to make the new attribute
1345                 $teststrtemp = array();
1346                 foreach($same[$this-> template_options['ppage'][$tag.'->' . $n]] as
1347                 {
1348                     $teststrtemp[] = $oldattr.'+'.$attr[$oldattr];
1349                 }
1350                 $teststrs = array();
1351                 $num = count($same[$this-> template_options['ppage'][$tag.'->' . $n]]);
1352                 for($i=0;$i< $num;$i++)
1353                 {
1354                     $started = false;
1355                     $a = '';
1356                     for($j=$i;!$started || $j != $i;$j = ($j + $i) % $num)
1357                     {
1358                         if (!empty($a)) $a .= '|';
1359                         $a .= $teststrtemp[$j];
1360                     }
1361                     $teststrs[$i] = $a;
1362                 }
1363                 $done = false;
1364                 foreach($teststrs as $test)
1365                 {
1366                     if ($done) break;
1367                     if (isset($this-> template_options['ppage'][$tag.'->' . $test]))
1368                     {
1369                         $done = true;
1370                         $v = $this-> template_options['ppage'][$tag.'->' . $test];
1371                     }
1372                     $ret .= $this-> template_options['ppage'][$tag.'->' . $n].' =
1373 "' . $v.'"';
1374                 }
1375                 if (!isset($this-> template_options['ppage'][$tag.'!'. $n]))
1376                 {
1377                     if (isset($this-> template_options['ppage'][[$attr$. $n]]))

```

```

1378         $ret .= $this-> template_options['ppage'][$attr.$n].' =
1379         .$v.'"'; ;
1380         else
1381             $ret .= $n.' = "'.$v.'"'; ;
1382     }
1383 }
1384 return $ret;
1385 }
1386 /**
1387 * Convert the title of a Tutorial docbook tag section
1388 * to a string based on the template options.ini
1389 * @param string tag name
1390 * @param array
1391 * @param string title text
1392 * @param string
1393 * @return string
1394 */
1395 function ConvertTitle($tag,$attr,$title,$cdata)
1396 {
1397     if (!isset($this-> template_options[$tag.'_title'])) return array($attr,$cdata);
1398     if (isset($this-> template_options[$tag.'_title']['tag_attr']))
1399     {
1400         $attr[$this-> template_options[$tag.'_title']['tag_attr']] = urlencode($cdata);
1401         $cdata = '';
1402     } elseif(isset($this-> template_options[$tag.'_title']['cdata_start']))
1403     {
1404         $cdata = $this-> template_options[$tag.'_title']['open'] . $title .
1405             $this-> template_options[$tag.'_title']['close'] . $cdata;
1406     } else $cdata = $title.$cdata;
1407     return array($attr,$cdata);
1408 }
1409 /**
1410 * Return a converter-specific id to distinguish tutorials and their
1411 * sections
1412 *
1413 * Used by {@}id
1414 * @return string
1415 */
1416 function getTutorialId($package,$subpackage,$tutorial,$id)
1417 {
1418     return $package.$subpackage.$tutorial.$id;
1419 }
1420 /**
1421 * Create the {@link $elements, $pkg_elements} and {@link $links} arrays
1422 * @access private
1423 * @todo version 2.0 - faulty package_output logic should be removed
1424 *
1425 *      in this version, if the parent file isn't in the package, all
1426 *      the procedural elements are simply shunted to another package!
1427 */
1428 function _createPkgElements(& $pages)
1429 {
1430     if (empty($this-> elements))
1431     {
1432         $this-> elements = array();
1433         $this-> pkg_elements = array();
1434         $this-> links = array();
1435         phpDocumentor_out('Building indexes... ');
1436         flush();
1437         foreach($pages as $j => $flob)
1438         {
1439             $this-> package = $pages[$j]-> parent-> package;
1440             $this-> subpackage = $pages[$j]-> parent-> subpackage;
1441             $this-> class = false;
1442             $this-> curfile = $pages[$j]-> parent-> getFile();
1443             $this-> curname = $this-> getPageName($pages[$j]-> parent);
1444             $this-> curpath = $pages[$j]-> parent-> getPath();
1445             $use = true;
1446             if ($this-> package_output)
1447             {
1448                 if (in_array($this-> package,$this-> package_output))
1449                 {
1450                     $this-> addElement($pages[$j]-> parent,$pages[$j]);
1451                 } else
1452                 {
1453                     if (count($pages[$j]-> classelements))
1454                 }
1455             }
1456         }
1457     }
1458 }

```

```

1457             {
1458                 list($pages[$j]->parent->package) = each($this-
1459 > package_output);
1460
1461                 reset($this->package_output);
1462                 $pages[$j]->parent->subpackage = '';
1463                 $this->addElement($pages[$j]->parent,$pages[$j]);
1464             }
1465             {
1466                 unset($pages[$j]);
1467                 continue;
1468             }
1469         }
1470         {
1471             $this->addElement($pages[$j]->parent,$pages[$j]);
1472         }
1473         if ($use)
1474         for($i=0; $i< count($pages[$j]->elements); $i++)
1475         {
1476             $pages[$j]->elements[$i]->docblock->package = $this->package;
1477             $pages[$j]->elements[$i]->docblock->subpackage = $this-
1478 > subpackage;
1479             $this->proceduralpages->replaceElement($pages[$j]->elements[$i]);
1480             $this->addElement($pages[$j]->elements[$i]);
1481         }
1482         for($i=0; $i< count($pages[$j]->classelements); $i++)
1483         {
1484             if ($this->class)
1485             {
1486                 if ($pages[$j]->classelements[$i]->type == 'class')
1487                 {
1488                     if ($this->checkKillClass($pages[$j]->classelements[$i]-
1489 > getName(),$pages[$j]->classelements[$i]->getPath())) continue;
1490                     $this->package = $pages[$j]->classelements[$i]->docblock-
1491 > package;
1492                     if ($this->package_output) if (!in_array($this->package,$this-
1493 > package_output)) continue;
1494                     $this->subpackage = $pages[$j]->classelements[$i]->
1495 > docblock->subpackage;
1496                     $this->class = $pages[$j]->classelements[$i]->name;
1497                 }
1498                 if ($pages[$j]->classelements[$i]->type == 'class')
1499                 {
1500                     if ($this->checkKillClass($pages[$j]->classelements[$i]-
1501 > getName(),$pages[$j]->classelements[$i]->getPath())) continue;
1502                     $this->package = $pages[$j]->classelements[$i]->docblock-
1503 > package;
1504                     if ($this->package_output) if (!in_array($this->package,$this-
1505 > package_output)) continue;
1506                     $this->subpackage = $pages[$j]->classelements[$i]->docblock-
1507 > subpackage;
1508                     $this->class = $pages[$j]->classelements[$i]->name;
1509                 }
1510                 phpDocumentor_out("done\n");
1511                 flush();
1512             }
1513             $this->sortIndexes();
1514             $this->sortTodos();
1515             if ($this->sort_page_contents_by_type) $this-
1516 > sortPageContentsByElementType($pages);
1517         }
1518         /**
1519          * Process the {@link $Tutorial} array
1520          *
1521          * Using the tutorialname.ini files, this method sets up tutorial
1522          * hierarchy. There is some minimal error checking to make sure that no

```

```

1523 * tutorial links to itself, even two levels deep as in tute->next->tute.
1524 *
1525 * If all tests pass, it creates the hierarchy
1526 * @uses generateTutorialOrder()
1527 * @uses _setupTutorialTree()
1528 * @access private
1529 */
1530 function _processTutorials()
1531 {
1532     $parents = $all = array();
1533     foreach($this-> tutorials as $package => $els)
1534     {
1535         if ($this-> package_output)
1536         {
1537             if (!in_array($package,$this-> package_output))
1538             {
1539                 unset($this-> tutorials[$package]);
1540                 continue;
1541             }
1542         }
1543         if (!isset($this-> pkg_elements[$package]))
1544         {
1545             unset($this-> tutorials[$package]);
1546             continue;
1547         }
1548         foreach($els as $subpackage => $els2)
1549         {
1550             foreach($els2 as $type => $tutorials)
1551             {
1552                 foreach($tutorials as $tutorial)
1553                 {
1554                     if ($tutorial-> ini)
1555                     {
1556                         if (isset($tutorial-> ini['Linked Tutorials']))
1557                         {
1558                             foreach($tutorial-> ini['Linked Tutorials'] as $child)
1559                             {
1560                                 $sub = (empty($tutorial-> subpackage) ? '' : $tutorial-
> subpackage . '/');
1561                                 $kid = $tutorial-> package . '/' . $sub . $child . '.';
1562                                 $tutorial-> tutorial_type;
1563                                 // parent includes self as a linked tutorial?
1564                                 $kidlink = $this-
> getTutorialLink($kid,false,false,array($tutorial-> package));
1565                                 if (is_object($kidlink) && $this-
> returnSee($kidlink) == $tutorial-> getLink($this))
1566                                 { // bad!
1567                                     addErrorDie(PDERROR TUTORIAL IS OWN CHILD,$tutorial-
> name,$tutorial-> name.'.ini');
1568                                 }
1569                                 $parents[] = $tutorial;
1570                             }
1571                         }
1572                         $all[$package][$subpackage][$type][] = $tutorial;
1573                     }
1574                 }
1575             }
1576         }
1577         // loop error-checking, use this to eliminate possibility of accidentally linking to a
parent as a child
1578         $testlinks = array();
1579         foreach($parents as $parent)
1580         {
1581             $testlinks[$parent-> name]['links'][] = $parent-> getLink($this);
1582             $testlinks[$parent-> name]['name'][$parent-> getLink($this)] = $parent-> name;
1583         }
1584         // generate the order of tutorials, and link them together
1585         foreach($parents as $parent)
1586         {
1587             foreach($parent-> ini['Linked Tutorials'] as $child)
1588             {
1589                 $sub = (empty($parent-> subpackage) ? '' : $parent-> subpackage . '/');
1590                 $kid = $parent-> package . '/' . $sub . $child . '.' . $parent-
> tutorial_type;
1591                 // child tutorials must be in the same package AND subpackage
1592                 // AND have the same extension as the parent, makes things clearer for both ends
1593                 if (in_array($this-> returnSee($this-
> getTutorialLink($kid,false,false,array($parent-> package))),$testlinks[$parent-
> name]['links']))

```

```

1594             addErrorDie(PDERROR_TUTORIAL_IS OWN GRANDPA,$testlinks[$parent-
> name][$this-> returnSee($this-> getTutorialLink($kid,false,false,array($parent-
> package))),$kid-> name,$testlinks[$parent-> name][$this-> returnSee($this-
> getTutorialLink($kid,false,false,array($parent-> package))),$kid-> name.'.ini']);
1595             if ($this-> returnSee($this-
> getTutorialLink($kid,false,false,array($parent-> package))) == $kid)
1596             {
1597                 addWarning(PDERROR_CHILD TUTORIAL NOT FOUND, $child . '.' . $parent-
> tutorial_type, $parent-> name .'ini',$parent-> package, $parent-> subpackage);
1598             }
1599         }
1600     }
1601     $new = $tree = $roots = array();
1602     // build a list of all 'root' tutorials (tutorials without parents).
1603     foreach($parents as $i => $parent)
1604     {
1605         if (! $parent-> isChildOf($parents)) {
1606             $roots[] = $parent;
1607         }
1608     }
1609     $parents = $roots;
1610     // add the parents and all child tutorials in order to the list of tutorials to process
1611     foreach($parents as $parent)
1612     {
1613         $this-> generateTutorialOrder($parent,$all,$new);
1614     }
1615     if (count($all))
1616     {
1617         // add the leftover tutorials
1618         foreach($all as $package => $els)
1619         {
1620             foreach($els as $subpackage => $els2)
1621             {
1622                 foreach($els2 as $type => $tutorials)
1623                 {
1624                     foreach($tutorials as $tutorial)
1625                     {
1626                         $new[$package][$subpackage][$type][] = $tutorial;
1627                     }
1628                 }
1629             }
1630         }
1631     }
1632     // remove the old, unprocessed tutorials, and set it up with the next code
1633     $this-> tutorials = array();
1634     // reset integrity of the tutorial list
1635     $prev = false;
1636     uksort($new, 'tutorialcmp');
1637     // debug($this->vardump_tree($new));exit;
1638     foreach($new as $package => $els)
1639     {
1640         foreach($els as $subpackage => $els2)
1641         {
1642             foreach($els2 as $type => $tutorials)
1643             {
1644                 foreach($tutorials as $tutorial)
1645                 {
1646                     if ($prev)
1647                     {
1648                         $this-
1649                         tutorials[$prevpackage][$prevsubpackage][$prevtype][$prevname]-> setNext($tutorial,$this);
1650                         $tutorial-> setPrev($prev,$this);
1651                     }
1652                     $this-> tutorials[$package][$subpackage][$type][$tutorial-> name] =
1653                         $tutorial;
1654                     $prev = $tutorial-> getLink($this,true);
1655                     $prevpackage = $package;
1656                     $prevsubpackage = $subpackage;
1657                     $prevtype = $type;
1658                     $prevname = $tutorial-> name;
1659                 }
1660             }
1661         }
1662     }
1663     $this-> tutorial_tree = $this-> _setupTutorialTree();
1664     return $new;
1665 }
1666 /**
 * called by {@link phpDocumentor_IntermediateParser::Convert()} to traverse

```

```

1667 * the array of pages and their elements, converting them to the output format
1668 *
1669 * The walk() method should be flexible enough such that it never needs
1670 * modification. walk() sets up all of the indexes, and sorts everything in
1671 * logical alphabetical order. It then passes each element individually to
1672 * {@link Convert()}, which then passes to the Convert*() methods. A child
1673 * Converter need not override any of these unless special functionality must
1674 * be added. see {@tutorial Converters/template.vars.cls} for details.
1675 * {@internal}
1676 * walk() first creates all of the indexes {@link $elements}, {@link $pkg_elements}
1677 * and the left indexes specified by {@link $leftindexes},
1678 * and then sorts them by calling {@link sortIndexes()}.
1679 *
1680 * Next, it converts all README/CHANGELOG/INSTALL-style files, using
1681 * {@link Convert_RIC}.
1682 *
1683 * After this, it
1684 * passes all package-level docs to Convert(). Then, it calls the index
1685 * sorting functions {@link formatPkgIndex()}, {@link formatIndex()} and
1686 * {@link formatLeftIndex()}.
1687 *
1688 * Finally, it converts each procedural page in alphabetical order. This
1689 * stage passes elements from the physical file to Convert() in alphabetical
1690 * order. First, procedural page elements {@link parserDefine}, {@link parserInclude}
1691 * {@link parserGlobal}, and {@link parserFunction} are passed to Convert().
1692 *
1693 * Then, class elements are passed in this order: {@link parserClass}, then
1694 * all of the {@link parserVar}s in the class and all of the
1695 * {@link parserMethod}s in the class. Classes are in alphabetical order,
1696 * and both vars and methods are in alphabetical order.
1697 *
1698 * Finally, {@link ConvertErrorLog()} is called and the data walk is complete.})
1699 * @param array Format: array(fullpath => {@link parserData} structure with full {@link
1700 * parserData::$elements}
1701 * and {@link parserData::class_elements}.
1702 * @param array Format: array({@link parserPackagePage} 1, {@link parserPackagePage} 2,...)
1703 * @uses Converter::_createPkgElements() sets up {@link $elements} and
1704 * {@link $pkg_elements} array, as well as {@link $links}
1705 */
1706 function walk(& $pages, & $package_pages)
1707 {
1708     if (empty($pages))
1709     {
1710         die("<b>ERROR</b>: nothing parsed" );
1711     }
1712     $this-> _createPkgElements($pages);
1713     if (count($this-> ric))
1714     {
1715         phpDocumentor_out("Converting README/INSTALL/CHANGELOG contents...\n");
1716         flush();
1717         foreach($this-> ric as $name => $contents)
1718         {
1719             phpDocumentor_out("      $name..." );
1720             flush();
1721             $this-> Convert_RIC($name,$contents);
1722         }
1723         phpDocumentor_out("\ndone\n");
1724         flush();
1725     }
1726     foreach($package_pages as $i => $perp)
1727     {
1728         if ($this-> package_output)
1729         {
1730             if (!in_array($package_pages[$i]-> package,$this-> package_output))
1731             continue;
1732             phpDocumentor_out('Converting package page for package '. $package_pages[$i]-> package.'...');
1733             flush();
1734             $this-> package = $package_pages[$i]-> package;
1735             $this-> subpackage = '';
1736             $this-> class = false;
1737             $this-> Convert($package_pages[$i]);
1738             phpDocumentor_out("done\n");
1739             flush();
1740             phpDocumentor_out("Converting tutorials/extended docs\n");
1741             flush();
1742             // get tutorials into the order they will display, and set next/prev links
1743             $new = $this-> _processTutorials();

```

```

1744     foreach($this-> tutorials as $package => $els)
1745     {
1746         foreach($els as $subpackage => $els2)
1747         {
1748             foreach($els2 as $type => $tutorials)
1749             {
1750                 foreach($tutorials as $tutorial)
1751                 {
1752                     switch ($type)
1753                     {
1754                         case 'pkg' :
1755                             $a = '';
1756                             if ($tutorial-> ini)
1757                                 $a .= 'Top-level ';
1758                             if (!empty($tutorial-> subpackage))
1759                                 $a .= 'Sub-';
1760                             $ptext = "    Converting ${a}Package-level tutorial
"
1761                             . $tutorial-> name.'...';
1762                             break;
1763                         case 'cls' :
1764                             $a = '';
1765                             if ($tutorial-> ini)
1766                                 $a .= 'Top-level ';
1767                             $ptext = "    Converting ${a}Class-level tutorial "
1768 $tutorial-> name ." and associating...";
1769                         $link = Converter::getClassLink(str_replace('.cls','', $tutorial-
1770 > name), $tutorial-> package);
1771                         if (is_object($link))
1772                         {
1773                             if ($this-> sort_absolutely_everything)
1774                             {
1775                                 $addend = 'unsuccessful ';
1776                                 if (isset($this-> package_elements[$tutorial-
1777 > package][$tutorial-> subpackage]['class'][$link-> name]))
1778                                 {
1779                                     $this-> package_elements[$tutorial-
1780 > package][$tutorial-> subpackage]['class'][$link-> name][0]-> addTutorial($tutorial,$this);
1781                                     $addend = 'success ';
1782                                 }
1783                             } else
1784                             {
1785                                 $addend = 'unsuccessful ';
1786                                 if (!isset($this-> classes-
1787 > killclass[str_replace('.cls','', $tutorial-> name)]) && !isset($this-> classes-
1788 > killclass[str_replace('.cls','', $tutorial-> name)][$tutorial-> path]))
1789                                 {
1790                                     foreach($pages as $j => $inf)
1791                                     {
1792                                         foreach($inf-> classelements as $i =>
1793                                         {
1794                                             if ($class-> type == 'class' &&
1795 $class-> name == str_replace('.cls','', $tutorial-> name)) &&
1796 $class-> path == $link-
1797 > path)
1798                                     {
1799                                         if ($pages[$j]-> classelements[$i]->
1800 > addTutorial($tutorial,$this));
1801                                         $addend = 'success ';
1802                                     }
1803                                 }
1804                                 $ptext .= $addend;
1805                             } else $ptext .= "unsuccessful ";
1806                         break;
1807                         case 'proc' :
1808                             $a = '';
1809                             if ($tutorial-> ini)
1810                                 $a .= 'Top-level ';
1811                             $ptext = "    Converting ${a}Procedural-level tutorial
"
1812                             . $tutorial-> name." and associating...";
1813                             $link = Converter::getPageLink(str_replace('.proc','', $tutorial-
1814 > name), $tutorial-> package);
1815                             if (is_object($link))
1816                             {
1817                                 $addend = 'unsuccessful ';
1818                                 if ($this-> sort_absolutely_everything)
1819                                 {
1820                                     if (isset($this-> package_elements[$tutorial-

```

```

> package][$tutorial-> subpackage]['page'][$link-> path)))
1811                                     {
1812                                         $this-> package_elements[$tutorial-
1813                                         subpackage]['page'][$link-> path][0]-> addTutorial($tutorial,$this);
1814                                         $addend = "success ";
1815                                     }
1816                                     {
1817                                         foreach($pages as $j => $info)
1818                                         {
1819                                             if ($j == $link-> path)
1820                                             {
1821                                                 $pages[$j]-> addTutorial($tutorial,$this);
1822                                                 $addend = "success ";
1823                                             }
1824                                         }
1825                                     }
1826                                     $ptext .= $addend;
1827                                 } else $ptext .= "unsuccessful ";
1828                                 break;
1829                             }
1830                             phpDocumentor_out($ptext);
1831                             flush();
1832                         $this-> package = $tutorial-> package;
1833                         $this-> subpackage = $tutorial-> subpackage;
1834                         $this-> Convert($tutorial);
1835                         phpDocumentor_out("done\n");
1836                         flush();
1837                     }
1838                 }
1839             }
1840         }
1841         phpDocumentor_out("Formatting Package Indexes...");
1842         flush();
1843         $this-> formatPkgIndex();
1844         phpDocumentor_out("done\n");
1845         flush();
1846         phpDocumentor_out("Formatting Index...");
1847         flush();
1848         $this-> formatIndex();
1849         phpDocumentor_out("done\n\n");
1850         flush();
1851         phpDocumentor_out("Formatting Left Quick Index...");
1852         flush();
1853         $this-> formatLeftIndex();
1854         phpDocumentor_out("done\n\n");
1855         flush();
1856         if ($this-> sort_absolutely_everything) return $this-> walk_everything();
1857         foreach($pages as $j => $flub)
1858         {
1859             phpDocumentor_out('Converting '.$pages[$j]-> parent-> getPath());
1860             flush();
1861             $this-> package = $pages[$j]-> parent-> package;
1862             $this-> subpackage = $pages[$j]-> parent-> subpackage;
1863             $this-> class = false;
1864             $this-> curfile = $pages[$j]-> parent-> getFile();
1865             $this-> curname = $this-> getPageName($pages[$j]-> parent);
1866             $this-> curpath = $pages[$j]-> parent-> getPath();
1867             $use = true;
1868             if ($this-> package_output)
1869             {
1870                 if (in_array($this-> package,$this-> package_output))
1871                 {
1872                     $this-> Convert($pages[$j]);
1873                 } else
1874                 {
1875                     $use = false;
1876                 }
1877             } else
1878             {
1879                 $this-> Convert($pages[$j]);
1880             }
1881             phpDocumentor_out(" Procedural Page Elements...");
1882             flush();
1883             if ($use)
1884             for($i=0; $i< count($pages[$j]-> elements); $i++)
1885             {
1886                 $a = $pages[$j]-> elements[$i]-> docblock-> getKeyword('access');
1887                 if (is_object($a)) $a = $a-> getString();
1888                 if (!$this-> parseprivate && ($a == 'private'))

```

```

1889         continue;
1900     //      phpDocumentor_out("    ".$pages[$j]->elements[$i]-
>name."\n");
1901     $pages[$j]-> elements[$i]-> docblock-> package = $this-> package;
1902     $pages[$j]-> elements[$i]-> docblock-> subpackage = $this-> subpackage;
1903     $this-> Convert($pages[$j]-> elements[$i]);
1904   }
1905   phpDocumentor_out(" Classes..." );
1906   $this-> class = false;
1907   flush();
1908   for($i=0; $i< count($pages[$j]-> classelements); $i++)
1909   {
1910     if ($this-> class)
1911     {
1912       if ($pages[$j]-> classelements[$i]-> type == 'class')
1913       {
1914         if (!$this-> killclass) $this-> endClass();
1915         $this-> killclass = false;
1916         if ($this-> checkKillClass($pages[$j]-> classelements[$i]-
> getName(),$pages[$j]-> classelements[$i]-> getPath())) continue;
1917         $this-> package = $pages[$j]-> classelements[$i]-> docblock-
> package;
1918         if ($this-> package_output) if (!in_array($this-> package,$this-
> package_output)) continue;
1919         $this-> subpackage = $pages[$j]-> classelements[$i]-> docblock-
> subpackage;
1920         $this-> class = $pages[$j]-> classelements[$i]-> name;
1921       } else
1922       {
1923         $a = $pages[$j]-> classelements[$i]-> docblock-
> getKeyword('access');
1924         if (is_object($a)) $a = $a-> getString();
1925         if (!$this-> parseprivate && ( $a == 'private'))
1926           continue;
1927         if ($this-> killclass) continue;
1928         // force all contained elements to have parent package/subpackage
1929         $pages[$j]-> classelements[$i]-> docblock-> package = $this-
> package;
1930         $pages[$j]-> classelements[$i]-> docblock-> subpackage = $this-
> subpackage;
1931       }
1932     }
1933     if ($pages[$j]-> classelements[$i]-> type == 'class')
1934     {
1935       $this-> killclass = false;
1936       if ($this-> checkKillClass($pages[$j]-> classelements[$i]-
> getName(),$pages[$j]-> classelements[$i]-> getPath())) continue;
1937       $this-> package = $pages[$j]-> classelements[$i]-> docblock-
> package;
1938       if ($this-> package_output) if (!in_array($this-> package,$this-
> package_output)) continue;
1939       $this-> subpackage = $pages[$j]-> classelements[$i]-> docblock-
> subpackage;
1940       $this-> class = $pages[$j]-> classelements[$i]-> name;
1941     }
1942     if ($this-> killclass) continue;
1943     phpDocumentor_out("    ".$pages[$j]-> classelements[$i]-
>name."\n");
1944     $this-> Convert($pages[$j]-> classelements[$i]);
1945   }
1946   if (count($pages[$j]-> classelements) && !$this-> killclass) $this-
> endClass();
1947   phpDocumentor_out(" done\n");
1948   flush();
1949   if (count($this-> todoList))
1950   {
1951     $this-> ConvertTodoList();
1952   }
1953   phpDocumentor_out("done\n");
1954   flush();
}

```

```

1955
1956
1957 /**
1958 * Get a tree structure representing the hierarchy of tutorials
1959 *
1960 * Returns an array in format:
1961 * <pre>
1962 * array('tutorial' => { @link parserTutorial},
1963 *       'kids' => array( // child tutorials
1964 *           array('tutorial' => child { @link parserTutorial},
1965 *                 'kids' => array(...))
1966 *             )
1967 *         )
1968 *     )
1969 * </pre>
1970 * @param parserTutorial/array
1971 * @tutorial tutorials.pkg
1972 * @return array
1973 */
1974 function getTutorialTree($tutorial)
1975 {
1976     if (is_object($tutorial))
1977     {
1978         $path = $this-> _tutorial_path($tutorial,$tutorial,$tutorial);
1979         if (isset($this-> tutorial_tree[$path])) {
1980             $tutorial = $this-> tutorial_tree[$path];
1981         } else {
1982             return false;
1983         }
1984     }
1985     $tree = array();
1986     if (isset($tutorial['tutorial']))
1987     {
1988         $tree['tutorial'] = $tutorial['tutorial'];
1989         if (isset($tutorial['child']))
1990         {
1991             foreach($tutorial['child'] as $a => $b)
1992             {
1993                 $btut = $b['tutorial'];
1994                 $res = array(
1995                     'tutorial' => $this-> tutorials
1996                         [$btut-> package][$btut-> subpackage]
1997                         [$btut-> tutorial_type][$btut-> name]
1998                 );
1999                 if (isset($b['child']))
2000                 {
2001                     $tempres = Converter::getTutorialTree($b);
2002                     $res['kids'] = $tempres['kids'];
2003                 }
2004                 $tree['kids'][] = $res;
2005             }
2006         }
2007     }
2008     return $tree;
2009 }
2010
2011 /**
2012 * Remove tutorials one by one from $all, and transfer them into $new in the
2013 * order they should be parsed
2014 * @param parserTutorial
2015 * @param array
2016 * @param array
2017 * @access private
2018 */
2019 function generateTutorialOrder($parent,& $all,& $new)
2020 {
2021     // remove from the list of tutorials to process
2022     foreach($all[$parent-> package][$parent-> subpackage][$parent-> tutorial_type] as
2023     $t)
2024     {
2025         if ($t-> name == $parent-> name) {
2026             unset($all[$parent-> package][$parent-> subpackage][$parent-> tutorial_type][$ind]);
2027         }
2028         // add to the new ordered list of tutorials
2029         $x = & $new[$parent-> package][$parent-> subpackage][$parent-> tutorial_type];
2030         if (!is_object($x[count($x) - 1]) || $x[count($x) - 1]-> name != $parent-> name)
2031         { // only add if the parent isn't also a child
2032             $new[$parent-> package][$parent-> subpackage][$parent-> tutorial_type][] =

```

```

$parent;
2033         // add a new branch to the tree
2034     }
2035     // process all child tutorials, and insert them in order
2036     // debug("processing parent ".$parent->name);
2037     if ($parent-> ini)
2038     {
2039         foreach($parent-> ini['Linked Tutorials'] as $child)
2040         {
2041             $sub = (empty($parent-> subpackage) ? '' : $parent-> subpackage . '/');
2042             $kid = $parent-> package . '/' . $sub . $child . '' . $parent-
2043             > tutorial_type;
2044             $klink = $this-> getTutorialLink($kid, false, false, array($parent-
2045             > package));
2046             if (is_object($klink))
2047             {
2048                 $klink = $this-> returnSee($klink);
2049             } else {
2050                 $klink = false;
2051             }
2052             // remove the child from the list of remaining tutorials
2053             foreach($all[$parent-> package][$parent-> subpackage][$parent-
2054             > tutorial_type] as $ind => $tute)
2055             {
2056                 if ($klink && $tute-> getLink($this) == $klink)
2057                 {
2058                     // set up parent, next and prev links
2059                     $tute-> setParent($parent, $this);
2060                     // remove the child from the list of tutorials to process
2061                     foreach($all[$parent-> package][$parent-> subpackage][$parent-
2062             > tutorial_type] as $ind => $t)
2063                     {
2064                         if ($t-> name == $tute-> name)
2065                             unset($all[$parent-> package][$parent-> subpackage][$parent-
2066             > tutorial_type][$ind]);
2067                     }
2068                     // add to the new ordered list of tutorials
2069                     $new[$parent-> package][$parent-> subpackage][$parent-
2070             > tutorial_type][] = $tute;
2071                     if ($tute-> ini)
2072                     {
2073                         // add all the child's child tutorials to the list
2074                         $this-> generateTutorialOrder($tute, $all, $new);
2075                     }
2076                 }
2077             }
2078         }
2079         return;
2080     }
2081     /**
2082      * Returns the path to this tutorial as a string
2083      * @param parserTutorial $pkg
2084      * @param parserTutorial $subpkg
2085      * @param parserTutorial $namepkg
2086      * @return string */
2087     function _tutorial_path($pkg, $subpkg = 0, $namepkg = 0)
2088     {
2089         if (!$subpkg)
2090             $subpkg = $pkg;
2091         if (!$namepkg)
2092             $namepkg = $pkg;
2093         $subpackagename = ($subpkg-> subpackage ? '/' . $subpkg-> subpackage : '');
2094         return $pkg-> package . $subpackagename . '/' . $namepkg-> name;
2095     }
2096     /**
2097      * Creates a tree structure of tutorials
2098      *
2099      * Format:
2100      * <pre>
2101      * array('package/subpackage/tutorial1.ext' =>
2102      *       array('tutorial' => {@link parserTutorial},
2103      *             'child'   =>
2104      *                   array('package/subpackage/child1tutorial.ext' => ...,
2105      *                         'package/subpackage/child2tutorial.ext' => ...,
2106      *                         ...
2107      *                   )
2108      *     )

```

```

2106     *      'package/subpackage/tutorial2.ext' => ...,
2107     *      ...
2108     *  )
2109     * </pre>
2110     * @return array the tutorial tree
2111     * @access private
2112     */
2113 function _setupTutorialTree($parent = false)
2114 {
2115     if (! isset($this-> processed_tutorials)) {
2116         $this-> processed_tutorials = array();
2117     }
2118     $tree = array();
2119     if (!$parent)
2120     {
2121         foreach($this-> tutorials as $package => $s)
2122         {
2123             foreach($s as $subpackage => $t)
2124             {
2125                 foreach($t as $type => $n)
2126                 {
2127                     foreach($n as $name => $tutorial)
2128                     {
2129                         if ($tutorial-> parent) {
2130                             continue;
2131                         }
2132
2133                         $child_path = $this-
2134 > _tutorial_path($tutorial,$tutorial,$tutorial);
2135                         if (isset($this-> processed_tutorials[$child_path])) {
2136                             continue;
2137                         }
2138                         $this-> processed_tutorials[$child_path] = $tutorial;
2139                         //debug("parent ".$tutorial->name);
2140                         $ret = $this-> _setupTutorialTree($tutorial);
2141                         if (!count($tree)) {
2142                             $tree = $ret;
2143                         } else {
2144                             $tree = array_merge($tree,$ret);
2145                         }
2146                     }
2147                 }
2148             }
2149             return $tree;
2150         }
2151         $parent_path = $this-> _tutorial_path($parent);
2152         $tree[$parent_path]['tutorial'] = $parent;
2153         // process all child tutorials, and insert them in order
2154         if ($parent-> ini)
2155         {
2156             foreach($parent-> ini['Linked Tutorials'] as $child)
2157             {
2158                 if (isset($this-> tutorials[$parent-> package][$parent-> subpackage]
2159                           [$parent-> tutorial_type][$child . '.' .
2160                           $parent-> tutorial_type])) {
2161                     // remove the child from the list of remaining tutorials
2162                     $tute = $this-> tutorials[$parent-> package][$parent-> subpackage]
2163                           [$parent-> tutorial_type][$child . '.' .
2164                           $parent-> tutorial_type];
2165                 } else {
2166                     $tute = false;
2167                 }
2168
2169                 if (!$tute) {
2170                     continue;
2171                 }
2172                 $child_path = $this-> _tutorial_path($parent,$parent,$tute);
2173                 if (isset($this-> processed_tutorials[$child_path])) {
2174                     continue;
2175                 }
2176                 $this-> processed_tutorials[$child_path] = $tute;
2177                 if ($tute-> name != $child . '.' . $parent-> tutorial_type) {
2178                     continue;
2179                 }
2180                 //echo "Adding [$child_path] to [$parent_path]<br>";
2181                 $tree[$parent_path]['child'][$this-
2182 > _tutorial_path($parent,$parent,$tute)]['tutorial']
2183                     = $tute;
2184                     if (!$tute-> ini) {

```

```

2184         continue;
2185     }
2186     // add all the child's child tutorials to the list
2187     if (!isset($tree[$parent_path]['child'])) {
2188         $tree[$parent_path]['child'] = $this-> _setupTutorialTree($tute);
2189     } else {
2190         $tree[$parent_path]['child'] = array_merge($tree[$parent_path]['child'],
2191             $this-> _setupTutorialTree($tute));
2192     }
2193 }
2194 return $tree;
2195 }
2196
2197 /**
2198 * Debugging function for dumping {@link $tutorial_tree}
2199 * @return string
2200 */
2201 function vardump_tree($tree,$indent='')
2202 {
2203     if (phpDocumentor_get_class($tree) == 'parsertutorial') return $tree-> name.' extends
2204     '. $tree-> parent? $tree-> parent-> name : 'nothing');
2205     $a = '';
2206     foreach($tree as $ind => $stuff)
2207     {
2208         $x = $this-> vardump_tree($stuff, " $indent      ");
2209         $a .= $indent.'['.$ind.' => \n      ' . $indent.$x." ]\n";
2210     }
2211     return substr($a,0,strlen($a) - 1);
2212 }
2213
2214 /**
2215 * @access private
2216 */
2217 function sort_package_elements($a,$b)
2218 {
2219     if (($a-> type == $b-> type) && (isset( $a-> isConstructor) && $a-
2220 > isConstructor)) return -1;
2221     if (($a-> type == $b-> type) && (isset( $b-> isConstructor) && $b-
2222 > isConstructor)) return 1;
2223     if ($a-> type == $b-> type) return strnatcasecmp($a-> name,$b-> name);
2224     if ($a-> type == 'class') return -1;
2225     if ($b-> type == 'class') return 1;
2226     if ($a-> type == 'const') return -1;
2227     if ($b-> type == 'const') return 1;
2228     if ($a-> type == 'var') return -1;
2229     if ($b-> type == 'var') return 1;
2230     if ($a-> type == 'page') return -1;
2231     if ($b-> type == 'page') return 1;
2232     if ($a-> type == 'include') return -1;
2233     if ($b-> type == 'include') return 1;
2234     if ($a-> type == 'define') return -1;
2235     if ($b-> type == 'define') return 1;
2236     if ($a-> type == 'global') return -1;
2237     if ($b-> type == 'global') return 1;
2238 }
2239
2240 /**
2241 * @access private
2242 */
2243 function defpackagesort($a,$b)
2244 {
2245     if ($a == $GLOBALS['phpDocumentor_DefaultPackageName']) return -1;
2246     if ($b == $GLOBALS['phpDocumentor_DefaultPackageName']) return 0;
2247     return strnatcasecmp($a,$b);
2248 }
2249
2250 /**
2251 * @access private
2252 */
2253 function Pc_sort($a,$b)
2254 {
2255     return strnatcasecmp(key($a),key($b));
2256 }
2257
2258 /**
2259 * walk over elements by package rather than page
2260 */

```



```

2337             if ($is_object($a)) $a = $a-> getString();
2338             if (!$this-> parseprivate && ($a == 'private'))
2339                 continue;
2340         }
2341         if ($notyet)
2342         {
2343             phpDocumentor_out(" Procedural Page Elements..." );
2344             flush();
2345             $notyet = false;
2346         }
2347         $this-> Convert($element);
2348     }
2349 }
2350 $this-> endPage();
2351 phpDocumentor_out("done\n");
2352 flush();
2353 }
2354
2355 $start_classes = true;
2356 if (isset($r['class']))
2357 {
2358     foreach($r['class'] as $class => $elements)
2359     {
2360         foreach($elements as $element)
2361         {
2362             if ($element-> type == 'class')
2363             {
2364                 if (!$start_classes)
2365                 {
2366                     if (count($elements) && !$this-> killclass) $this-
2367                         > endClass();
2368
2369                     phpDocumentor_out("done\n");
2370                     flush();
2371
2372                     $start_classes = false;
2373                     $this-> class = $element-> getName();
2374                     $this-> killclass = false;
2375                     if ($this-> checkKillClass($element-> getName(), $element-
2376                         > getPath())) continue;
2377
2378                     if (!$this-> killclass)
2379                     {
2380                         phpDocumentor_out('Converting '.$this-
2381                             > class."...");
2382
2383                     } else
2384                     {
2385                         if ($notyet)
2386                         {
2387                             phpDocumentor_out("Variables/methods/Class
2388                             flush();
2389                             $notyet = false;
2390                         }
2391                         $a = $element-> docblock-> getKeyword('access');
2392                         if ($is_object($a)) $a = $a-> getString();
2393                         if (!$this-> parseprivate && ($a == 'private'))
2394                             continue;
2395                         if ($this-> killclass) continue;
2396                         // force all contained elements to have parent
2397                         package/subpackage
2398                         $element-> docblock-> package = $this-> package;
2399                         $element-> docblock-> subpackage = $this-> subpackage;
2400
2401                         if ($this-> killclass) continue;
2402                         $this-> Convert($element);
2403
2404                     }
2405                     if (count($elements) && !$this-> killclass) $this-> endClass();
2406                     phpDocumentor_out("done\n");
2407                     flush();
2408                     if (count($this-> todoList))
2409                     {
2410                         $this-> ConvertTodoList();

```

```

2412     }
2413     phpDocumentor_out("done\n") ;
2414     flush();
2415     phpDocumentor_out("\\nConverting Error Log...") ;
2416     flush();
2417     $this-> ConvertErrorLog();
2418     phpDocumentor_out("done\\n") ;
2419     flush();
2420 }
2421 /**
2422 * Convert the phpDocumentor parsing/conversion error log
2423 * @abstract
2424 */
2425 function ConvertErrorLog()
2426 {
2427 }
2428 /**
2429 * Convert the list of all @todo tags
2430 * @abstract
2431 */
2432 function ConvertTodoList()
2433 {
2434 }
2435 /**
2436 * Sorts the @todo list - do not override or modify this function
2437 * @access private
2438 * @uses _sortTodos passed to {@link usort()} to sort the todo list
2439 */
2440 function sortTodos()
2441 {
2442     phpDocumentor_out("\\nSorting @todo list...") ;
2443     flush();
2444     foreach($this-> todoList as $package => $r) {
2445         usort($this-> todoList[$package], array('Converter', '_sortTodoPackage'));
2446         foreach ($r as $a => $sub) {
2447             if (is_array($this-> todoList[$package][$a][1])) {
2448                 usort($this-> todoList[$package][$a][1],array('Converter', '_sortTodos'));
2449             }
2450         }
2451     }
2452     phpDocumentor_out("done\\n") ;
2453 }
2454 /**
2455 * @access private
2456 */
2457 function _sortTodoPackage($a, $b)
2458 {
2459     return strnatcasecmp($a[0]-> name, $b[0]-> name);
2460 }
2461 /**
2462 * @access private
2463 */
2464 function _sortTodos($a, $b)
2465 {
2466     if (!is_object($a)) {
2467         var_dump($a);
2468     }
2469     return strnatcasecmp($a-> getString(), $b-> getString());
2470 }
2471 /**
2472 * Sorts all indexes - do not override or modify this function
2473 * @uses $leftindex based on the value of leftindex, sorts link arrays
2474 * @uses $class_elements sorts with {@link compareLink}
2475 * @uses $page_elements sorts with {@link compareLink}
2476 * @uses $define_elements sorts with {@link compareLink}
2477 * @uses $global_elements sorts with {@link compareLink}
2478 * @uses $function_elements sorts with {@link compareLink}
2479 * @uses $elements sorts with {@link elementCmp}
2480 * @uses $pkg_elements sorts with {@link elementCmp} after sorting by
2481 * package/subpackage alphabetically
2482 * @access private
2483 */
2484 function sortIndexes()
2485 {
2486     phpDocumentor_out("\\nSorting Indexes...") ;
2487     flush();
2488     uksort($this-> elements,'strnatcasecmp');
2489     if ($this-> leftindex['classes'])
2490     {
2491     }
2492 }
2493 
```

```

2492     {
2493         foreach($this-> class_elements as $package => $o1)
2494         {
2495             foreach($o1 as $subpackage => $links)
2496             {
2497                 usort($this-
> class_elements[$package][$subpackage],array($this,'compareLink'));
2498             }
2499         }
2500     }
2501     if ($this-> leftindex['pages'])
2502     {
2503         foreach($this-> page_elements as $package => $o1)
2504         {
2505             uksort($this-> page_elements[$package],'strnatcasecmp');
2506             foreach($o1 as $subpackage => $links)
2507             {
2508                 usort($this-
> page_elements[$package][$subpackage],array($this,'compareLink'));
2509             }
2510         }
2511     }
2512     if ($this-> leftindex['defines'])
2513     {
2514         foreach($this-> define_elements as $package => $o1)
2515         {
2516             uksort($this-> define_elements[$package],'strnatcasecmp');
2517             foreach($o1 as $subpackage => $links)
2518             {
2519                 usort($this-
> define_elements[$package][$subpackage],array($this,'compareLink'));
2520             }
2521         }
2522     }
2523     if ($this-> leftindex['globals'])
2524     {
2525         foreach($this-> global_elements as $package => $o1)
2526         {
2527             uksort($this-> global_elements[$package],'strnatcasecmp');
2528             foreach($o1 as $subpackage => $links)
2529             {
2530                 usort($this-
> global_elements[$package][$subpackage],array($this,'compareLink'));
2531             }
2532         }
2533     }
2534     if ($this-> leftindex['functions'])
2535     {
2536         foreach($this-> function_elements as $package => $o1)
2537         {
2538             uksort($this-> function_elements[$package],'strnatcasecmp');
2539             foreach($o1 as $subpackage => $links)
2540             {
2541                 usort($this-
> function_elements[$package][$subpackage],array($this,'compareLink'));
2542             }
2543         }
2544     }
2545     foreach($this-> elements as $letter => $nothuing)
2546     {
2547         uasort($this-> elements[$letter],array($this,"elementCmp"
));
2548     }
2549     foreach($this-> pkg_elements as $package => $els)
2550     {
2551         uksort($this-> pkg_elements[$package],'strnatcasecmp');
2552         foreach($this-> pkg_elements[$package] as $subpackage => $els)
2553         {
2554             if (empty($els)) continue;
2555             uksort($this-> pkg_elements[$package][$subpackage],'strnatcasecmp');
2556             foreach($els as $letter => $yuh)
2557             {
2558                 usort($this-
> pkg_elements[$package][$subpackage][$letter],array($this,"elementCmp"
));
2559             }
2560         }
2561     }
2562     phpDocumentor_out("done\n");
2563     flush();
2564 }
2565

```

```

2566 /**
2567 * sorts {@link $page_contents} by element type as well as alphabetically
2568 * @see $sort_page_contents_by_element_type
2569 */
2570 function sortPageContentsByElementType(& $pages)
2571 {
2572     foreach($this-> page_contents as $package => $els)
2573     {
2574         foreach($this-> page_contents[$package] as $subpackage => $els)
2575         {
2576             if (empty($els)) continue;
2577             foreach($this-> page_contents[$package][$subpackage] as $path => $stuff)
2578             {
2579                 if (!count($pages[$path]-> elements)) continue;
2580                 usort($pages[$path]-> elements, array($this, 'eltypecmp'));
2581                 usort($this-
2582 > page_contents[$package][$subpackage][$path], array($this, 'eltypecmp'));
2583                 if (isset($this-> page_contents[$package][$subpackage][$path][0]))
2584                     $this-> page_contents[$package][$subpackage][$path]['##main'] = $this-
2585 > page_contents[$package][$subpackage][$path][0];
2586                 unset($this-> page_contents[$package][$subpackage][$path][0]);
2587             }
2588         }
2589     }
2590 /**
2591 * @access private
2592 * @see Converter::sortIndexes()
2593 */
2594 function compareLink($a, $b)
2595 {
2596     return strnatcasecmp($a-> name,$b-> name);
2597 }
2598 /**
2599 * @access private
2600 * @see Converter::sortPageContentsByElementType()
2601 */
2602 function eltypecmp($a, $b)
2603 {
2604     if ($a-> type == 'page') return -1;
2605     if ($b-> type == 'page') return 1;
2606     return strnatcasecmp($a-> type.$a-> name,$b-> type.$b-> name);
2607 }
2608 /**
2609 * does a nat case sort on the specified second level value of the array
2610 *
2611 * @param mixed $a
2612 * @param mixed $b
2613 * @return int
2614 * @access private
2615 */
2616 function elementCmp ($a, $b)
2617 {
2618     return strnatcasecmp($a-> getName(), $b-> getName());
2619 }
2620 /**
2621 * Used to stop conversion of @ignored or private @access classes
2622 * @uses $killclass sets killclass based on the value of {@link Classes::$killclass}
2623 * and {@link $package_output}
2624 * @access private
2625 */
2626 function checkKillClass($class, $path)
2627 {
2628     $this-> killclass = false;
2629     if (isset($this-> classes-> killclass[$class]) && !isset(
2630 > killclass[$class][$path])) $this-> killclass = true;
2631     if ($this-> package_output)
2632     {
2633         $a = $this-> classes-> getClass($class, $path);
2634         if (!in_array($a-> docblock-> package,$this-> package_output)) $this-
2635 > killclass = true;
2636     }
2637     if (PHPDOCUMENTOR_DEBUG && $this-> killclass) debug(" $class $path
2638 killed");
2639     return $this-> killclass;
2640 }

```

```

2641
2642 /**
2643 * @param abstractLink descendant of abstractLink
2644 * @param array/parserTaglist of @todos/@todo tag
2645 * @access private
2646 */
2647 function addTodoLink($link, $todos)
2648 {
    $this-> todoList[$link-> package][] = array($link, $todos);
}
2651
2652 /**
2653 * Adds all elements to the {@link $elements, $pkg_elements, $links},
2654 * {@link $linkswithfile} and left indexes - Do not modify or override
2655 * @access private
2656 * @param parserBase any documentable element descendant of parserBase
2657 * except parserTutorial
2658 * @param false/parserPageonly used to add a {@link parserPage} if the
2659 * $element passed is a parserPage
2660 * @staticvar string path of current page, used for {@link $page_contents} setup
2661 */
2662 function addElement(& $element,$pageel=false)
2663 {
    static $curpath = '';
    if ($this-> package_output)
    {
        if (!in_array($this-> package, $this-> package_output)) return;
    }
    if ($pageel && phpDocumentor_get_class($pageel) == 'parserdata')
    {
        if (isset($pageel-> docblock) && phpDocumentor_get_class($pageel-> docblock) == 'parserdocblock')
        {
            $a = $pageel-> docblock-> getKeyword('todo');
            if ($a)
            {
                $this-> addTodoLink($this-> addLink($element),$a);
            }
        }
        if (isset($element-> docblock))
        {
            $a = $element-> docblock-> getKeyword('access');
            if (is_object($a)) $a = $a-> getString();
            if (!$this-> parseprivate && ($a == 'private'))
                return;
            $a = $element-> docblock-> getKeyword('todo');
            if ($a)
            {
                if ($element-> type != 'include') {
                    $this-> addTodoLink($this-> addLink($element),$a);
                } else {
                    addWarning(PDERROR_NOTODO_INCLUDE, $element-> getLineNumber(),
                               $element-> getPath());
                }
            }
        }
        $StartPositionOfElementName = 0; // which character of the element name actually
        starts its textual name
        switch($element-> type)
        {
            case 'page' :
                if ($this-> sort_absolutely_everything)
                {
                    $this-> package_elements[$element-> package][$element-> subpackage]['page'][$element-> getPath()][] = $pageel;
                }
                $link = $this-> addLink($element);
                $curpath = $element-> getPath();
                if ($this-> leftindex['pages'])
                    $this-> page_elements[$element-> package][$element-> subpackage][] =
$link;
                $this-> page_contents[$element-> package][$element-> subpackage][$curpath]['##main'] = $link;
                break;
            case 'class' :
                if ($this-> sort_absolutely_everything)
                {
                    $this-> package_elements[$element-> docblock-> package][$element-> subpackage][$element-> class] = $element;
                }
        }
    }
}

```

```

2715         }
2716         $link = $this-> addLink($element);
2717         if ($this-> leftIndex['classes'])
2718             $this-> class_elements[$element-> docblock-> package][$element-
2719             subpackage][] = $link;
2720             $this-> class_contents[$element-> docblock-> package][$element-
2721             break;
2722             case 'include' :
2723                 if ($this-> sort_absolutely_everything)
2724                 {
2725                     $this-> package_elements[$element-> docblock-> package][$element-
2726                     subpackage]['page'][$curpath][] = $element;
2727                     $link = $this-> addLink($element);
2728                     break;
2729                     case 'define' :
2730                         if ($this-> sort_absolutely_everything)
2731                         {
2732                             $this-> package_elements[$element-> docblock-> package][$element-
2733                             subpackage]['page'][$curpath][] = $element;
2734                             $link = $this-> addLink($element);
2735                             if ($this-> leftIndex['defines'])
2736                                 $this-> define_elements[$element-> docblock-> package][$element-
2737                                 subpackage][] = $link;
2738                                 $this-> page_contents[$element-> docblock-> package][$element-
2739                                 subpackage][$curpath][] = $link;
2740                                 break;
2741                                 case 'global' :
2742                                     if ($this-> sort_absolutely_everything)
2743                                     {
2744                                         $this-> package_elements[$element-> docblock-> package][$element-
2745                                         subpackage]['page'][$curpath][] = $element;
2746                                         $link = $this-> addLink($element);
2747                                         $startPositionOfElementName = 1; // lose the leading "$" character
2748                                         if ($this-> leftIndex['globals'])
2749                                             $this-> global_elements[$element-> docblock-> package][$element-
2750                                             subpackage][] = $link;
2751                                             $this-> page_contents[$element-> docblock-> package][$element-
2752                                             subpackage][$curpath][] = $link;
2753                                             break;
2754                                             case 'var' :
2755                                                 if ($this-> sort_absolutely_everything)
2756                                                 {
2757                                                     $this-> package_elements[$element-> docblock-> package][$element-
2758                                                     subpackage]['class'][$this-> class][] = $element;
2759                                                     $link = $this-> addLink($element);
2760                                                     $startPositionOfElementName = 1; // lose the leading "$" character
2761                                                     $this-> class_contents[$element-> docblock-> package][$element-
2762                                                     subpackage][$this-> class][] = $link;
2763                                                     break;
2764                                                     case 'const' :
2765                                                         if ($this-> sort_absolutely_everything)
2766                                                         {
2767                                                             $this-> package_elements[$element-> docblock-> package][$element-
2768                                                             subpackage]['class'][$this-> class][] = $element;
2769                                                             $link = $this-> addLink($element);
2770                                                             $this-> class_contents[$element-> docblock-> package][$element-
2771                                                             subpackage][$this-> class][] = $link;
2772                                                             break;
2773                                                             case 'method' :
2774                                                 if ($this-> sort_absolutely_everything)
2775                                                 {
2776                                                     $this-> package_elements[$element-> docblock-> package][$element-
2777                                                     subpackage]['class'][$this-> class][] = $element;
2778                                                     $link = $this-> addLink($element);
2779                                                     $this-> class_contents[$element-> docblock-> package][$element-
2780                                                     subpackage][$this-> class][] = $link;
2781                                                     break;
2782                                                     case 'function' :
2783                                                         if ($this-> sort_absolutely_everything)
2784                                                         {
2785                                                             $this-> package_elements[$element-> docblock-> package][$element-
2786                                                             subpackage]['page'][$curpath][] = $element;
2787                                                         }

```

```

2779         $link = $this-> addLink($element);
2780         if ($this-> leftIndex['functions'])
2781             $this-> functionElements[$element-> docblock-> package][$element-
> docblock-> subpackage][] = $link;
2782             $this-> pageContents[$element-> docblock-> package][$element-
> docblock-> subpackage][$curpath][] = $link;
2783             break;
2784         default :
2785             break;
2786     }
2787     if ($element-> getType() != 'include')
2788     {
2789         if ($element-> getType() == 'var' || $element-> getType() == 'method' ||
$element-> getType() == 'const')
2790         {
2791             $this-> links[$this-> package][$this-> subpackage][$element-
> getType()][$element-> class][$element-> getName()] = $link;
2792             $this-> linkswithfile[$this-> package][$this-> subpackage][$element-
> getType()][$element-> getPath()][$element-> class][$element-> getName()] = $link;
2793         } else
2794         {
2795             if ($element-> type == 'page')
2796             {
2797                 $this-> links[$this-> package][$this-> subpackage][$element-
> getType()][$element-> getFile()] = $link;
2798                 $this-> linkswithfile[$this-> package][$this-> subpackage][$element-
> getType()][$element-> getPath()][$element-> getFile()] = $link;
2799             } else
2800             {
2801                 $this-> links[$this-> package][$this-> subpackage][$element-
> getType()][$element-> getName()] = $link;
2802                 $this-> linkswithfile[$this-> package][$this-> subpackage][$element-
> getType()][$element-> getPath()][$element-> getName()] = $link;
2803             }
2804         }
2805     }
2806     if ($element-> type == 'page')
2807     {
2808         $this-> elements[substr(strtolower($element-
> getFile()),$startPositionOfElementName,1)][] = $element;
2809         $this-> pkgElements[$this-> package][$this-
> subpackage][substr(strtolower($element-> getFile()),$startPositionOfElementName,1)][] =
$element;
2810     } else
2811     {
2812         $this-> elements[substr(strtolower($element-
> getName()),$startPositionOfElementName,1)][] = $element;
2813         $this-> pkgElements[$this-> package][$this-
> subpackage][substr(strtolower($element-> getName()),$startPositionOfElementName,1)][] =
$element;
2814     }
2815 }
2816 /**
2817 * returns an abstract link to element. Do not modify or override
2818 *
2819 * This method should only be called in process of Conversion, unless
2820 * $element is a parserPage, or $page is set to true, and $element is
2821 * not a parserPage
2822 * @return abstractLink abstractLink descendant
2823 * @access private
2824 * @param parserElement element to add a new link (descended from
2825 *          {@link abstractLink}) to the {@link $links} array
2826 * @param string classname for elements that are class-based (this may be
2827 *          deprecated in the future, as the classname
2828 *          should be contained within the element. if $element is a
2829 *          page, this parameter is a package name
2830 * @param string subpackage name for page elements
2831 */
2832
2833 function addLink(& $element,$page = false)
2834 {
2835     if ($page)
2836     {
2837         // create a fake parserPage to extract the fileAlias for this link
2838         $fakepage = new parserPage;
2839         $fakepage-> setPath($element-> getPath());
2840         $fakepage-> setFile(basename($element-> getPath()));
2841         $this-> curname = $this-> getPageName($fakepage);
2842     }
2843     switch($element-> type)

```

```

2844     {
2845         case 'function':
2846             $x = new functionLink;
2847             $x-> addLink($element-> getPath(), $this-> curname, $element-> name,
2848             $element-> docblock-> package, $element-> docblock-> subpackage, $element-> docblock-
2849             > category);
2850             return $x;
2851         break;
2852         case 'define':
2853             $x = new defineLink;
2854             $x-> addLink($element-> getPath(), $this-> curname, $element-> name,
2855             $element-> docblock-> package, $element-> docblock-> subpackage, $element-> docblock-
2856             > category);
2857             return $x;
2858         break;
2859         case 'global':
2860             $x = new globalLink;
2861             $x-> addLink($element-> getPath(), $this-> curname, $element-> name,
2862             $element-> docblock-> package, $element-> docblock-> subpackage, $element-> docblock-
2863             > category);
2864             return $x;
2865         break;
2866         case 'class':
2867             $x = new classLink;
2868             $x-> addLink($element-> getPath(), $this-> curname, $element-> name,
2869             $element-> docblock-> package, $element-> docblock-> subpackage, $element-> docblock-
2870             > category);
2871             return $x;
2872         break;
2873         case 'method':
2874             $x = new methodLink;
2875             $x-> addLink($this-> class, $element-> getPath(), $this-> curname,
2876             $element-> name, $element-> docblock-> package, $element-> docblock-> subpackage, $element-
2877             > docblock-> category);
2878             return $x;
2879         break;
2880         case 'var':
2881             $x = new varLink;
2882             $x-> addLink($this-> class, $element-> getPath(), $this-> curname,
2883             $element-> name, $element-> docblock-> package, $element-> docblock-> subpackage, $element-
2884             > docblock-> category);
2885             return $x;
2886         break;
2887         case 'const':
2888             $x = new constLink;
2889             $x-> addLink($this-> class, $element-> getPath(), $this-> curname,
2890             $element-> name, $element-> docblock-> package, $element-> docblock-> subpackage, $element-
2891             > docblock-> category);
2892             return $x;
2893         break;
2894         case 'page':
2895             $x = new pageLink;
2896             $x-> addLink($element-> getPath(), $this-> getPageName($element), $element-
2897             > file, $element-> package, $element-> subpackage, $element-> category);
2898             return $x;
2899         break;
2900     }
2901 }
2902 /**
2903 * Return a tree of all classes that extend this class
2904 *
2905 * The data structure returned is designed for a non-recursive algorithm,
2906 * and is somewhat complex.
2907 * In most cases, the array returned is:
2908 *
2909 * <pre>
2910 * array('#root' =>
2911 *     array('link' => {@link classLink} to $class,
2912 *           'parent' => false,
2913 *           'children' => array(array('class' => 'childclass1',
2914 *                                     'package' => 'child1package'),
2915 *                               array('class' => 'childclass2',
2916 *                                     'package' => 'child2package'), ...
2917 *                               ),
2918 *           'child1package#childclass1' =>
2919 *               array('link' => {@link classLink} to childclass1,
2920 *                     'parent' => '#root',
2921 *                     'children' => array(array('class' => 'kidclass',

```

```

2909 *                               'package' => 'kidpackage'),...
2910 *
2911 *           ),
2912 *           'kidpackage#kidclass' =>
2913 *             array('link' => '{@link classLink} to kidclass,
2914 *                   'parent' => 'childpackage#childclassname',
2915 *                   'children' => array() // no children
2916 *             ),
2917 *
2918 *           ....
2919 *         )
2920 */
2921 * To describe this format using language, every class in the tree has an
2922 * entry in the first level of the array. The index for all child
2923 * classes that extend the root class is childpackage#childclassname.
2924 * Each entry in the array has 3 elements: link, parent, and children.
2925 * <ul>
2926 *   <li>link - a      {@link classLink} to the current class</li>
2927 *   <li>parent - a      {@link classLink} to the class's parent, or false (except for one
2928 * special case described below)</li>
2929 *   <li>children - an array of arrays, each entry has a 'class' and 'package' index
2930 * to the child class,
2931 * used to find the entry in the big array</li>
2932 * </ul>
2933 *
2934 * special cases are when the #root class has a parent in another package,
2935 * or when the #root class extends a class not found
2936 * by phpDocumentor. In the first case, parent will be a
2937 * classLink to the parent class. In the second, parent will be the
2938 * extends clause, as in:
2939 * <code>
2940 * class X extends Y
2941 * {
2942 * ...
2943 * </code>
2944 * in this case, the #root entry will be array('link' => classLink to X, 'parent' =>
2945 * 'Y', 'children' => array(...))
2946 *
2947 * The fastest way to design a method to process the array returned
2948 * is to copy HTMLframesConverter::getRootTree() into
2949 * your converter and to modify the html to whatever output format you are going to use
2950 * @see HTMLframesConverter::getRootTree()
2951 * @param string class name
2952 * @param string
2953 * @return array Format: see docs
2954 */
2955 function getSortedClassTreeFromClass($class,$package,$subpackage)
2956 {
2957     $my_tree = array();
2958     $root = $this-> classes-> getClassByPackage($class,$package);
2959     if (!$root) return false;
2960     $class_children = $this-> classes-> getClassChildren($class,$root-> curfile);
2961     if (!$class_children)
2962     {
2963         // special case: parent class is found, but is not part of this package, class has
2964         // no children
2965         if (is_array($root-> parent))
2966         {
2967             $x = $root-> getParent($this);
2968             if ($x-> docblock-> package != $package)
2969             {
2970                 $v = Converter::getClassLink($root-> getName(),$package,$root-
2971 > getPath());
2972                 return array('#root' => array( 'link' => $v,'parent' =>
2973 Converter::getClassLink($x-> getName(),$x-> docblock-> package,$x-> getPath()), 'children' => array()));
2974             }
2975         }
2976     }
2977     else
2978     {
2979         // class has normal situation, no children
2980         if (is_string($root-> getParent($this)))
2981             return array('#root' => array( 'link' => Converter::getClassLink($root-
2982 > getName(),$package,$root-> getPath()), 'parent' => $root-> getExtends(),'children' =>
2983 array()));
2984         else
2985             return array('#root' => array( 'link' => Converter::getClassLink($root-
2986 > getName(),$package,$root-> getPath()), 'parent' => false, 'children' => array()));
2987     }
2988 }

```

```

2979         // special case: parent class is found, but is not part of this package, class has
2980         children
2981     if (is_array($root-> parent))
2982     {
2983         $x = $root-> getParent($this);
2984         if ($x-> docblock-> package != $package)
2985         {
2986             $v = Converter::getClassLink($root-> getName(),$package,$root-> getPath());
2987             $my_tree = array('#root' => array( 'link' => $v, 'parent' =>
2988             Converter::getClassLink($x-> getName(),$x-> docblock-> package,$x-> getPath()), 'children'
2989             => array()));
2990         } else
2991         {
2992         }
2993     }
2994     $my_tree = array('#root' => array( 'link' => Converter::getClassLink($root-
2995 > getName(),$package,$root-> getPath()), 'parent' => false, 'children' => array()));
2996     // location of tree walker
2997     $cur = '#root';
2998     $lastcur = array(array(false,0));
2999     $childpos = 0;
3000     if (isset($class_children))
3001     {
3002         do
3003         {
3004             if (!$class_children)
3005             {
3006                 list($cur, $childpos) = array_pop($lastcur);
3007                 if (isset($my_tree[$cur]['children'][$childpos + 1]))
3008                 {
3009                     array_push($lastcur, array($cur, $childpos + 1));
3010                     $par = $cur;
3011                     $cur = $my_tree[$cur]['children'][$childpos + 1];
3012                     $x = $this-> classes-
3013 > getClassByPackage($cur['class'],$cur['package']);
3014                     $childpos = 0;
3015                     $cur = $cur['package'] . '#' . $cur['class'];
3016                     $my_tree[$cur]['link'] = Converter::getClassLink($x-> getName(),$x-
3017 > getPath());
3018                     $my_tree[$cur]['parent'] = $par;
3019                     $my_tree[$cur]['children'] = array();
3020                     $class_children = $this-> classes-> getDefiniteChildren($x-
3021 > getName(), $x-> curfile);
3022                     continue;
3023                 }
3024                 $class_children = false;
3025                 continue;
3026             }
3027             foreach($class_children as $chileclass => $chilefile)
3028             {
3029                 $ch = $this-> classes-> getClass($chileclass,$chilefile);
3030                 $my_tree[$cur]['children'][] = array('class' => $ch-> getName(),
3031 'package' => $ch-> docblock-> package);
3032                 usort($my_tree[$cur]['children'],'rootcmp');
3033                 if (isset($my_tree[$cur]['children'][$childpos]))
3034                 {
3035                     array_push($lastcur, array($cur, $childpos));
3036                     $par = $cur;
3037                     $cur = $my_tree[$cur]['children'][$childpos];
3038                     $x = $this-> classes-> getClassByPackage($cur['class'],$cur['package']);
3039                     $cur = $cur['package'] . '#' . $cur['class'];
3040                     $my_tree[$cur]['link'] = Converter::getClassLink($x-> getName(),$x-
3041 > getPath());
3042                     $my_tree[$cur]['parent'] = $par;
3043                     $my_tree[$cur]['children'] = array();
3044                     $childpos = 0;
3045                     $class_children = $this-> classes-> getDefiniteChildren($x-
3046 > getName(), $x-> curfile);
3047                 } else
3048                 {
3049                     list($cur, $childpos) = array_pop($lastcur);
3050                 }
3051             }
3052         } while ($cur);
3053     }
3054     return $my_tree;
3055 }

```

```

3049 /**
3050  * do not override
3051  * @return bool true if a link to this class exists in package $package and subpackage
3052  * $subpackage
3053  * @param string $expr class name
3054  * @param string $package package to search in
3055  * @param string $subpackage subpackage to search in
3056  * @access private
3057 */
3058 function isLinkedClass($expr,$package,$subpackage,$file=false)
3059 {
3060     if ($file)
3061         return isset($this-> linkswithfile[$package][$subpackage]['class'][$file][$expr]);
3062     return isset($this-> links[$package][$subpackage]['class'][$expr]);
3063 }
3064 /**
3065  * do not override
3066  * @return bool true if a link to this function exists in package $package and subpackage
3067  * $subpackage
3068  * @param string $expr function name
3069  * @param string $package package to search in
3070  * @param string $subpackage subpackage to search in
3071  * @access private
3072 */
3073 function isLinkedFunction($expr,$package,$subpackage,$file=false)
3074 {
3075     if ($file)
3076         return isset($this-> linkswithfile[$package][$subpackage]['function'][$file][$expr]);
3077     return isset($this-> links[$package][$subpackage]['function'][$expr]);
3078 }
3079 /**
3080  * do not override
3081  * @return bool true if a link to this define exists in package $package and subpackage
3082  * $subpackage
3083  * @param string $expr define name
3084  * @param string $package package to search in
3085  * @param string $subpackage subpackage to search in
3086  * @access private
3087 */
3088 function isLinkedDefine($expr,$package,$subpackage,$file=false)
3089 {
3090     if ($file)
3091         return isset($this-> linkswithfile[$package][$subpackage]['define'][$file][$expr]);
3092     return isset($this-> links[$package][$subpackage]['define'][$expr]);
3093 }
3094 /**
3095  * do not override
3096  * @return bool true if a link to this define exists in package $package and subpackage
3097  * $subpackage
3098  * @param string $expr define name
3099  * @param string $package package to search in
3100  * @param string $subpackage subpackage to search in
3101  * @access private
3102 */
3103 function isLinkedGlobal($expr,$package,$subpackage,$file=false)
3104 {
3105     if ($file)
3106         return isset($this-> linkswithfile[$package][$subpackage]['global'][$file][$expr]);
3107     return isset($this-> links[$package][$subpackage]['global'][$expr]);
3108 }
3109 /**
3110  * do not override
3111  * @return bool true if a link to this procedural page exists in package $package and
3112  * subpackage $subpackage
3113  * @param string $expr procedural page name
3114  * @param string $package package to search in
3115  * @param string $subpackage subpackage to search in
3116  * @access private
3117 */
3118 function isLinkedPage($expr,$package,$subpackage,$path=false)
3119 {
3120     if ($path)
3121         return isset($this-> linkswithfile[$package][$subpackage]['page'][$path][$expr]);
3122     return isset($this-> links[$package][$subpackage]['page'][$expr]);
3123 }

```

```

3124 /**
3125  * do not override
3126  * @return bool true if a link to this method exists in package $package, subpackage
3127  * $subpackage and class $class
3128  * @param string $expr method name
3129  * @param string $class class name
3130  * @param string $package package to search in
3131  * @param string $subpackage subpackage to search in
3132  * @access private
3133 */
3134 function isLinkedMethod($expr,$package,$subpackage,$class,$file=false)
3135 {
3136     if ($file)
3137         return isset($this-
3138 > linkswithfile[$package][$subpackage]['method'][$file][$class][$expr]);
3139     return isset($this->    links[$package][$subpackage]['method'][$class][$expr]);
3140 }
3141 /**
3142  * do not override
3143  * @return bool true if a link to this method exists in package $package, subpackage
3144  * $subpackage and class $class
3145  * @param string $expr var name
3146  * @param string $class class name
3147  * @param string $package package to search in
3148  * @param string $subpackage subpackage to search in
3149  * @access private
3150 */
3151 function isLinkedVar($expr,$package,$subpackage,$class,$file=false)
3152 {
3153     if ($file)
3154         return isset($this-
3155 > linkswithfile[$package][$subpackage]['var'][$file][$class][$expr]);
3156     return isset($this->    links[$package][$subpackage]['var'][$class][$expr]);
3157 }
3158 /**
3159  * do not override
3160  * @return bool true if a link to this method exists in package $package, subpackage
3161  * $subpackage and class $class
3162  * @param string $expr constant name
3163  * @param string $class class name
3164  * @param string $package package to search in
3165  * @param string $subpackage subpackage to search in
3166  * @access private
3167 */
3168 function isLinkedConst($expr,$package,$subpackage,$class,$file=false)
3169 {
3170     if ($file)
3171         return isset($this-
3172 > linkswithfile[$package][$subpackage]['const'][$file][$class][$expr]);
3173     return isset($this->    links[$package][$subpackage]['const'][$class][$expr]);
3174 }
3175 /**
3176  * return false or a {@link classLink} to $expr
3177  * @param string $expr class name
3178  * @param string $package package name
3179  * @return mixed returns a {@link classLink} or false if the element is not found in
3180  * package $package
3181  * @see classLink
3182 */
3183 function getClassLink($expr,$package,$file=false, $text = false)
3184 {
3185     if (!isset($this->    links[$package])) return false;
3186     foreach($this->    links[$package] as $subpackage =>    $notused)
3187     {
3188         if ($this->    isLinkedClass($expr,$package,$subpackage,$file))
3189         {
3190             if ($file)
3191                 {
3192                     return $this-
3193 > linkswithfile[$package][$subpackage]['class'][$file][$expr];
3194                 }
3195             return $this->    links[$package][$subpackage]['class'][$expr];
3196         }
3197     }
3198     return false;
3199 }

```

```

3196 /**
3197 * return false or a {@link functionLink} to $expr
3198 * @param string $expr function name
3199 * @param string $package package name
3200 * @return mixed returns a {@link functionLink} or false if the element is not found in
3201 package $package
3202 * @see functionLink
3203 */
3204 function getFunctionLink($expr,$package,$file=false, $text = false)
3205 {
3206     if (!isset($this-> links[$package])) return false;
3207     foreach($this-> links[$package] as $subpackage => $notused)
3208     {
3209         if ($this-> isLinkedFunction($expr,$package,$subpackage,$file))
3210         {
3211             if ($file)
3212             {
3213                 return $this-
3214             }
3215         }
3216     }
3217     return false;
3218 }
3219 /**
3220 * return false or a {@link defineLink} to $expr
3221 * @param string $expr constant name
3222 * @param string $package package name
3223 * @return mixed returns a {@link defineLink} or false if the element is not found in
3224 package $package
3225 * @see defineLink
3226 */
3227 function getDefineLink($expr,$package,$file=false, $text = false)
3228 {
3229     if (!isset($this-> links[$package])) return false;
3230     foreach($this-> links[$package] as $subpackage => $notused)
3231     {
3232         if ($this-> isLinkedDefine($expr,$package,$subpackage,$file))
3233         {
3234             if ($file)
3235             {
3236                 return $this-
3237             }
3238         }
3239     }
3240     return false;
3241 }
3242 /**
3243 * return false or a {@link globalLink} to $expr
3244 * @param string $expr global variable name (with leading $)
3245 * @param string $package package name
3246 * @return mixed returns a {@link defineLink} or false if the element is not found in
3247 package $package
3248 * @see defineLink
3249 */
3250 function getGlobalLink($expr,$package,$file=false, $text = false)
3251 {
3252     if (!isset($this-> links[$package])) return false;
3253     foreach($this-> links[$package] as $subpackage => $notused)
3254     {
3255         if ($this-> isLinkedGlobal($expr,$package,$subpackage,$file))
3256         {
3257             if ($file)
3258             {
3259                 return $this-
3260             }
3261         }
3262     }
3263     return false;
3264 }
3265 /**
3266 * return false or a {@link pageLink} to $expr

```

```

3270 * @param string $expr procedural page name
3271 * @param string $package package name
3272 * @return mixed returns a {@link pageLink} or false if the element is not found in package
3273 $package
3274 * @see pageLink
3275 */
3276 function getPageLink($expr,$package,$path = false, $text = false, $packages = false)
3277 {
3278     if (!isset($this-> links[$package])) return false;
3279     foreach($this-> links[$package] as $subpackage => $notused)
3280     {
3281         if ($this-> isLinkedPage($expr,$package,$subpackage,$path))
3282         {
3283             if ($path)
3284             {
3285                 return $this-> linkswithfile[$package][$subpackage]['page'][$path][$expr];
3286             }
3287             return $this-> links[$package][$subpackage]['page'][$expr];
3288         }
3289     }
3290     return false;
3291 }
3292 /**
3293 * return false or a {@link methodLink} to $expr in $class
3294 * @param string $expr method name
3295 * @param string $class class name
3296 * @param string $package package name
3297 * @return mixed returns a {@link methodLink} or false if the element is not found in
3298 package $package, class $class
3299 * @see methodLink
3300 */
3301 function getMethodLink($expr,$class,$package,$file=false, $text = false)
3302 {
3303     $expr = trim($expr);
3304     $class = trim($class);
3305     if (!isset($this-> links[$package])) return false;
3306     foreach($this-> links[$package] as $subpackage => $notused)
3307     {
3308         if ($this-> isLinkedMethod($expr,$package,$subpackage,$class,$file))
3309         {
3310             if ($file)
3311             {
3312                 return $this-
3313             > linkswithfile[$package][$subpackage]['method'][$file][$class][$expr];
3314         }
3315     }
3316     return false;
3317 }
3318 /**
3319 * return false or a {@link varLink} to $expr in $class
3320 * @param string $expr var name
3321 * @param string $class class name
3322 * @param string $package package name
3323 * @return mixed returns a {@link varLink} or false if the element is not found in package
3324 $package, class $class
3325 * @see varLink
3326 */
3327 function getVarLink($expr,$class,$package,$file=false, $text = false)
3328 {
3329     $expr = trim($expr);
3330     $class = trim($class);
3331     if (!isset($this-> links[$package])) return false;
3332     foreach($this-> links[$package] as $subpackage => $notused)
3333     {
3334         if ($this-> isLinkedVar($expr,$package,$subpackage,$class,$file))
3335         {
3336             if ($file)
3337             {
3338                 return $this-
3339             > linkswithfile[$package][$subpackage]['var'][$file][$class][$expr];
3340         }
3341     }
3342     return false;
3343 }
3344

```

```

3345
3346 /**
3347 * return false or a {@link constLink} to $expr in $class
3348 * @param string $expr constant name
3349 * @param string $class class name
3350 * @param string $package package name
3351 * @return mixed returns a {@link varLink} or false if the element is not found in package
3352 * $package, class $class
3353 * @see constLink
3354 */
3355 function getConstLink($expr,$class,$package,$file=false, $text = false)
3356 {
3357     $expr = trim($expr);
3358     $class = trim($class);
3359     if (!isset($this-> links[$package])) return false;
3360     foreach($this-> links[$package] as $subpackage => $notused)
3361     {
3362         if ($this-> isLinkedConst($expr,$package,$subpackage,$class,$file))
3363         {
3364             if ($file)
3365             {
3366                 return $this-
3367             > linkswithfile[$package][$subpackage]['const'][$file][$class][$expr];
3368         }
3369     }
3370     return false;
3371 }
3372 /**
3373 * The meat of the @tutorial tag and inline {@tutorial} tag
3374 *
3375 * Take a string and return an abstract link to the tutorial it represents.
3376 * Since tutorial naming literally works like the underlying filesystem, the
3377 * way to reference the tutorial is similar. Tutorials are located in a
3378 * subdirectory of any directory parsed, which is named 'tutorials/' (we
3379 * try to make things simple when we can :). They are further organized by
3380 * package and subpackage as:
3381 *
3382 * tutorials/package/subpackage
3383 *
3384 * and the files are named *.cls, *.pkg, or *.proc, and so a link to a tutorial
3385 * named file.cls can be referenced (depending on context) as any of:
3386 *
3387 * <code>
3388 * * @tutorial package/subpackage/file.cls
3389 * * @tutorial package/file.cls
3390 * * @tutorial file.cls
3391 * </code>
3392 *
3393 * The first case will only be needed if file.cls exists in both the current
3394 * package, in anotherpackage/file.cls and in anotherpackage/subpackage/file.cls
3395 * and you wish to reference the one in anotherpackage/subpackage.
3396 * The second case is only needed if you wish to reference file.cls in another
3397 * package and it is unique in that package. the third will link to the first
3398 * file.cls it finds using this search method:
3399 *
3400 * <ol>
3401 *   <li>current package/subpackage</li>
3402 *   <li>all other subpackages of current package</li>
3403 *   <li>parent package, if this package has classes that extend classes in
3404 *   another package</li>
3405 *   <li>all other packages</li>
3406 * </ol>
3407 * @return tutorialLink|string returns either a link, or the original text, if not found
3408 * @param string the original expression
3409 * @param string package to look in first
3410 * @param string subpackage to look in first
3411 * @param array array of package names to search in if not found in parent packages.
3412 *               This is used to limit the search, phpDocumentor automatically searches
3413 *               all packages
3414 * @since 1.2
3415 */
3416 function getTutorialLink($expr, $package = false, $subpackage = false, $packages = false)
3417 {
3418     // is $expr a comma-delimited list?
3419     if (strpos($expr,','))
3420     {
3421         $a = explode(',',$expr);

```

```

3423     $b = array();
3424     for($i=0;$i< count($a);$i++)
3425     {
3426         // if so return each component with a link
3427         $b[] = Converter::getTutorialLink(trim($a[$i]));
3428     }
3429     return $b;
3430 }
3431 $subsection = '';
3432 if (strpos($expr, '#'))
3433 {
3434     $a = explode('#', $expr);
3435     $org = $expr;
3436     $expr = $a[0];
3437     $subsection = $a[1];
3438 }
3439 if (strpos($expr, '/'))
3440 {
3441     $a = explode('/', $expr);
3442     if (count($a) == 3)
3443     {
3444         return Converter::getTutorialLink($a[2], $a[0], $a[1], array());
3445     }
3446     if (count($a) == 2)
3447     {
3448         return Converter::getTutorialLink($a[1], $a[0], false, array());
3449     }
3450 }
3451 if (!$package) $package = $this-> package;
3452 if (!$subpackage) $subpackage = $this-> subpackage;
3453 if (!isset($this-> all_packages[$package])) return $expr;
3454 elseif (isset($packages[$package])) unset($packages[$package]);
3455 $ext = pathinfo($expr, PATHINFO_EXTENSION);
3456 if (isset($this-> tutorials[$package][$subpackage][$ext][$expr]))
3457 {
3458     $a = $this-> tutorials[$package][$subpackage][$ext][$expr];
3459     $link = new tutorialLink;
3460     $link-> addLink($subsection, $a-> path, $a-> name, $a-> package, $a-
> subpackage, $a-> getTitle($this, $subsection));
3461     return $link;
3462 }
3463 do
3464 {
3465     if (!is_array($packages))
3466     {
3467         $packages = $this-> all_packages;
3468         if (isset($packages[$package])) unset($packages[$package]);
3469     }
3470     if (isset($this-> tutorials[$package]))
3471     {
3472         if (isset($this-> tutorials[$package][$subpackage][$ext][$expr]))
3473         {
3474             $a = $this-> tutorials[$package][$subpackage][$ext][$expr];
3475             $link = new tutorialLink;
3476             $link-> addLink($subsection, $a-> path, $a-> name, $a-> package, $a-
> subpackage, $a-> getTitle($this));
3477             return $link;
3478         }
3479     }
3480     foreach($this-> tutorials[$package] as $subpackage => $stuff)
3481     {
3482         if (isset($stuff[$ext][$expr]))
3483         {
3484             $a = $stuff[$ext][$expr];
3485             $link = new tutorialLink;
3486             $link-> addLink($subsection, $a-> path, $a-> name, $a-> package, $a-
> package, $a-> subpackage, $a-> getTitle($this));
3487             return $link;
3488         }
3489     }
3490 }
3491 }
3492 // try other packages
3493 // look in parent package first, if found
3494 if (isset($this-> package_parents[$package]))
3495 {
3496     $p1 = $package;
3497     $package = $this-> package_parents[$package];
3498 } else
3499 {

```

```

3500           // no parent package, so start with the first one that's left
3501           list($package,) = @each($packages);
3502       }
3503       if ($package)
3504     {
3505         if (isset($packages[$package])) unset($packages[$package]);
3506     }
3507   } while (count($packages) || $package);
3508   addWarning(PDERROR_TUTORIAL_NOT_FOUND,$expr);
3509   return $expr;
3510 }
3511
3512 /**
3513 * The meat of the @see tag and inline {@}link tag
3514 *
3515 * $expr is a string with many allowable formats:
3516 * <ol>
3517 *   <li>proceduralpagename.ext</li>
3518 *   <li>constant_name</li>
3519 *   <li>classname::function()</li>
3520 *   <li>classname::constantname</li> (new 1.2.4)
3521 *   <li>classname::$variablename</li>
3522 *   <li>classname</li>
3523 *   <li>object classname</li>
3524 *   <li>function functionname()</li>
3525 *   <li>global $globalvarname</li>
3526 *   <li>packagename#expr where expr is any of the above</li>
3527 * </ol>
3528 *
3529 * New in version 1.1, you can explicitly specify a package to link to that
3530 * is different from the current package. Use the # operator
3531 * to specify a new package, as in tests#bug-540368.php (which should appear
3532 * as a link like: "@link tests#bug-540368.php"). This
3533 * example links to the procedural page bug-540368.php in package
3534 * tests. Also, the "function" operator is now used to specifically
3535 * link to a function instead of a method in the current class.
3536 *
3537 * <code>
3538 * class myclass
3539 * {
3540 *   // from inside the class definition, use "function conflict()" to refer to
3541 *   function conflict()
3542 *   {
3543 *   }
3544 * }
3545 *
3546 * function conflict()
3547 * {
3548 * }
3549 * </code>
3550 *
3551 * If classname:: is not present, and the see tag is in a documentation
3552 * block within a class, then the function uses the classname to
3553 * search for $expr as a function or variable within classname, or any of its parent
3554 * classes.
3555 * given an $expr without '$', '::' or '()' getLink first searches for
3556 * classes, procedural pages, constants, global variables, and then searches for
3557 * methods and variables within the default class, and finally for any function
3558 * @param string $expr expression to search for a link
3559 * @param string $package package to start searching in
3560 * @param array $packages list of all packages to search in
3561 * @return mixed getLink returns a descendant of {@link abstractLink} if it finds a link,
3562 * otherwise it returns a string
3563 * @see getPageLink(), getDefineLink(), getVarLink(), getFunctionLink(), getClassLink()
3564 * @see pageLink, functionLink, defineLink, classLink, methodLink, varLink
3565 */
3566 function & getLink($expr, $package = false, $packages = false)
3567 {
3568     // is $expr a comma-delimited list?
3569     if (strpos($expr, ','))
3570     {
3571         $a = explode(',', $expr);
3572         $b = array();
3573         for($i=0;$i< count($a);$i++)
3574         {
3575             // if so return each component with a link
3576             $b[] = Converter::getLink(trim($a[$i]));
3577         }
3578     }
3579 }

```

```

3577         return $b;
3578     }
3579     if (strpos($expr, '#'))
3580     {
3581         $a = explode('#', $expr);
3582         if (count($a) == 2)
3583             { // can have exactly 1 package override, otherwise it's ignored
3584                 // feature 564991, link to php manual
3585                 if ($a[0] == 'PHP MANUAL')
3586                     $s = 'http://www.php.net/'. $a[1];
3587                     return $s;
3588                 }
3589                 $s = & Converter::getLink($a[1], $a[0], array());
3590                 return $s;
3591             }
3592         }
3593         $a = & $this-> _getLink($expr, $package, $packages);
3594         return $a;
3595     }
3596
3597 /**
3598 * @access private
3599 */
3600 function & _getLink($expr, $package = false, $packages = false)
3601 {
3602     if (!$package) $package = $this-> package;
3603     //
3604     if (!isset($this-> all_packages[$package])) return $expr;
3605     elseif (isset($packages[$package])) unset($packages[$package]);
3606     $links = & $this-> links;
3607     $class = $this-> class;
3608     if (strpos($expr, 'function ') === 0)
3609         { // asking for a function, not a method
3610             if ($test = Converter::getFunctionLink(str_replace('function
', ', str_replace('()', '', $expr), $package))) return $test;
3611             else return $expr;
3612         }
3613         if (strpos($expr, 'global ') === 0)
3614             { // asking for a global variable
3615                 if ($test = Converter::getGlobalLink(str_replace('global ', '', $expr), $package))
3616                     return $test;
3617                 else return $expr;
3618             }
3619             if (strpos($expr, 'object ') === 0)
3620                 { // asking for a class
3621                     if ($test = Converter::getClassLink(str_replace('object ', '', $expr), $package))
3622                         return $test;
3623                     else return $expr;
3624                 }
3625                 if (strpos($expr, 'constant ') === 0)
3626                     { // asking for a class
3627                         if ($test = Converter::getDefineLink(str_replace('constant ', '', $expr), $package))
3628                             return $test;
3629                         else return $expr;
3630                     }
3631                     // are we in a class?
3632                     if ($class)
3633                     {
3634                         // is $expr simply a word? see if it is the class
3635                         if (trim($expr) == $class)
3636                             {
3637                                 if ($test = Converter::getClassLink(trim(str_replace('object
', ', $expr)), $package)) return $test;
3638                             }
3639                             // if not, check to see if it is a method or variable of this class tree
3640                             if (!strpos($expr, '::'))
3641                             {
3642                                 if ($a = $this-> getLinkMethod($expr, $class, $package)) return $a;
3643                                 if ($a = $this-> getLinkConst($expr, $class, $package)) return $a;
3644                                 if ($a = $this-> getLinkVar('$'. $expr, $class, $package)) return $a;
3645                             }
3646                             if (strpos($expr, '$') !== 0 && ! strpos($expr, '()')) // $get = $get.'();
3647                             {
3648                                 if ($a = $this-> getLinkMethod($expr, $class, $package)) return $a;
3649                                 if ($a = $this-> getLinkConst($expr, $class, $package)) return $a;
3650                                 if ($a = $this-> getLinkVar($expr, $class, $package)) return $a;
3651                             }
3652                         }
3653                     }
3654                 }
3655             }
3656         }
3657     }
3658     if (is_numeric(strpos($expr, '$'))) if ($a = $this-
> getLinkMethod($expr, $class, $package)) return $a;
3659     if (is_numeric(strpos($expr, '$'))) if ($a = $this-
> getLinkVar($expr, $class, $package)) return $a;
3660     }

```

```

3649     }
3650     if ($test = Converter::getClassLink(trim(str_replace('object ','',$expr)), $package))
3651         return $test;
3652     if ($test = Converter::getPageLink(trim($expr), $package)) return $test;
3653     if ($test = Converter::getDefineLink(trim($expr), $package)) return $test;
3654     if ($test = Converter::getGlobalLink(trim($expr), $package)) return $test;
3655     // package specified
3656
3657     if (!is_array($packages))
3658     {
3659         $packages = $this-> all_packages;
3660     }
3661     do
3662     {
3663         if (isset($packages[$package])) unset($packages[$package]);
3664         if ($test = Converter::getClassLink(str_replace('object ','',$expr), $package))
3665             return $test;
3666         if ($test = Converter::getPageLink($expr, $package)) return $test;
3667         if ($test = Converter::getDefineLink($expr, $package)) return $test;
3668         if ($test = Converter::getGlobalLink($expr, $package)) return $test;
3669         // is $expr in class::method() or class::$variable format?
3670         if (strpos($expr, 'function ') === 0)
3671         { // asking for a function, not a method
3672             if ($test =
3673                 Converter::getFunctionLink(str_replace('function','',$str_replace('(',')',$expr)), $package)) return
3674                     $test;
3675             else return $expr;
3676         }
3677         $test = $this-> _getDoubleColon($expr, $package, $packages, $class, $links);
3678         if (!is_string($test)) return $test;
3679         if (strpos($test, 'parent::') === 0) return $test;
3680         // $expr does not have ::;
3681         if (is_numeric(@strpos('$', $expr)))
3682         {
3683             // default to current class, whose name is contained in $this->render-
3684             >parent
3685             if ($test = Converter::getVarLink($expr, $class, $package)) return $test;
3686         }
3687         // $expr is a function? (non-method)
3688         if (@strpos($expr, '()' ))
3689         {
3690             // otherwise, see if it is a method
3691             if ($class)
3692             {
3693                 if ($test = Converter::getMethodLink(str_replace('()', '',$expr), $class,
3694 $package)) return $test;
3695             }
3696             // extract the function name, use it to retrieve the file that the function is
3697             // in
3698             $page = $this->func_page[str_replace('function
3699 ', '',$str_replace('()', '',$expr))];
3700             // return the link
3701             if ($test = Converter::getFunctionLink(str_replace('function
3702 ', '',$str_replace('()', '',$expr)), $package)) return $test;
3703             // $expr is just a word. First, test to see if it is a function of the current
3704             package
3705             if ($test = Converter::getFunctionLink(str_replace('function
3706 ', '',$str_replace('()', '',$expr)), $package)) return $test;
3707             // try other packages
3708             // look in parent package first, if found
3709             if (isset($this-> package_parents[$package]) && in_array($this-
3710             > package_parents[$package], $packages))
3711             {
3712                 $p1 = $package;
3713                 $package = $this-> package_parents[$package];
3714                 if ($package)
3715                 {
3716                     if (isset($packages[$package])) unset($packages[$package]);
3717                 }
3718                 continue;
3719             }
3720             // no parent package, so start with the first one that's left
3721             $package = @array_shift(@array_keys($packages));
3722             if ($package && isset($packages[$package]))
3723             {
3724                 unset($packages[$package]);
3725             }
3726         } while (count($packages) || $package);

```

```

3717     $funcs = get_defined_functions();
3718     // feature 564991, link to php manual
3719     if (in_array(str_replace(array('(', ')'),array(' ',' '),$expr),$funcs['internal']))
3720     {
3721         $return = 'http://www.php.net/'.str_replace(array('(',')'),array(' ',' '),$expr);
3722         return $return;
3723     }
3724     // no links found
3725     return $expr;
3726 }
3727
3728 /**
3729 * Split up getLink to make it easier to debug
3730 * @access private
3731 */
3732 function _getDoubleColon(&      $expr, &      $package, &      $packages, $class, $links)
3733 {
3734     if (@strpos($expr, '::'))
3735     {
3736         $class_method = explode('::',$expr);
3737         if ($class_method[0] == 'parent')
3738         {
3739             // can only have parent in the same package as the class! subtle bug
3740             $package = $this-> package;
3741             $packages = array();
3742             $cl = $this-> classes-> getClassByPackage($class,$package);
3743             if (!$cl)
3744             { // this is possible if an example file has parent::method()
3745                 return $expr;
3746             }
3747             $par = $cl-> getParent($this);
3748             $phpparent = false;
3749             if (is_object($par))
3750             {
3751                 $package = $par-> docblock-> package;
3752                 $phpparent = $par-> getName();
3753             } else
3754             {
3755                 addWarning(PDERROR_CLASS_PARENT_NOT_FOUND,$class,$package,$class_method[1]);
3756                 return $expr;
3757             }
3758             if ($phpparent) $class_method[0] = $phpparent;
3759         }
3760         if (strpos($class_method[1], '('))
3761         {
3762             // strip everything but the function name, return a link
3763             if ($test = Converter::getMethodLink(str_replace('()','',$class_method[1]),
3764 $class_method[0], $package)) return $test;
3765             if ($test = Converter::getVarLink($class_method[1], $class_method[0], $package))
3766             if ($test = Converter::getConstLink($class_method[1], $class_method[0], $package))
3767             return $test;
3768         }
3769     }
3770
3771 /**
3772 * cycle through parent classes to retrieve a link to a method
3773 * do not use or override, used by getLink
3774 * @access private
3775 */
3776 function &      getLinkMethod($expr, $class, $package)
3777 {
3778     $links = &      $this-> links;
3779     do
3780     {
3781         // is $expr in class::method() or class::$variable format?
3782         if (@strpos($expr, '::'))
3783         {
3784             $class_method = explode('::',$expr);
3785             if ($class_method[0] == 'parent')
3786             {
3787                 $cl = $this-> classes-> getClassByPackage($class,$package);
3788                 $par = $cl-> getParent($this);
3789                 $phpparent = false;
3790                 if (is_object($par))
3791                 {
3792                     $package = $par-> docblock-> package;
3793                     $phpparent = $par-> getName();

```

```

3794     } else
3795 addWarning(PDERROR_CLASSPARENT_NOTFOUND,$class,$package,$class_method[1]);
3796     if ($phpparent) $class_method[0] = $phpparent;
3797   }
3798   {
3799     $cl = $this-> classes-> getClassByPackage($class,$package);
3800   }
3801   if (strpos($class_method[1],')')))
3802   {
3803     // strip everything but the function name, return a link
3804     if ($test = Converter::getMethodLink(str_replace('function
',',',$class_method[1])), $class_method[0], $package)) return $test;
3805   }
3806   if ($test = Converter::getMethodLink(str_replace('()',',',$expr), $class, $package))
3807 return $test;
3808   $cl = $this-> classes-> getClassByPackage($class,$package);
3809   if ($cl)
3810   {
3811     $par = $cl-> getParent($this);
3812     if (is_object($par))
3813     {
3814       $package = $par-> docblock-> package;
3815       $class = $par-> getName();
3816     } else $class = $par;
3817   } else $class = false;
3818   } while ($class);
3819   // no links found
3820   $flag = false;
3821   return $flag;
3822 }
3823 /**
3824 * cycle through parent classes to retrieve a link to a var
3825 * do not use or override, used by getLink
3826 * @access private
3827 */
3828 function & getLinkVar($expr, $class, $package)
3829 {
3830   $links = & $this-> links;
3831   do
3832   {
3833     // is $expr in class::method() or class::$variable format?
3834     if (@strpos($expr,'::'))
3835     {
3836       $class_method = explode('::',$expr);
3837       if ($class_method[0] == 'parent')
3838       {
3839         $cl = $this-> classes-> getClassByPackage($class,$package);
3840         $phpparent = false;
3841         $par = $cl-> getParent($this);
3842         if (is_object($par))
3843         {
3844           $package = $par-> docblock-> package;
3845           $phpparent = $par-> getName();
3846         } else
3847 addWarning(PDERROR_CLASSPARENT_NOTFOUND,$class,$package,$class_method[1]);
3848         if ($phpparent) $class_method[0] = $phpparent;
3849       }
3850       $cl = $this-> classes-> getClassByPackage($class,$package);
3851     }
3852     if ($test = Converter::getVarLink($class_method[1], $class_method[0],
$package)) return $test;
3853     if ($test = Converter::getVarLink('$' . $class_method[1], $class_method[0],
$package)) return $test;
3854   }
3855   if ($test = Converter::getVarLink($expr, $class, $package)) return $test;
3856   if ($test = Converter::getVarLink('$' . $expr, $class, $package)) return $test;
3857   $cl = $this-> classes-> getClassByPackage($class,$package);
3858   if ($cl)
3859   {
3860     $par = $cl-> getParent($this);
3861     if (is_object($par))
3862     {
3863       $package = $par-> docblock-> package;
3864       $class = $par-> getName();
3865     } else $class = $par;
3866   } else $class = false;
3867 } while ($class);

```

```

3868     // no links found
3869     $class = false;
3870     return $class;
3871 }
3872 /**
3873 * cycle through parent classes to retrieve a link to a class constant
3874 * do not use or override, used by getLink
3875 * @access private
3876 * @since 1.2.4
3877 */
3878 function & getLinkConst($expr, $class, $package)
3879 {
3880     $links = & $this-> links;
3881     do
3882     {
3883         // is $expr in class::method() or class::$variable format?
3884         if (@strpos($expr, '::'))
3885         {
3886             $class_method = explode('::', $expr);
3887             if ($class_method[0] == 'parent')
3888             {
3889                 $cl = $this-> classes-> getClassByPackage($class, $package);
3890                 $phpparent = false;
3891                 $par = $cl-> getParent($this);
3892                 if (is_object($par))
3893                 {
3894                     $package = $par-> docblock-> package;
3895                     $phpparent = $par-> getName();
3896                 } else
3897                 addWarning(PDERROR_CLASSPARENT_NOTFOUND, $class, $package, $class_method[1]);
3898                 if ($phpparent) $class_method[0] = $phpparent;
3899             }
3900             {
3901                 $cl = $this-> classes-> getClassByPackage($class, $package);
3902             }
3903             if ($test = Converter::getConstLink($class_method[1], $class_method[0],
3904 $package)) return $test;
3905             if ($test = Converter::getConstLink($expr, $class, $package)) return $test;
3906             $cl = $this-> classes-> getClassByPackage($class, $package);
3907             if ($cl)
3908             {
3909                 $par = $cl-> getParent($this);
3910                 if (is_object($par))
3911                 {
3912                     $package = $par-> docblock-> package;
3913                     $class = $par-> getName();
3914                 } else $class = $par;
3915             } else $class = false;
3916         } while ($class);
3917         // no links found
3918         $flag = false;
3919         return $flag;
3920     }
3921 /**
3922 * take URL $link and text $text and return a link in the format needed for the Converter
3923 * @param string URL
3924 * @param string text to display
3925 * @return string link to $link
3926 * @abstract
3927 */
3928 function returnLink($link, $text)
3929 {
3930 }
3931 /**
3932 * take {@link abstractLink} descendant and text $eltext and return a link
3933 * in the format needed for the Converter
3934 * @param abstractLink
3935 * @param string
3936 * @return string link to $element
3937 * @abstract
3938 */
3939 function returnSee(& $link, $eltext = false)
3940 {
3941 }
3942 /**
3943 */

```

```

3946 * take {@link abstractLink} descendant and text $eltext and return a
3947 * unique ID in the format needed for the Converter
3948 * @param abstractLink
3949 * @return string unique identifier of $element
3950 * @abstract
3951 */
3952 function getId(& $link)
3953 {
3954 }
3955 /**
3956 * Convert README/INSTALL/CHANGELOG file contents to output format
3957 * @param README/INSTALL/CHANGELOG
3958 * @param string contents of the file
3959 * @abstract
3960 */
3961 function Convert_RIC($name, $contents)
3962 {
3963 }
3964 /**
3965 * Convert all elements to output format
3966 *
3967 * This will call ConvertXxx where Xxx is {@link ucfirst}{$element->type}.
3968 * It is expected that a child converter defines a handler for every
3969 * element type, even if that handler does nothing. phpDocumentor will
3970 * terminate with an error if a handler doesn't exist.
3971 * {@internal}
3972 * Since 1.2.0 beta 3, this function has been moved from child converters
3973 * to the parent, because it doesn't really make sense to put it in the
3974 * child converter, and we can add error handling.
3975 *
3976 * {@source}
3977 * @throws {@link PDERROR_NO_CONVERT_HANDLER}
3978 * @param mixed {@link parserElement} descendant or {@link parserPackagePage} or {@link
3979 parserData}
3980 */
3981 function Convert(& $element)
3982 {
3983     $handler = 'convert'.ucfirst($element-> type);
3984     if (method_exists($this,$handler))
3985     {
3986         $this-> $handler($element);
3987     } else
3988     {
3989         addErrorDie(PDERROR_NO_CONVERTER_HANDLER,$element-
3990 > type,$handler,phpDocumentor_get_class($this));
3991     }
3992 }
3993 /**
3994 * Conversion Handlers
3995 *
3996 * All of the convert* handlers set up template variables for the Smarty
3997 * template.{@internal In addition, the {@link newSmarty()} method is
3998 * called to retrieve the global Smarty template}
3999 */
4000 /**
4001 * Default Tutorial Handler
4002 *
4003 * Sets up the tutorial template, and its prev/next/parent links
4004 * {@internal}
4005 * Retrieves the title using {@link parserTutorial::getTitle()} and uses the
4006 * {@link parserTutorial::prev, parserTutorial::next, parserTutorial::parent}
4007 * links to set up those links.}}
4008 * @param parserTutorial
4009 */
4010 function & convertTutorial(& $element)
4011 {
4012     $this-> package = $element-> package;
4013     $this-> subpackage = $element-> subpackage;
4014     $x = $element-> Convert($this);
4015     $template = & $this-> newSmarty();
4016     $template-> assign('contents',$x);
4017     $template-> assign('title',$element-> getTitle($this));
4018     $template-> assign('nav',$element-> parent || $element-> prev || $element-
4019 > next);
4020     if ($element-> parent)
4021     {
4022         $template-> assign('up',$this-> getId($element-> parent));
4023         $template-> assign('uptitle',$element-> parent-> title);

```

```

4023     }
4024     if ($element-> prev)
4025     {
4026         $template-> assign('prev',$this-> getId($element-> prev));
4027         $template-> assign('prevtitle',$element-> prev-> title);
4028     }
4029     if ($element-> next)
4030     {
4031         $template-> assign('next',$this-> getId($element-> next));
4032         $template-> assign('nexttitle',$element-> next-> title);
4033     }
4034     return $template;
4035 }
4036 /**
4037 * Default Class Handler
4038 *
4039 * Sets up the class template.
4040 * {@internal special methods
4041 * {@link generateChildClassList(), generateFormattedClassTree()},
4042 * {@link getFormattedConflicts, getFormattedInheritedMethods},
4043 * and {@link getFormattedInheritedVars} are called to complete vital
4044 * template setup.}}
4045 */
4046 function convertClass(& $element)
4047 {
4048     $this-> class = $element-> getName();
4049     $this-> class_data = & $this-> newSmarty();
4050     $this-> class_data-> assign("class_name" , $element-> getName());
4051     $this-> class_data-> assign("vars" , array());
4052     $this-> class_data-> assign("methods" , array());
4053     $this-> class_data-> assign("consts" , array());
4054     $this-> class_data-> assign("is_interface" , $element-> isInterface());
4055     $this-> class_data-> assign("implements" , $this-
4056 > getFormattedImplements($element));
4056     $this-> class_data-> assign("package" , $element-> docblock-> package);
4057     $this-> class_data-> assign("line_number" , $element-> getLineNumber());
4058     $this-> class_data-> assign("source_location" , $element-
4059 > getSourceLocation($this));
4060     $this-> class_data-> assign("page_link" , $this-
4061 > getCurrentPageLink());
4062     $docblock = $this-> prepareDocBlock($element, false);
4063     $this-> class_data-> assign("sdesc" , $docblock['sdesc']);
4064     $this-> class_data-> assign("desc" , $docblock['desc']);
4065     $this-> class_data-> assign("access" , $docblock['access']);
4066     $this-> class_data-> assign("abstract" , $docblock['abstract']);
4067     $this-> class_data-> assign("tags" , $docblock['tags']);
4068     $this-> class_data-> assign("api_tags" , $docblock['api_tags']);
4069     $this-> class_data-> assign("info_tags" , $docblock['info_tags']);
4070     $this-> class_data-> assign("utags" , $docblock['utags']);
4071     $this-> class_data-> assign("prop_tags" , $docblock['property_tags']);
4072     if ($this-> hasSourceCode($element-> getPath())) {
4073         $this-> class_data-> assign("class_slink" , $this-
4074 > getSourceAnchor($element-> getPath(),$element-> getLineNumber(),$element-
4075 > getLineNumber(),true));
4076     }
4077     else
4078         $this-> class_data-> assign("class_slink" , false);
4079     $this-> class_data-> assign("children" , $this-
4080 > generateChildClassList($element));
4081     $this-> class_data-> assign("class_tree" , $this-
4082 > generateFormattedClassTree($element));
4083     $this-> class_data-> assign("conflicts" , $this-
4084 > getFormattedConflicts($element,"classes"));
4085     $inherited_methods = $this-> getFormattedInheritedMethods($element);
4086     if (!empty($inherited_methods))
4087     {
4088         $this-> class_data-> assign("imethods" , $inherited_methods);
4089     }
4090     else
4091     {
4092         $this-> class_data-> assign("imethods" , false);
4093     }
4094     $inherited_vars = $this-> getFormattedInheritedVars($element);
4095     if (!empty($inherited_vars))
4096     {
4097         $this-> class_data-> assign("ivars" , $inherited_vars);
4098     }
4099     else
4100     {
4101         $this-> class_data-> assign("ivars" , false);
4102     }
}

```

```

4095     $inherited_consts = $this->  getFormattedInheritedConsts($element);
4096     if (!empty($inherited_consts))
4097     {
4098         $this->  class_data->  assign("iconsts"           , $inherited_consts);
4099     } else
4100     {
4101         $this->  class_data->  assign("iconsts"           , false);
4102     }
4103 }
4104
4105 /**
4106 * Converts method for template output
4107 *
4108 * This function must be called by a child converter with any extra
4109 * template variables needed in the parameter $addition
4110 * @param parserMethod
4111 */
4112
4113 function convertMethod(&    $element, $additions = array())
4114 {
4115     $fname = $element->  getName();
4116     $docblock = $this->  prepareDocBlock($element);
4117     $returntype = 'void';
4118     if ($element->  isConstructor) $returntype = $element->  class;
4119     if ($element->  docblock->  return)
4120     {
4121         $a = $element->  docblock->  return->  Convert($this);
4122         $returntype = $element->  docblock->  return->  converted_returnType;
4123     }
4124     $params = $param_i = array();
4125     if (count($element->  docblock->  params))
4126         foreach($element->  docblock->  params as $param =>      $val)
4127     {
4128         $a = $val->  Convert($this);
4129         $params[] = $param_i[$param] = array("var"          =>
4130 $param, "datatype"        =>      $val->  converted_returnType, "data"          =>      $a);
4131
4132         if ($element->  docblock->  hasaccess) {
4133             $acc = $docblock['access'];
4134         } else {
4135             $acc = 'public';
4136         }
4137
4138         if ($this->  hasSourceCode($element->  getPath()))
4139             $additions["slink"] = $this->  getSourceAnchor($element-
4140 >  getPath(), $element->  getLineNumber(), $element->  getLineNumber(), true);
4141         $this->  class_data->  append('methods', array_merge(
4142                                         array('sdesc' =>      $docblock['sdesc'],
4143                                               'desc'  =>      $docblock['desc'],
4144                                               'static' =>      $docblock['static'],
4145                                               'abstract' =>      $docblock['abstract'],
4146                                               'tags'   =>      $docblock['tags'],
4147                                               'api_tags' =>      $docblock['api_tags'],
4148                                               'see_tags' =>      $docblock['see_tags'],
4149                                               'info_tags_sorted' =>
4150 $docblock['info_tags_sorted'],
4151
4152 >  isConstructor,
4153
4154 >  getFunctionCall(),
4155
4156 >  getIntricateFunctionCall($this, $param_i),
4157
4158 >  getFormattedDescMethods($element),
4159
4160 >  getFormattedOverrides($element),
4161
4162 >  getFormattedMethodImplements($element),
4163
4164     )
4165
4166     array('info_tags' =>      $docblock['info_tags'],
4167           'utags'   =>      $docblock['utags'],
4168           'constructor' =>      $element-
4169
4170           'access' =>      $acc,
4171           'function_name' =>      $fname,
4172           'function_return' =>      $returntype,
4173           'function_call' =>      $element-
4174
4175           'ifunction_call' =>      $element-
4176
4177           'descmethod' =>      $this-
4178
4179           'method_overrides' =>      $this-
4180
4181           'method_implements' =>      $this-
4182
4183           'line_number' =>      $element-
4184
4185           'id' =>      $this->  getId($element),
4186           'params' =>      $params),
4187           $additions));
4188 }
```

```

4165
4166 /**
4167 * Converts class variables for template output.
4168 *
4169 * This function must be called by a child converter with any extra
4170 * template variables needed in the parameter $addition
4171 * @param parserVar
4172 */
4173 function convertVar(& $element, $additions = array())
4174 {
4175     $docblock = $this-> prepareDocBlock($element);
4176     $b = 'mixed';
4177
4178     if ($element-> docblock-> hasaccess)
4179         $acc = $element-> docblock-> tags['access'][0]-> value;
4180     else
4181         $acc = 'public';
4182
4183     if ($element-> docblock-> var)
4184     {
4185         $b = $element-> docblock-> var-> converted_returnType;
4186     }
4187     if ($this-> hasSourceCode($element-> getPath()))
4188         $additions["slink"] = $this-> getSourceAnchor($element-
4189 > getPath(),$element-> getLineNumber(),$element-> getLineNumber(),true);
4190     $this-> class_data-> append('vars',array_merge(
4191
4192         array('sdesc' => $docblock['sdesc'],
4193               'desc' => $docblock['desc'],
4194               'static' => $docblock['static'],
4195               'abstract' => $docblock['abstract'],
4196               'utags' => $docblock['utags'],
4197               'tags' => $docblock['tags'],
4198               'api_tags' => $docblock['api_tags'],
4199               'info_tags' => $docblock['info_tags'],
4200               'var_name' => $element-> getName(),
4201               'has_default' => strlen($element-
4202 > getValue()),
4203
4204         'var_default' => $this-
4205
4206         'var_type' => $b,
4207         'access' => $acc,
4208         'line_number' => $element-
4209
4210         'descvar' => $this-
4211
4212         'var_overrides' => $this-
4213
4214         'id' => $this-> getId($element),
4215
4216     $additions));
4217 }
4218
4219 /**
4220 * Converts class constants for template output.
4221 *
4222 * This function must be called by a child converter with any extra
4223 * template variables needed in the parameter $addition
4224 * @param parserConst
4225 */
4226 function convertConst(& $element, $additions = array())
4227 {
4228     $docblock = $this-> prepareDocBlock($element);
4229
4230     if ($element-> docblock-> hasaccess)
4231         $acc = $element-> docblock-> tags['access'][0]-> value;
4232     else
4233         $acc = 'public';
4234
4235     if ($this-> hasSourceCode($element-> getPath()))
4236         $additions["slink"] = $this-> getSourceAnchor($element-
4237 > getPath(),$element-> getLineNumber(),$element-> getLineNumber(),true);
4238     $this-> class_data-> append('consts',array_merge(
4239
4240         array('sdesc' => $docblock['sdesc'],
4241               'desc' => $docblock['desc'],
4242               'access' => $docblock['access'],
4243               'abstract' => $docblock['abstract'],
4244               'utags' => $docblock['utags'],
4245               'tags' => $docblock['tags'],
4246               'api_tags' => $docblock['api_tags'],
4247               'info_tags' => $docblock['info_tags'],
4248               'const_name' => $element-> getName(),
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606
4607
4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703
4704
4705
4706
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859
4860
4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
4999

```

```

4238                                         'const_value' =>      $this-
4239 >  postProcess($element->    getValue()),           'access'   =>      $acc,
4240                                         'line_number' =>      $element-
4241                                         ,  

4242                                         'id'   =>      $this->    getId($element),
4243                                         $additions));
4244 }
4245 /**
4246 * Default Page Handler
4247 *
4248 * {@internal In addition to setting up the smarty template with {@link newSmarty()},
4249 * this class uses {@link getSourceLocation()} and {@link getClassesOnPage()}
4250 * to set template variables. Also used is {@link getPageName()}, to get
4251 * a Converter-specific name for the page.})
4252 * @param parserPage
4253 */
4254 function convertPage(&    $element)
4255 {
4256     $this->  page_data = &    $this->  newSmarty(true);
4257     $this->  page = $this->  getPageName($element->  parent);
4258     $this->  path = $element->  parent->  getPath();
4259     $this->  curpage = &    $element->  parent;
4260     $this->  page_data->  assign("source_location"           , $element->  parent-
4261 >  getSourceLocation($this));
4262     $this->  page_data->  assign("functions"            , array());
4263     $this->  page_data->  assign("includes"             , array());
4264     $this->  page_data->  assign("defines"              , array());
4265     $this->  page_data->  assign("globals"              , array());
4266     $this->  page_data->  assign("classes"              , $this-
4267 >  getClassesOnPage($element));
4268     $this->  page_data->  assign("hasclasses"           , $element->  hasClasses());
4269     $this->  page_data->  assign("hasinterfaces"        , $element->  hasInterfaces());
4270     $this->  page_data->  assign("name"                 , $element->  parent->  getFile());
4271     if ($t = $element->  getTutorial())
4272     {
4273         $this->  page_data->  assign("tutorial"            , $this->  returnSee($t));
4274     } else
4275     {
4276         $this->  page_data->  assign("tutorial"            , false);
4277     }
4278     if ($element->  docblock)
4279     {
4280         $docblock = $this->  prepareDocBlock($element, false);
4281         $this->  page_data->  assign("sdesc"                , $docblock['sdesc']);
4282         $this->  page_data->  assign("desc"                 , $docblock['desc']);
4283         $this->  page_data->  assign("tags"                 , $docblock['tags']);
4284         $this->  page_data->  assign("api_tags"            , $docblock['api_tags']);
4285         $this->  page_data->  assign("info_tags"           , $docblock['info_tags']);
4286         $this->  page_data->  assign("utags"                , $docblock['utags']);
4287     }
4288 /**
4289 * Converts global variables for template output
4290 *
4291 * This function must be called by a child converter with any extra
4292 * template variables needed in the parameter $addition
4293 * {@internal
4294 * In addition to using {@link prepareDocBlock()}, this method also
4295 * uses utility functions {@link getGlobalValue(), getFormattedConflicts()}}}
4296 * @param parserGlobal
4297 * @uses postProcess() on global_value template value, makes it displayable
4298 * @param array any additional template variables should be in this array
4299 */
4300 function convertGlobal(&    $element, $addition = array())
4301 {
4302     $docblock = $this->  prepareDocBlock($element);
4303     $value = $this->  getGlobalValue($element->  getValue());
4304     if ($this->  hasSourceCode($element->  getPath()))
4305     $addition["slink"] = $this->  getSourceAnchor($element-
4306 >  getPath(), $element->  getLineNumber(), $element->  getLineNumber(), true);
4307     $this->  page_data->  append('globals', array_merge(
4308                                         array('sdesc' =>      $docblock['sdesc'],
4309                                               'desc'  =>      $docblock['desc'],
4310                                               'tags'  =>      $docblock['tags'],
4311                                               'api_tags' =>      $docblock['api_tags'],
4312                                               'info_tags' =>      $docblock['info_tags'],
4313                                               'utags'  =>      $docblock['utags'],

```

```

4313                                     'global_name'      =>      $element-
4314 >   getName(),
4315                                     'global_type'    =>      $element-
4316 >   getDataType($this),
4317                                     'global_value'   =>      $value,
4318 >   getLineNumber(),
4319                                     'line_number'    =>      $element-
4320                                     'global_conflicts' =>      $this-
4321                                     ), 'id' =>      $this->  getId($element)),
4322 }                                     $addition));
4323
4324 /**
4325  * Converts defines for template output
4326 *
4327  * This function must be called by a child converter with any extra
4328  * template variables needed in the parameter $addition
4329  * {@internal
4330  * In addition to using {@link prepareDocBlock()}, this method also
4331  * uses utility functions {@link getGlobalValue(), getFormattedConflicts()}}}
4332  * @param parserDefine
4333  * @uses postProcess() on define_value template value, makes it displayable
4334  * @param array any additional template variables should be in this array
4335 */
4336 function convertDefine(&      $element, $addition = array())
4337 {
4338     $docblock = $this->  prepareDocBlock($element);
4339     if ($this->  hasSourceCode($element->  getPath()))
4340         $addition["slink"] = $this->  getSourceAnchor($element-
4341 >  getPath(),$element->  getLineNumber(),$element->  getLineNumber(),true);
4342     $this->  page_data->  append('defines',array_merge(
4343                                     array('sdesc' =>      $docblock['sdesc'],
4344                                         'desc' =>      $docblock['desc'],
4345                                         'tags' =>      $docblock['tags'],
4346                                         'api_tags' =>      $docblock['api_tags'],
4347                                         'info_tags' =>      $docblock['info_tags'],
4348                                         'utags' =>      $docblock['utags'],
4349                                         'define_name' =>      $element-
4350                                     'line_number' =>      $element-
4351                                     'define_value' =>      $this-
4352                                     'define_conflicts' =>      $this-
4353                                     ), 'id' =>      $this->  getId($element),
4354                                     $addition));
4355 }
4356
4357 /**
4358  * Converts includes for template output
4359  *
4360  * This function must be called by a child converter with any extra
4361  * template variables needed in the parameter $addition
4362  * @see prepareDocBlock()
4363  * @param parserInclude
4364 */
4365 function convertInclude(&      $element, $addition = array())
4366 {
4367     $docblock = $this->  prepareDocBlock($element);
4368     $per = $this->  getIncludeValue($element->  getValue(), $element->  getPath());
4369     if ($this->  hasSourceCode($element->  getPath()))
4370         $addition["slink"] = $this->  getSourceAnchor($element-
4371 >  getPath(),$element->  getLineNumber(),$element->  getLineNumber(),true);
4372     $this->  page_data->  append('includes',array_merge(
4373                                     array('sdesc' =>      $docblock['sdesc'],
4374                                         'desc' =>      $docblock['desc'],
4375                                         'tags' =>      $docblock['tags'],
4376                                         'api_tags' =>      $docblock['api_tags'],
4377                                         'info_tags' =>      $docblock['info_tags'],
4378                                         'utags' =>      $docblock['utags'],
4379                                         'include_name' =>      $element-
4380                                     'line_number' =>      $element-
4381                                     'include_value' =>      $per,
4382                                     $addition));

```

```

4381 }
4382 /**
4383 * Converts function for template output
4384 *
4385 * This function must be called by a child converter with any extra
4386 * template variables needed in the parameter $addition
4387 * @see prepareDocBlock()
4388 * @param parserFunction
4389 */
4390
4391 function convertFunction(& $element, $addition = array())
4392 {
4393     $docblock = $this-> prepareDocBlock($element);
4394     $fname = $element-> getName();
4395     $params = $param_i = array();
4396     if (count($element-> docblock-> params))
4397         foreach($element-> docblock-> params as $param => $val)
4398     {
4399         $a = $val-> Convert($this);
4400         $params[] = $param_i[$param] = array("var" =>
4401 $param, "datatype" => $val-> converted_returnType, "data" => $a);
4402     }
4403     $returntype = 'void';
4404     if ($element-> docblock-> return)
4405     {
4406         $a = $element-> docblock-> return-> Convert($this);
4407         $returntype = $element-> docblock-> return-> converted_returnType;
4408     }
4409     if ($this-> hasSourceCode($element-> getPath()))
4410         $addition["slink"] = $this-> getSourceAnchor($element-
4411 > getPath(), $element-> getLineNumber(), $element-> getLineNumber(), true);
4412     $this-> page_data-> append('functions', array_merge(
4413         array('sdesc' => $docblock['sdesc'],
4414             'desc' => $docblock['desc'],
4415             'tags' => $docblock['tags'],
4416             'api_tags' => $docblock['api_tags'],
4417             'info_tags' => $docblock['info_tags'],
4418             'utags' => $docblock['utags'],
4419             'function_name' => $fname,
4420             'function_return' => $returntype,
4421             'function_conflicts' => $this-
4422 > getFormattedConflicts($element, "functions"
4423 > getIntricateFunctionCall($this, $param_i),
4424 > getFunctionCall(),
4425 > getLineNumber(),
4426 > $addition));
4427 }
4428 /**#@-*/
4429
4430 /**
4431 * convert the element's DocBlock for output
4432 *
4433 * This function converts all tags and descriptions for output
4434 * @param mixed any descendant of {@link parserElement}, or {@link parserData}
4435 * @param array used to translate tagnames into other tags
4436 * @param boolean set to false for pages and classes, the only elements allowed to specify
4437 @package
4438 * @return array
4439 * Format:
4440 * <pre>
4441 * array('sdesc' => DocBlock summary
4442 *       'desc' => DocBlock detailed description
4443 *       'tags' => array('keyword' => tagname, 'data' => tag description)
4444 *           known tags
4445 *       'api_tags' => array('keyword' => tagname, 'data' => tag description)
4446 *           known api documentation tags
4447 *       'info_tags' => array('keyword' => tagname, 'data' => tag description)
4448 *           known informational tags
4449 *       [ 'utags' => array('keyword' => tagname, 'data' => tag description
4450 *           unknown tags ]
4451 *       [ 'vartype' => type from @var/@return tag ]
4452 *       [ 'var_descrip' => description from @var/@return tag ]
4453 *)

```

```

4454     *  </pre>
4455     */
4456     function prepareDocBlock(& $element, $names = array(), $nopackage = true)
4457     {
4458         $stagses = $element-> docblock-> listTags();
4459         $tags = $ret = $api_tags = $info_tags = array();
4460         $api_tags_arr = array("abstract" , "access" , "deprecated"
4461                               , "filesource" , "global" , "internal" , "name"
4462                               , "see" , "property" , "property-read" , "property-write"
4463                               , "method" , "staticvar" , "usedby" , "uses"
4464                               );
4465         if (!$nopackage)
4466         {
4467             $tags[] = array('keyword' => 'package', 'data' => $element-> docblock-
4468 > package);
4469             if (!empty($element-> docblock-> subpackage)) $tags[] = array('keyword' =>
4470 'subpackage', 'data' => $element-> docblock-> subpackage);
4471             if ($element-> docblock-> var)
4472             {
4473                 $a = $element-> docblock-> var-> Convert($this);
4474                 $ret['vartype'] = $element-> docblock-> var-> converted_returnType;
4475                 if (!empty($a))
4476                 {
4477                     $tags[] = array('keyword' => 'var', 'data' => $a);
4478                     $ret["var_descip"] = $a;
4479                 }
4480             if ($element-> docblock-> return)
4481             {
4482                 $a = $element-> docblock-> return-> Convert($this);
4483                 $ret['vartype'] = $element-> docblock-> return-> converted_returnType;
4484                 if (!empty($a))
4485                 {
4486                     $tags[] = $api_tags[] = array('keyword' => 'return', 'data' => $a);
4487                     $ret["var_descip"] = $a;
4488                 }
4489             if ($element-> docblock-> funcglobals)
4490             foreach($element-> docblock-> funcglobals as $global => $val)
4491             {
4492                 if ($a = $this-> getGlobalLink($global,$element-> docblock-> package))
4493                 {
4494                     $global = $a;
4495                 }
4496                 $b = Converter::getLink($val[0]);
4497                 if (is_object($b) && phpDocumentor_get_class($b) == 'classlink')
4498                 {
4499                     $val[0] = $this-> returnSee($b);
4500                 }
4501                 $tags[] = $api_tags[] = array('keyword' => 'global', 'data' => $val[0].'
4502 '. $global.' : ' . $val[1]-> Convert($this));
4503                 if ($element-> docblock-> statics)
4504                 foreach($element-> docblock-> statics as $static => $val)
4505                 {
4506                     $a = $val-> Convert($this);
4507                     $tags[] = $api_tags[] = array('keyword' => 'staticvar', 'data' => $val-
4508 > converted_returnType.' : ' . $static.' : ' . $a);
4509                     $property_tags = array();
4510                     foreach ( $element-> docblock-> properties as $prop_name => $val )
4511                     {
4512                         $a = $val-> Convert( $this );
4513                         if ( !empty( $a ) )
4514                         {
4515                             $tags[] = $api_tags[] = array( 'keyword' => $val-> keyword ,
4516                               'data' => $val-> converted_returnType
4517                               . ' . $prop_name . ' : ' . $a );
4518                             $prop['prop_name'] = $prop_name;
4519                             $prop['access'] = $val-> keyword == 'property-read' ? 'read' :
4520                               ( $val-> keyword == 'property-write' ? 'write' :
4521                                 'read/write' );
4522                             $prop['prop_type'] = $val-> converted_returnType;
4523                             $prop['sdesc'] = $a;
4524                             $property_tags[ $prop_name ] = $prop;
4525                         }
4526                     }
4527                 }
4528             }

```

```

4524     }
4525    ksort( $property_tags, SORT_STRING );
4526     $property_tags = array_values( $property_tags );
4527     $info_tags_sorted = array();
4528     $ret['static'] = false;
4529     foreach($tagses as $tag)
4530     {
4531         if (isset($names[$tag-> keyword])) $tag-> keyword = $names[$tag-> keyword];
4532         if ($tag-> keyword == 'static') {
4533             $ret['static'] = true;
4534             continue;
4535         }
4536         if ($tag-> keyword)
4537             $tags[] = array("keyword" => $tag-> keyword, "data" => Convert($this));
4538         if (in_array($tag-> keyword, $api_tags_arr)) {
4539             $api_tags[] = array("keyword" => $tag-> keyword, "data" => Convert($this));
4540         } else {
4541             $info_tags[] = array("keyword" => $tag-> keyword);
4542             $tag-> keyword => Convert($this));
4543             @list( $className, $desc ) = explode( " " , $tag-> Convert($this), 2 );
4544             $info_tags_sorted[ $tag-> keyword ][] = array( 'keyword' => $className,
4545             'data' => $desc );
4546         }
4547     }
4548     $utags = array();
4549     foreach($element-> docblock-> unknown_tags as $keyword => $tag)
4550     {
4551         foreach($tag as $t)
4552             $utags[] = array('keyword' => $keyword, 'data' => $t-> Convert($this));
4553     }
4554     $ret['abstract'] = false;
4555     $ret['access'] = 'public';
4556     $see_tags = array();
4557     foreach($tags as $tag)
4558     {
4559         if ($tag['keyword'] == 'access') {
4560             $ret['access'] = $tag['data'];
4561         }
4562         if ($tag['keyword'] == 'abstract') {
4563             $ret['abstract'] = true;
4564         }
4565         if ($tag['keyword'] == 'see' || $tag['keyword'] == 'uses' ||
4566             $tag['keyword'] == 'usedby') {
4567             $see_tags[] = $tag['data'];
4568         }
4569     }
4570     $ret['sdesc'] = $element-> docblock-> getSDesc($this);
4571     $ret['desc'] = $element-> docblock-> getDesc($this);
4572     $ret['tags'] = $tags;
4573     $ret['see_tags'] = $see_tags;
4574     $ret['info_tags_sorted'] = $info_tags_sorted;
4575     $ret['api_tags'] = $api_tags;
4576     $ret['info_tags'] = $info_tags;
4577     $ret['utags'] = $utags;
4578     $ret['property_tags'] = $property_tags;
4579     return $ret;
4580 }
4581 /**
4582 * gets a list of all classes declared on a procedural page represented by
4583 * $element, a {@link parserData} class
4584 * @param parserData &$element
4585 * @return array links to each classes documentation
4586 *
4587 * Format:
4588 * <pre>
4589 * array('name' => class name,
4590 *       'sdesc' => summary of the class
4591 *       'link' => link to the class's documentation)
4592 * </pre>
4593 */
4594 function getClassesOnPage(& $element)
4595 {
4596     global $_phpDocumentor_setting;
4597     $a = $element-> getClasses($this);
4598     $classes = array();
4599     foreach($a as $package => $clas)

```

```

4599     {
4600         if (!empty($_phpDocumentor_setting['packageoutput']))
4601     {
4602         $packages = explode(',', $_phpDocumentor_setting['packageoutput']);
4603         if (!in_array($package, $packages)) continue;
4604     }
4605     for($i=0; $i< count($clas); $i++)
4606     {
4607         if ($this-> parseprivate || ! ($clas[$i]-> docblock && $clas[$i]->
4608 > docblock-> hasaccess && $clas[$i]-> docblock-> tags['access'][0]-> value ==
4609 'private'))
4610     {
4611         $sdesc = '';
4612         $r = array();
4613         $sdesc = $clas[$i]-> docblock-> getsDesc($this);
4614         if ($clas[$i]-> docblock-> hasaccess)
4615             $r['access'] = $clas[$i]-> docblock-
4616 > tags['access'][0]-> value;
4617         else
4618             $r['access'] = 'public';
4619         if (isset ($clas[$i]-> docblock-> tags['abstract']))
4620             $r['abstract'] = TRUE;
4621         else
4622             $r['abstract'] = FALSE;
4623         $r['name'] = $clas[$i]-> getName();
4624         $r['sdesc'] = $sdesc;
4625         $r['link'] = $this-> getClassLink($clas[$i]-
4626 > getName(), $package, $clas[$i]-> getPath());
4627         $classes[] = $r;
4628     }
4629 }
4630 /**
4631 * returns an array containing the class inheritance tree from the root
4632 * object to the class.
4633 *
4634 * This method must be overridden, or phpDocumentor will halt with a fatal
4635 * error
4636 * @return string Converter-specific class tree for an individual class
4637 * @param parserClass class variable
4638 * @abstract
4639 */
4640
4641 function generateFormattedClassTree($class)
4642 {
4643     addErrorDie(PDERROR_CONVERTER_OVR_GFCT,phpDocumentor_get_class($this));
4644 }
4645
4646 /**
4647 * returns an array containing the class inheritance tree from the root
4648 * object to the class.
4649 *
4650 * This method must be overridden, or phpDocumentor will halt with a fatal
4651 * error
4652 * @return string Converter-specific class tree for an individual class
4653 * @param parserClass class variable
4654 * @abstract
4655 */
4656
4657 function getFormattedImplements($el)
4658 {
4659     $ret = array();
4660     foreach ($el-> getImplements() as $interface)
4661     {
4662         $link = $this-> getLink($interface);
4663         if ($link && is_object($link)) {
4664             $ret[] = $this-> returnSee($link);
4665         } else {
4666             if (class_exists('ReflectionClass')) {
4667                 if (interface_exists($interface)) {
4668                     $inter = new ReflectionClass($interface);
4669                     if ($inter-> isInternal()) {
4670                         $ret[] = $interface . ' (internal interface)';
4671                     } else {
4672                         $ret[] = $interface;
4673                     }
4674                 }

```

```

4675         } else {
4676             $ret[] = $interface;
4677         }
4678     }
4680     return $ret;
4681 }
4682
4683 /**
4684 * @param mixed {@link parserClass, parserFunction, parserDefine} or
4685 * {@link parserGlobal}
4686 * @param string type to display. either 'class','function','define'
4687 * or 'global variable'
4688 * @return array links to conflicting elements, or empty array
4689 * @uses parserClass::getConflicts()
4690 * @uses parserFunction::getConflicts()
4691 * @uses parserDefine::getConflicts()
4692 * @uses parserGlobal::getConflicts()
4693 */
4694 function getFormattedConflicts(& $element,$type)
4695 {
4696     $conflicts = $element-> getConflicts($this);
4697     $r = array();
4698     if (!$conflicts) return false;
4699     foreach($conflicts as $package => $class)
4700     {
4701         $r[] = $class-> getLink($this,$class-> docblock-> package);
4702     }
4703     if (!empty($r)) $r = array('conflicttype' => $type, 'conflicts' => $r);
4704     return $r;
4705 }
4706
4707 /**
4708 * Get a list of methods in child classes that override this method
4709 * @return array empty array or array(array('link'=>link to method,
4710 * 'sdesc'=>short description of the method),...)
4711 * @uses parserMethod::getOverridingMethods()
4712 * @param parserMethod
4713 */
4714 function getFormattedDescMethods(& $element)
4715 {
4716     $meths = $element-> getOverridingMethods($this);
4717     $r = array();
4718     for($i=0; $i< count($meths); $i++)
4719     {
4720         $ms = array();
4721         $ms['link'] = $meths[$i]-> getLink($this);
4722         $ms['sdesc'] = $meths[$i]-> docblock-> getSDesc($this);
4723         $r[] = $ms;
4724     }
4725     return $r;
4726 }
4727
4728 /**
4729 * Get a list of vars in child classes that override this var
4730 * @return array empty array or array('link'=>link to var,
4731 * 'sdesc'=>short description of the method
4732 * @uses parserVar::getOverridingVars()
4733 * @param parserVar
4734 */
4735 function getFormattedDescVars(& $element)
4736 {
4737     $vars = $element-> getOverridingVars($this);
4738     $r = array();
4739     for($i=0; $i< count($vars); $i++)
4740     {
4741         $vs = array();
4742         $vs['link'] = $vars[$i]-> getLink($this);
4743         $vs['sdesc'] = $vars[$i]-> docblock-> getSDesc($this);
4744         $r[] = $vs;
4745     }
4746     return $r;
4747 }
4748
4749 /**
4750 * Get the method this method overrides, if any
4751 * @return array/falsearray('link'=>link to overridden method,
4752 * 'sdesc'=>short description
4753 * @see parserMethod::getOverrides()
4754 * @param parserMethod

```

```

4755      */
4756      function getFormattedOverrides(& $element)
4757      {
4758          $ovr = $element-> getOverrides($this);
4759          if (!$ovr) return false;
4760          $sdesc = $ovr-> docblock-> getSDesc($this);
4761          $name = method_exists($ovr, 'getFunctionCall') ? $ovr-> getFunctionCall() : $ovr-
> getName();
4762          $link = ($link = $ovr-> getLink($this)) ? $link : $ovr-> getClass() . '::' . $name;
4763          return array('link' => $link, 'sdesc' => $sdesc);
4764      }
4765
4766      /**
4767      * Get the method this method(s) implemented from an interface, if any
4768      * @return array|false array('link'=>link to implemented method,
4769      * 'sdesc'=>short description
4770      * @uses parserMethod::getImplements()
4771      * @param parserMethod
4772      */
4773      function getFormattedMethodImplements(& $element)
4774      {
4775          $ovr = $element-> getImplements($this);
4776          if (!$ovr) return false;
4777          $ret = array();
4778          foreach ($ovr as $impl) {
4779              $sdesc = $impl-> docblock-> getSDesc($this);
4780              $name = $impl-> getName();
4781              $link = ($link = $impl-> getLink($this)) ? $link : $impl-> getClass() . '::' .
$name;
4782              $ret[] = array('link' => $link, 'sdesc' => $sdesc);
4783          }
4784          return $ret;
4785      }
4786
4787      /**
4788      * returns a list of child classes
4789      *
4790      * @param parserClass class variable
4791      * @uses parserClass::getChildClassList()
4792      */
4793
4794      function generateChildClassList($class)
4795      {
4796          $kids = $class-> getChildClassList($this);
4797          $list = array();
4798          if (count($kids))
4799          {
4800              for($i=0; $i< count($kids); $i++)
4801              {
4802                  $lt['link'] = $kids[$i]-> getLink($this);
4803                  $lt['sdesc'] = $kids[$i]-> docblock-> getSDesc($this);
4804
4805                  if ($kids[$i]-> docblock-> hasaccess)
4806                      $lt['access'] = $kids[$i]-> docblock-
> tags['access'][0]-> value;
4807                  else
4808                      $lt['access'] = 'public';
4809
4810                  $lt['abstract'] = isset ($kids[$i]-> docblock-
> tags['abstract'][0]);
4811
4812                  $list[] = $lt;
4813              }
4814          } else return false;
4815          return $list;
4816      }
4817
4818      /**
4819      * Return template-enabled list of inherited variables
4820      *
4821      * uses parserVar helper function getInheritedVars and generates a
4822      * template-enabled list using getClassLink()
4823      * @param parserVar $child class var
4824      * @see getClassLink(), parserVar::getInheritedVars()
4825      * @return array Format:
4826      * <pre>
4827      * array(
4828      *     array('parent_class' => link to parent class's documentation,
4829      *           'ivars' =>
4830      *             array(

```

```

4831     *           array('name' => inherited variable name,
4832     *           'link' => link to inherited variable's documentation,
4833     *           'default' => default value of inherited variable,
4834     *           'sdesc' => summary of inherited variable),
4835     *           ...),
4836     *       ...)
4837     *   </pre>
4838   */
4839
4840   function getFormattedInheritedVars($child)
4841 {
4842     $package = $child-> docblock-> package;
4843     $subpackage = $child-> docblock-> subpackage;
4844     $ivars = $child-> getInheritedVars($this);
4845     $results = array();
4846     if (!count($ivars)) return $results;
4847     foreach($ivars as $parent => $vars)
4848     {
4849       $file = $vars['file'];
4850       $vars = $vars['vars'];
4851       $par = $this-> classes-> getClass($parent,$file);
4852       if ($par) {
4853         $package = $par-> docblock-> package;
4854       }
4855       usort($vars,array($this,"sortVar"));
4856       $result['parent_class'] = $this-> getClassLink($parent, $package);
4857       if (!$result['parent_class']) {
4858         $result['parent_class'] = $parent . ' (Internal Class)';
4859       }
4860       foreach($vars as $var)
4861       {
4862         $info = array();
4863
4864         if ($var-> docblock-> hasaccess) {
4865           $info['access'] = $var-> docblock-> tags['access'][0]-> value;
4866         } else {
4867           $info['access'] = 'public';
4868         }
4869
4870         $info['abstract'] = isset ($var-> docblock-> tags['abstract'][0]);
4871
4872         $info['name'] = $var-> getName();
4873         $info['link'] = $var-> getLink($this);
4874         if (!$info['link']) {
4875           $info['link'] = $info['name'];
4876         }
4877         $info['default'] = $this-> postProcess($var-> getValue());
4878         if ($var-> docblock)
4879           $info['sdesc'] = $var-> docblock-> getSDesc($this);
4880         $result["ivars"] [] = $info;
4881       }
4882       $results[] = $result;
4883       $result = array();
4884     }
4885   }
4886
4887 /**
4888 * Return template-enabled list of inherited methods
4889 *
4890 * uses parserMethod helper function getInheritedMethods and generates a
4891 * template-enabled list using getClassLink()
4892 * @param parserMethod $child class method
4893 * @see getClassLink(), parserMethod::getInheritedMethods()
4894 * @return array Format:
4895 * <pre>
4896 * array(
4897 *   array('parent_class' => link to parent class's documentation,
4898 *         'ivars' =>
4899 *           array(
4900 *             array('name' => inherited variable name,
4901 *                   'link' => link to inherited variable's documentation,
4902 *                   'function_call' => {@link
4903 *                               returned array,
4904 *                               'sdesc' => summary of inherited variable),
4905 *                   ...),
4906 *             ...
4907 *           </pre>
4908 *         )
4909 */

```

```

4910
4911     function getFormattedInheritedMethods($child)
4912     {
4913         $package = $child-> docblock-> package;
4914         $subpackage = $child-> docblock-> subpackage;
4915         $imethods = $child-> getInheritedMethods($this);
4916         $results = array();
4917         if (!count($imethods)) return $results;
4918         foreach($imethods as $parent => $methods)
4919         {
4920             $file = $methods['file'];
4921             $methods = $methods['methods'];
4922             $par = $this-> classes-> getClass($parent,$file);
4923             if ($par) {
4924                 $package = $par-> docblock-> package;
4925             }
4926             usort($methods, array($this, "sortMethod"));
4927             $result['parent_class'] = $this-> getClassLink($parent,$package);
4928             if (!$result['parent_class']) {
4929                 $result['parent_class'] = $parent . ' (Internal Class)';
4930             }
4931             foreach($methods as $method)
4932             {
4933                 $info = array();
4934
4935                 if ($method-> docblock-> hasaccess) {
4936                     $info['access'] = $method-> docblock-> tags['access'][0]-> value;
4937                 } else {
4938                     $info['access'] = 'public';
4939                 }
4940                 $info['abstract'] = isset ($method-> docblock-> tags['abstract'][0]);
4941
4942                 if ($method-> isConstructor) $info['constructor'] = 1;
4943                 $returntype = 'void';
4944                 if ($method-> isConstructor) {
4945                     $returntype = $method-> getClass();
4946                 }
4947                 if ($method-> docblock-> return) {
4948                     $a = $method-> docblock-> return-> Convert($this);
4949                     $returntype = $method-> docblock-> return-> converted_returnType;
4950                 }
4951                 $info['function_return'] = $returntype;
4952                 $info['static'] = isset ($method-> docblock-> tags['static'][0]);
4953                 $info['link'] = $method-> getLink($this);
4954                 if (!$info['link']) {
4955                     $info['link'] = $method-> getFunctionCall();
4956                 }
4957                 $info['name'] = $method-> getName();
4958                 if ($method-> docblock)
4959                     $info['sdesc'] = $method-> docblock-> getSDesc($this);
4960                 $params = array();
4961                 if (count($method-> docblock-> params))
4962                     foreach($method-> docblock-> params as $param => $val)
4963                     {
4964                         $a = $val-> Convert($this);
4965                         $params[$param] = array("var" => $param, "datatype" => $val-> converted_returnType, "data" => $a);
4966                     }
4967
4968                     $info['function_call'] = $method-> getIntricateFunctionCall($this,$params);
4969                     $result["imethods"] [] = $info;
4970                 }
4971                 $results[] = $result;
4972                 $result = array();
4973             }
4974             return $results;
4975         }
4976     }
4977 /**
4978 * Return template-enabled list of inherited class constants
4979 *
4980 * uses parserConst helper function getInheritedConsts and generates a
4981 * template-enabled list using getClassLink()
4982 * @param parserConst $child class constant
4983 * @see getClassLink(), parserMethod::getInheritedConsts()
4984 * @return array Format:
4985 * <pre>
4986 * array(
4987 *     array('parent_class' => link to parent class's documentation,
4988 *           'ivars' =>

```

```

4989     *
4990     *      array(
4991     *          'name' => inherited constant name,
4992     *          'link' => link to inherited constant's documentation,
4993     *          'value' => constant value,
4994     *          'sdesc' => summary of inherited constant),
4995     *          ...
4996     *      </pre>
4997     */
4998
4999     function getFormattedInheritedConsts($child)
5000 {
5001     $package = $child-> docblock-> package;
5002     $subpackage = $child-> docblock-> subpackage;
5003     $ivars = $child-> getInheritedConsts($this);
5004     $results = array();
5005     if (!count($ivars)) return $results;
5006     foreach($ivars as $parent => $vars)
5007     {
5008         $file = $vars['file'];
5009         $vars = $vars['consts'];
5010         $par = $this-> classes-> getClass($parent,$file);
5011         if ($par) {
5012             $package = $par-> docblock-> package;
5013         }
5014         usort($vars,array($this,"sortVar" ));
5015         $result['parent_class'] = $this-> getClassLink($parent,$package);
5016         if (!$result['parent_class']) {
5017             $result['parent_class'] = $parent . ' (Internal Class)';
5018         }
5019         foreach($vars as $var)
5020         {
5021             $info = array();
5022
5023             if ($var-> docblock-> hasaccess) {
5024                 $info['access'] = $var-> docblock-> tags['access'][0]-> value;
5025             } else {
5026                 $info['access'] = 'public';
5027             }
5028             $info['name'] = $var-> getName();
5029             $info['link'] = $var-> getLink($this);
5030             if (!$info['link']) {
5031                 $info['link'] = $info['name'] . ' = ' . $var-> getValue();
5032             }
5033             $info['value'] = $this-> postProcess($var-> getValue());
5034             if ($var-> docblock)
5035                 $info['sdesc'] = $var-> docblock-> getSDesc($this);
5036             $result["iconsts"] [] = $info;
5037         }
5038         $results[] = $result;
5039         $result = array();
5040     }
5041     return $results;
5042 }
5043
5044 /**
5045 * Return a Smarty template object to operate with
5046 *
5047 * This returns a Smarty template with pre-initialized variables for use.
5048 * If the method "SmartyInit()" exists, it is called.
5049 * @return Smarty
5050 */
5051 function & newSmarty()
5052 {
5053     $templ = new Smarty;
5054     $templ-> use_sub_dirs = false;
5055     $templ-> template_dir = realpath($this-> smarty_dir . PATH_DELIMITER . 'templates');
5056     $templatename = get_class($this) . $this-> templateName;
5057     if (!file_exists($this-> targetDir . DIRECTORY_SEPARATOR . md5($templatename))) {
5058         // we'll delete this on finishing conversion
5059         $this-> _compiledDir[$this-> targetDir . DIRECTORY_SEPARATOR .
5060 md5($templatename)] = 1;
5061         mkdir($this-> targetDir . DIRECTORY_SEPARATOR . md5($templatename),0775);
5062     }
5063     $templ-> compile_dir = realpath($this-> targetDir . PATH_DELIMITER .
5064 md5($templatename));
5065     $templ-> config_dir = realpath($this-> smarty_dir . PATH_DELIMITER . 'configs');
5066     $templ-> assign("date" ,date("r" ,time()));
5067     $templ-> assign("maintitle" ,$_this-> title);
5068     $templ-> assign("package" ,$_this-> package);

```

```

5067     $templ-> assign("phpdocversion" ,  

5068     $templ-> assign("phpdocwebsite" ,  

5069     $templ-> assign("subpackage" ,  

5070     if (method_exists($this,'SmartyInit')) return $this-> SmartyInit($templ);  

5071     return $templ;  

5072   }  

5073  

5074   /**
5075    * Finish up parsing/cleanup directories
5076   */
5077   function cleanup()
5078   {
5079     foreach ($this-> _compiledDir as $dir => $one) {
5080       $this-> _rmdir($dir);
5081     }
5082   }
5083  

5084   /**
5085    * Completely remove a directory and its contents
5086   *
5087   * @param string $directory
5088   */
5089   function _rmdir($directory)
5090   {
5091     $handle = @opendir($directory);
5092     if ($handle) {
5093       while (false !== ($file = readdir($handle))) {
5094         if ($file == '.') || $file == '..') {
5095           continue;
5096         }
5097         if (is_dir($directory . DIRECTORY_SEPARATOR . $file)) {
5098           $this-> _rmdir($directory . DIRECTORY_SEPARATOR . $file);
5099         }
5100         @unlink($directory . DIRECTORY_SEPARATOR . $file);
5101       }
5102       closedir($handle);
5103       @rmdir($directory);
5104     }
5105   }
5106  

5107   /**
5108    * do all necessary output
5109    * @see Converter
5110    * @abstract
5111   */
5112   function Output($title)
5113   {
5114     phpDocumentor_out("WARNING: Generic Converter::Output was used, no output will be  

generated");
5115   }
5116  

5117   /**
5118    * Set the template directory with a different template base directory
5119    * @tutorial phpDocumentor.howto.pkg#using.command-line.templatebase
5120    * @param string template base directory
5121    * @param string template name
5122   */
5123   function setTemplateBase($base, $dir)
5124   {
5125     // remove trailing /'s from the base path, if any
5126     $base = str_replace('\\','/',$base);
5127     while ($base[strlen($base) - 1] == '/') $base = substr($base,0,strlen($base) - 1);
5128     $this-> templateName = substr($dir,0,strlen($dir) - 1);
5129     $this-> templateDir = $base . "/Converters/" . $this-> outputformat .  

"/";
5130     if (!is_dir($this-> templateDir))
5131     {
5132       addErrorDie(PDERROR_TEMPLATEDIR_DOESNT_EXIST, $this-> templateDir);
5133     }
5134  

5135     $this-> smarty_dir = $this-> templateDir;
5136     if (file_exists($this-> templateDir . PATH_DELIMITER . 'options.ini'))
5137     {
5138       // retrieve template options, allow array creation
5139       $this-> template_options = phpDocumentor_parse_ini_file($this-> templateDir .  

PATH_DELIMITER . 'options.ini',true);
5140     }
5141   }
5142  

5143   /**

```

```

5144 * sets the template directory based on the {@link $outputformat} and {@link $name}
5145 * Also sets {@link $templateName} to the $dir parameter
5146 * @param string subdirectory
5147 */
5148 function setTemplateDir($dir)
5149 {
5150     if ('@DATA-DIR@' != '@'. 'DATA-DIR@') {
5151         $templateBase = str_replace('\\', '/', '@DATA-DIR@/PhpDocumentor/phpDocumentor');
5152     } else {
5153         $templateBase = str_replace('\\', '/', $GLOBALS['_phpDocumentor_install_dir']) .
5154         '/phpDocumentor';
5155     }
5156     $this-> setTemplateBase($templateBase, $dir);
5157 }
5158 /**
5159 * Get the absolute path to the converter's base directory
5160 * @return string
5161 */
5162 function getConverterDir()
5163 {
5164     if ('@DATA-DIR@' != '@'. 'DATA-DIR@') {
5165         return str_replace('\\', '/', "@DATA-
5166 DIR@/PhpDocumentor/phpDocumentor/Converters/")
5167     } else {
5168         return str_replace('\\', '/', $GLOBALS['_phpDocumentor_install_dir'])
5169         .'./phpDocumentor/Converters/'
5170     }
5171 /**
5172 * Parse a global variable's default value for class initialization.
5173 *
5174 * If a global variable's default value is "new class" as in:
5175 * <code>
5176 * $globalvar = new Parser
5177 * </code>
5178 * This method will document it not as "new Parser" but instead as
5179 * "new {@link Parser}". For examples, see {@link phpdoc.inc}.
5180 * Many global variables are classes, and phpDocumentor links to their
5181 * documentation
5182 * @return string default global variable value with link to class if
5183 * it's "new Class"
5184 * @param string default value of a global variable.
5185 */
5186 function getGlobalValue($value)
5187 {
5188     if (strpos($value, 'new') === 0)
5189     {
5190         preg_match('/new([^(]*)(?:.|\\r|\\n)*)/', $value, $newval);
5191         if (isset($newval[1]))
5192         {
5193             $a = Converter::getLink(trim($newval[1]));
5194             if (!isset($newval[2])) $newval[2] = '';
5195             if ($a && $phpDocumentor_get_class($a) == 'classlink') $value = 'new
5196 .' . $this-> returnSee($a) .
5197                 $this-> postProcess($newval[2]);
5198             }
5199         }
5200         return $this-> postProcess($value);
5201     }
5202 /**
5203 * Parse an include's file to see if it is a file documented in this project
5204 *
5205 * Although not very smart yet, this method will try to look for the
5206 * included file file.ext:
5207 *
5208 * <code>
5209 * include ("file.ext");
5210 * </code>
5211 *
5212 * If it finds it, it will return a link to the file's documentation. As of
5213 * 1.2.0rc1, phpDocumentor is smarty enough to find these cases:
5214 * <ul>
5215 *   <li>absolute path to file</li>
5216 *   <li>./file.ext or ../file.ext</li>
5217 *   <li>relpath/to/file.ext if relpath is a subdirectory of the base parse

```

```

5219     *      directory</li>
5220     * </ul>
5221     * For examples, see {@link Setup.inc.php} includes.
5222     * Every include auto-links to the documentation for the file that is included
5223     * @return string included file with link to docs for file, if found
5224     * @param string file included by include statement.
5225     * @param string path of file that has the include statement
5226     */
5227 function getIncludeValue($value, $ipath)
5228 {
5229     preg_match('/"/([^\"]*\. [^\"]*)"/' , $value, $match);
5230     if (!isset($match[1]))
5231         preg_match('/\\"([^\"]*\. [^\"]*)\\"' , $value, $match);
5232     if (isset($match[1]))
5233     {
5234         $fancy_per = $this-> proceduralpages-> pathMatchesParsedFile($match[1], $ipath);
5235         if ($fancy_per)
5236         {
5237             $link = $this-> addLink($fancy_per);
5238             if (is_object($link) && phpDocumentor_get_class($link) == 'pagelink'
5239                 && isset($this-> all_packages[$link-> package]))
5240             {
5241                 $value = $this-> returnSee($link, $value);
5242             }
5243         } else
5244         {
5245             $per = Converter::getLink($match[1]);
5246             if (is_object($per) && phpDocumentor_get_class($per) == 'pagelink')
5247                 $value = $this-> returnSee($per);
5248         }
5249     }
5250     return $value;
5251 }
5252 /**
5253 * Recursively creates all subdirectories that don't exist in the $dir path
5254 * @param string $dir
5255 */
5256 function createParentDir($dir)
5257 {
5258     if (empty($dir)) return;
5259     $tmp = explode(SMART_PATH_DELIMITER, $dir);
5260     array_pop($tmp);
5261     $parent = implode(SMART_PATH_DELIMITER, $tmp);
5262     if ($parent != '' && ! file_exists($parent))
5263     {
5264         $test = @mkdir($parent, 0775);
5265         if (!$test)
5266         {
5267             $this-> createParentDir($parent);
5268             $test = @mkdir($parent, 0775);
5269             phpDocumentor_out(" Creating Parent Directory $parent\n");
5270         } else
5271         {
5272             phpDocumentor_out(" Creating Parent Directory $parent\n");
5273         }
5274     }
5275 }
5276 /**
5277 * Sets the output directory for generated documentation
5278 *
5279 * As of 1.3.0RC6, this also sets the compiled templates directory inside
5280 * the target directory
5281 * @param string $dir the output directory
5282 */
5283 function setTargetDir($dir)
5284 {
5285     if (strlen($dir) > 0)
5286     {
5287         $this-> targetDir = $dir;
5288         // if directory does exist create it, this should have more error checking in the
5289         // future
5290         if (!file_exists($dir))
5291         {
5292             $tmp =
5293             str_replace(array("/", "\\", ), SMART_PATH_DELIMITER, $dir);
5294             if (substr($tmp, -1) == SMART_PATH_DELIMITER)
5295             {

```

```

5296             $tmp = substr($tmp,0,(strlen($tmp)-1));
5297         }
5298         $this-> createParentDir($tmp);
5299         phpDocumentor_out("      Creating Directory $dir\n" );
5300         mkdir($dir,0775);
5301     } elseif (!is_dir($dir))
5302     {
5303         echo "      Output path: '$dir' is not a directory\n";
5304         die();
5305     }
5306 } else {
5307     echo "a target directory must be specified\n try phpdoc -h\n";
5308     die();
5309 }
5310 }

5312 /**
5313 * Writes a file to target dir
5314 * @param string
5315 * @param string
5316 * @param boolean true if the data is binary and not text
5317 */
5318 function writeFile($file,$data,$binary = false)
5319 {
5320     if (!file_exists($this-> targetDir))
5321     {
5322         mkdir($this-> targetDir,0775);
5323     }
5324     $string = '';
5325     if ($binary) $string = 'binary file ';
5326     phpDocumentor_out("      Writing $string" . $this-> targetDir . PATH_DELIMITER
. $file . "\n");
5327     flush();
5328     $write = 'w';
5329     if ($binary) $write = 'wb';
5330     $fp = fopen($this-> targetDir . PATH_DELIMITER . $file,$write);
5331     set_file_buffer($fp, 0);
5332     fwrite($fp,$data,strlen($data));
5333     fclose($fp);
5334 }

5336 /**
5337 * Copies a file from the template directory to the target directory
5338 * thanks to Robert Hoffmann for this fix
5339 * @param string
5340 */
5341 function copyFile($file, $subdir = '')
5342 {
5343     if (!file_exists($this-> targetDir))
5344     {
5345         mkdir($this-> targetDir,0775);
5346     }
5347     copy($this-> templateDir . $subdir . PATH_DELIMITER . $file, $this-> targetDir .
PATH_DELIMITER . $file);
5348 }

5350 /**
5351 * Return parserStringWithInlineTags::Convert() cache state
5352 * @see parserStringWithInlineTags::Convert()
5353 * @abstract
5354 */
5355 function getState()
5356 {
5357     return true;
5358 }

5360 /**
5361 * Compare parserStringWithInlineTags::Convert() cache state to $state
5362 * @param mixed
5363 * @see parserStringWithInlineTags::Convert()
5364 * @abstract
5365 */
5366 function checkState($state)
5367 {
5368     return true;
5369 }
5370 }

5371 }
5372 /**

```

```

5374     * @access private
5375     * @see Converter::getSortedClassTreeFromClass()
5376     */
5377     function rootcmp($a, $b)
5378     {
5379         return strnatcasecmp($a['class'], $b['class']);
5380     }
5381
5382     /**
5383     * @access private
5384     * @global string used to make the first tutorials converted the default package tutorials
5385     */
5386     function tutorialcmp($a, $b)
5387     {
5388         global $phpDocumentor_DefaultPackageName;
5389         if ($a == $phpDocumentor_DefaultPackageName) return -1;
5390         if ($b == $phpDocumentor_DefaultPackageName) return 1;
5391         return strnatcasecmp($a, $b);
5392     }
5393
5394     /**
5395     * smart htmlentities, doesn't entity the allowed tags list
5396     * Since version 1.1, this function uses htmlspecialchars instead of
5397     * htmlentities, for international support
5398     * This function has been replaced by functionality in {@link ParserDescCleanup.inc}
5399     * @param string $s
5400     * @return string browser-displayable page
5401     * @deprecated As of v1.2, No longer needed, as valid tags are parsed out of the source,
5402     * and everything else is {@link Converter::postProcess()} handled
5403     */
5404     function adv_htmlentities($s)
5405     {
5406         return;
5407         global $phpDocumentor__html,$_phpDocumentor_html_allowed;
5408         $result = htmlspecialchars($s);
5409         $entities = array_flip(get_html_translation_table(HTML_SPECIALCHARS));
5410         $result = strtr($result,$phpDocumentor__html);
5411         $matches = array();
5412         preg_match_all('/(<img.*>)/U' , $result,$matches);
5413         for($i=0;$i< count($matches[1]);$i++)
5414         {
5415             $result =
5416             str_replace($matches[1][$i],strtr($matches[1][$i],array_flip(get_html_translation_table(HTML_SPECIALCHARS))),$result);
5417         }
5418         preg_match_all('/(<font.*>)/U' , $result,$matches);
5419         for($i=0;$i< count($matches[1]);$i++)
5420         {
5421             $result =
5422             str_replace($matches[1][$i],strtr($matches[1][$i],array_flip(get_html_translation_table(HTML_SPECIALCHARS))),$result);
5423         }
5424         preg_match_all('/(<ol.*>)/U' , $result,$matches);
5425         for($i=0;$i< count($matches[1]);$i++)
5426         {
5427             $result =
5428             str_replace($matches[1][$i],strtr($matches[1][$i],array_flip(get_html_translation_table(HTML_SPECIALCHARS))),$result);
5429         }
5430         preg_match_all('/(<ul.*>)/U' , $result,$matches);
5431         for($i=0;$i< count($matches[1]);$i++)
5432         {
5433             $result =
5434             str_replace($matches[1][$i],strtr($matches[1][$i],array_flip(get_html_translation_table(HTML_SPECIALCHARS))),$result);
5435         }
5436         preg_match_all('/(<li.*>)/U' , $result,$matches);
5437         for($i=0;$i< count($matches[1]);$i++)
5438         {
5439             $result =
5440             str_replace($matches[1][$i],strtr($matches[1][$i],array_flip(get_html_translation_table(HTML_SPECIALCHARS))),$result);
5441         }

```

```
5442     return $result;
5443 }
5444
5445 /**
5446 * Used solely for setting up the @uses list
5447 * @package ignore
5448 * @ignore
5449 */
5450 class __dummyConverter extends Converter
5451 {
5452     function setTemplateDir(){}
5453     function setTargetDir(){}
5454     function getPageName(& $element)
5455     {
5456         if (phpDocumentor_get_class($element) == 'parserpage') return '_'.$element-
> getName();
5457         return '_'.$element-> parent-> getName();
5458     }
5459 }
5460 ?>
```

# File Source for CHMdefaultConverter.inc

Documentation for this file is available at [CHMdefaultConverter.inc](#)

```
1  <?php
2  /**
3  * CHM (Compiled Help Manual) output converter for Smarty Template.
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @package Converters
29 * @subpackage CHMdefault
30 * @author Joshua Eichorn <jeichorn@phpdoc.org>
31 * @author Greg Beaver <cellog@php.net>
32 * @copyright 2000-2006 Joshua Eichorn, Gregory Beaver
33 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34 * @version CVS: $Id: CHMdefaultConverter.inc 234145 2007-04-19 20:20:57Z ashnazg $
35 * @filesource
36 * @link http://www.phpdoc.org
37 * @link http://pear.php.net/PhpDocumentor
38 * @since 1.0rc1
39 */
40 /**
41 * Generates files that MS HTML Help Workshop can use to create a MS Windows
42 * compiled help file (CHM)
43 *
44 * The free MS HTML Help compiler takes the project file (phpdoc.hhp) and reads
45 * the table of contents file specified in the project (which is always contents.hhc
46 * in phpDocumentor). When the converter reaches stable state, it will also
47 * output an index file index.hhk. The free download for MS HTML Help Workshop
48 * is available below
49 * @link http://www.microsoft.com/downloads/release.asp?releaseid=33071 MS HTML Help Workshop
download
50 * @package Converters
51 * @subpackage CHMdefault
52 * @author Greg Beaver <cellog@php.net>
53 * @since 1.0rc1
54 * @version $Revision: 234145 $
55 */
56 class CHMdefaultConverter extends Converter
57 {
58 /**
59 * CHMdefaultConverter wants elements sorted by type as well as alphabetically
60 * @see Converter::$sort_page_contents_by_type
61 * @var boolean
62 */
63 var $sort_page_contents_by_type = true;
64 /** @var string */
65 var $outputformat = 'CHM';
66 /** @var string */
```

```

67     var $name = 'default';
68     /**
69      * indexes of elements by package that need to be generated
70      * @var array
71      */
72     var $leftindex = array('classes' => true, 'pages' => true, 'functions' => true,
73 'defines' => false, 'globals' => false);
74     /**
75      * output directory for the current procedural page being processed
76      * @var string
77      */
78     var $page_dir;
79     /**
80      * target directory passed on the command-line.
81      * {@link $targetDir} is malleable, always adding package/ and package/subpackage/
82      subdirectories onto it.
83      * @var string
84      */
85     var $base_dir;
86     /**
87      * output directory for the current class being processed
88      * @var string
89      */
90     var $class_dir;
91     /**
92      * array of converted package page names.
93      * Used to link to the package page in the left index
94      * @var array Format: array(package => 1)
95      */
96     var $package_pages = array();
97     /**
98      * controls formatting of parser informative output
99      *
100     * Converter prints:
101     * "Converting /path/to/file.php... Procedural Page Elements... Classes..."
102     * Since CHMdefaultConverter outputs files while converting, it needs to send a \n to start
103     a new line. However, if there
104     * is more than one class, output is messy, with multiple \n's just between class file
105     output. This variable prevents that
106     * and is purely cosmetic
107     * @var boolean
108     */
109     var $juststarted = false;
110     /**
111      * contains all of the template procedural page element loop data needed for the current
112      template
113      * @var array
114      */
115     var $current;
116     /**
117      * contains all of the template class element loop data needed for the current template
118      * @var array
119      */
120     var $currentclass;
121     var $wrote = false;
122     var $ric_set = array();
123     /**
124      * Table of Contents entry for index.hhk
125      * @var array
126      */
127     var $KLinks = array();
128     /**
129      * sets {@link $base_dir} to $targetDir
130      * @see Converter()
131      */
132     function CHMdefaultConverter(& $allp, & $packp, & $classes, & $procpages, $po,
133 $pp, $qm, $targetDir, $templateDir, $title)
134     {
135         Converter::Converter($allp, $packp, $classes, $procpages, $po, $pp, $qm, $targetDir,
136 $templateDir, $title);
137         $this-> base_dir = $targetDir;
138     }

```

```

140
141  /**
142   * @deprecated in favor of PHP 4.3.0+ tokenizer-based source highlighting
143   */
144  function unmangle($sourcecode)
145  {
146      $sourcecode = str_replace('<code>', '<pre>', $sourcecode);
147      $sourcecode = str_replace('</code>', '</pre>', $sourcecode);
148      $sourcecode = str_replace('<br />', '\n', $sourcecode);
149      $sourcecode = str_replace('&nbsp;', ' ', $sourcecode);
150      $sourcecode = str_replace('&lt;', '<', $sourcecode);
151      $sourcecode = str_replace('&gt;', '>', $sourcecode);
152      $sourcecode = str_replace('&amp;', '&', $sourcecode);
153      return $sourcecode;
154  }
155
156 /**
157  * @param string full path to the source file
158  * @param string fully highlighted source code
159  */
160 function writeSource($path, $value)
161 {
162     $templ = & $this-> newSmarty();
163     $pathinfo = $this-> proceduralpages-> getPathInfo($path, $this);
164     $templ-> assign('source', $value);
165     $templ-> assign('package', $pathinfo['package']);
166     $templ-> assign('subpackage', $pathinfo['subpackage']);
167     $templ-> assign('name', $pathinfo['name']);
168     $templ-> assign('source_loc', $pathinfo['source_loc']);
169     $templ-> assign('docs', $pathinfo['docs']);
170     $templ-> assign("subdir", '../');
171     $templ-> register_outputfilter('CHMdefault_outputfilter');
172     $this-> setTargetDir($this-> getFileSourcePath($this-> base_dir));
173     $this-> addSourceTOC($pathinfo['name'], $this-
> getFileSourceName($path), $pathinfo['package'], $pathinfo['subpackage'], true);
174     phpDocumentor_out("\n");
175     $this-> setSourcePaths($path);
176     $this-> writefile($this-> getFileSourceName($path).'.html', $templ-
> fetch('filesource.tpl'));
177 }
178
179 function writeExample($title, $path, $source)
180 {
181     $templ = & $this-> newSmarty();
182     $templ-> assign('source', $source);
183     if (empty($title))
184     {
185         $title = 'example';
186         addWarning(PDERROR_EMPTY_EXAMPLE_TITLE, $path, $title);
187     }
188     $templ-> assign('title', $title);
189     $templ-> assign('file', $path);
190     $templ-> assign("subdir", '../');
191     $templ-> register_outputfilter('CHMdefault_outputfilter');
192     $pathinfo = $this-> proceduralpages-> getPathInfo($path, $this);
193     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . '__examplesource');
194     $this-
> addSourceTOC($title, 'exsource_'.$path, $pathinfo['package'], $pathinfo['subpackage'], false);
195     phpDocumentor_out("\n");
196     $this-> writefile('exsource_'.$path.'.html', $templ-> fetch('examplesource.tpl'));
197 }
198
199 function getExampleLink($path, $title)
200 {
201     return $this-> returnLink('{$subdir}__examplesource' . PATH_DELIMITER .
'__exsource_'.$path.'.html', $title);
202 }
203
204 function getSourceLink($path)
205 {
206     return $this-> returnLink('{$subdir}__filesource/' .
$this-> getFileSourceName($path).'.html', 'Source Code for this file');
207 }
208
209 /**
210  * Retrieve a Converter-specific anchor to a segment of a source code file
211  * parsed via a {@tutorial tags.filesource.pkg} tag.
212  * @param string full path to source file
213  * @param string name of anchor
214  * @param string link text, if this is a link

```

```

216 * @param boolean returns either a link or a destination based on this
217 * parameter
218 * @return string link to an anchor, or the anchor
219 */
220 function getSourceAnchor($sourcefile,$anchor,$text = '' , $link = false)
221 {
222     if ($link) {
223         return $this-> returnLink('{$subdir}__filesource/' .
224             $this-> getFileSourceName($sourcefile) . '.html#a' . $anchor, $text);
225     } else {
226         return '<a name="a' . $anchor. '"></a>' ;
227     }
228 }
229
230 /**
231 * Return a line of highlighted source code with formatted line number
232 *
233 * If the $path is a full path, then an anchor to the line number will be
234 * added as well
235 * @param integer line number
236 * @param string highlighted source code line
237 * @param false/string full path to @filesource file this line is a part of,
238 * if this is a single line from a complete file.
239 * @return string formatted source code line with line number
240 */
241 function sourceLine($linenumber, $line, $path = false)
242 {
243     $extra = '';
244     if (strlen(str_replace("\n" , ' ', $line)) == 0) {
245         $extra = ' ' ;
246     }
247     if ($path)
248     {
249         return '<li><div class="src-line">' . $this-
> getSourceAnchor($path, $linenumber) .
250             str_replace("\n" , ' ', $line) . $extra .
251             "</div></li>\n" ;
252     }
253     else
254     {
255         return '<li><div class="src-line">' .
256             str_replace("\n" , ' ', $line) .
257             " " . $extra . "</div></li>\n" ;
258     }
259 }
260 /**
261 * Used to convert the <> tag in a docblock
262 * @param string
263 * @param boolean
264 * @return string
265 */
266 function ProgramExample($example, $tutorial = false, $inlinesourceparse = null/*false*/,
267                         $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
null/*false*/)
268 {
269     return $this-> PreserveWhiteSpace(parent::ProgramExample($example, $tutorial,
$inlinesourceparse, $class, $linenum, $filesourcepath));
270 }
271 /**
272 * @param string
273 */
274 function TutorialExample($example)
275 {
276     $trans = $this-> template_options['desctranslate'];
277     $this-> template_options['desctranslate'] = array();
278     $example = '<ol>' . parent::TutorialExample($example)
279             . '</ol>' ;
280     $this-> template_options['desctranslate'] = $trans;
281     if (!isset($this-> template_options['desctranslate'])) return $example;
282     if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
283     $example = $this-> template_options['desctranslate'][['code']] . $example;
284     if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
285     return $example . $this-> template_options['desctranslate'][['code']];
286 }
287
288 function getCurrentPageLink()
289 {
290     return $this-> curname . '.html';

```

```

292     }
293
294     /**
295      * Uses htmlspecialchars() on the input
296      */
297     function postProcess($text)
298     {
299         if ($this-> highlightingSource) {
300             return str_replace(array(' ', "\t"), array(' ', ' ', ,
301 'nbsp;&nbsp;'), htmlspecialchars($text));
302         }
303         return htmlspecialchars($text);
304     }
305
306     /**
307      * Use the template tutorial_toc.tpl to generate a table of contents for HTML
308      * @return string table of contents formatted for use in the current output format
309      * @param array format: array(array('tagname' => section, 'link' => returnsee link,
310      'id' => anchor name, 'title' => from title tag),...)
311     */
312     function formatTutorialTOC($toc)
313     {
314         $template = & $this-> newSmarty();
315         $template-> assign('toc', $toc);
316         return $template-> fetch('tutorial_toc.tpl');
317     }
318
319     function & SmartyInit(& $templ)
320     {
321         if (!isset($this-> package_index))
322             foreach($this-> all_packages as $key => $val)
323             {
324                 if (isset($this-> pkg_elements[$key]))
325                 {
326                     if (!isset($start)) $start = $key;
327                     $this-> package_index[] = array('link' => "li_$key.html" ,
328 'title' => $key);
329                 }
330                 $templ-> assign("packageindex" , $this-> package_index);
331                 $templ-> assign("subdir" , '');
332             }
333
334
335     /**
336      * Writes out the template file of {@link $class_data} and unsets the template to save
337      * memory
338      * @see registerCurrentClass()
339      * @see parent::endClass()
340     */
341     function endClass()
342     {
343         $a = '../';
344         if (!empty($this-> subpackage)) $a .= '../';
345         if ($this-> juststarted)
346         {
347             $this-> juststarted = false;
348             phpDocumentor_out("\n");
349             flush();
350         }
351         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> class_dir);
352         $this-> class_data-> assign("subdir" , $a);
353         $this-> class_data-> register_outputfilter('CHMdefault_outputfilter');
354         $this-> addTOC($this-> class, $this-> class, $this-> package, $this-
355 > subpackage, true);
356         $this-> writefile($this-> class . '.html', $this-> class_data-
357 > fetch('class.tpl'));
358         unset($this-> class_data);
359     }
360
361     /**
362      * Writes out the template file of {@link $page_data} and unsets the template to save memory
363      * @see registerCurrent()
364      * @see parent::endPage()
365     */
366     function endPage()
367     {
368         $this-> package = $this-> curpage-> package;

```

```

366     $this-> subpackage = $this-> curpage-> subpackage;
367     $a = '../';
368     if (!empty($this-> subpackage)) $a .= '../';
369     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
370     $this-> page_data-> assign("package" , $this-> package);
371     $this-> page_data-> assign("subdir" , $a);
372     $this-> page_data-> register_outputfilter('CHMdefault_outputfilter');
373     $this-> addTOC($this-> curpage-> file,$this-> page,$this-> package,$this-
> subpackage);
374     $this-> writefile($this-> page . '.html',$this-> page_data-> fetch('page.tpl'));
375     unset($this-> page_data);
376 }
377 /**
378  * @param string
379  * @param string
380  * @return string <a href='".$link."'>.$text.</a>;
381  */
382 function returnLink($link,$text)
383 {
384     return '<a href=""' . $link. '">' . $text. '</a>';
385 }
386 }
387 /**
388  * CHMdefaultConverter chooses to format both package indexes and the complete index here
389  *
390  * This function formats output for the elementindex.html and pkgelementindex.html template
391  * files. It then
392  * writes them to the target directory
393  * @see generateElementIndex(), generatePkgElementIndex()
394  */
395 function formatPkgIndex()
396 {
397     list($package_indexes,$packages,$letters) = $this-> generatePkgElementIndexes();
398     for($i=0;$i< count($package_indexes);$i++)
399     {
400         $template = & $this-> newSmarty();
401         $this-> package = $package_indexes[$i]['package'];
402         $this-> subpackage = '';
403         $template-> assign("index" , $package_indexes[$i]['pindex']);
404         $template-> assign("package" , $package_indexes[$i]['package']);
405         $template-
> assign("letters" , $letters[$package_indexes[$i]['package']]);
406         $template-> assign("title" , "Package
" . $package_indexes[$i]['package'] . " Element Index" );
407         $template-> assign("subdir" , '../');
408         $template-> register_outputfilter('CHMdefault_outputfilter');
409         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER .
$package_indexes[$i]['package']);
410         $this-> addTOC($package_indexes[$i]['package'] . " Alphabetical
Index" , 'elementindex_'. $package_indexes[$i]['package'], $package_indexes[$i]['package'], '');
411         $this-
> writefile('elementindex_'. $package_indexes[$i]['package'] . '.html', $template-
> fetch('pkgelementindex.tpl'));
412         }
413         phpDocumentor_out("\n");
414         flush();
415     }
416 /**
417  * CHMdefaultConverter uses this function to format template index.html and packages.html
418  *
419  * This function generates the package list from {@link $all_packages}, eliminating any
420  * packages that don't have any entries in their package index (no files at all, due to
421  * @ignore
422  * or other factors). Then it uses the default package name as the first package index to
423  * display.
424  * It sets the right pane to be either a blank file with instructions on making package-
425  * level docs,
426  * or the package-level docs for the default package.
427  * @global string Used to set the starting package to display
428  */
429 function formatIndex()
430 {
431     global $phpDocumentor_DefaultPackageName;
432     list($elindex,$letters) = $this-> generateElementIndex();
433     $template = & $this-> newSmarty();
434     $template-> assign("index" , $elindex);
435     $template-> assign("letters" , $letters);
436     $template-> assign("title" , "Element Index" );

```

```

435     $template-> assign("date" ,date("r" ,time()));
436     $template-> register_outputfilter('CHMdefault_outputfilter');
437     phpDocumentor_out("\n" );
438     flush();
439     $this-> setTargetDir($this-> base_dir);
440     $this-> addTOC("Alphabetical Index Of All
441     'elementindex' 'Index" ,');
442     $this-> writefile('elementindex.html',$template-> fetch('elementindex.tpl'));
443     usort($this-> package_index,"CHMdefault_pindexcmp" );
444     $index = & $this-> newSmarty();
445     foreach($this-> all_packages as $key => $val)
446     {
447         if (isset($this-> pkg_elements[$key]))
448         {
449             if (!isset($start)) $start = $key;
450             if (!isset($this-> package_pages[$key])) $this-> writeNewPPage($key);
451         }
452     $this-> setTargetDir($this-> base_dir);
453     // Created index.html
454     if (isset($this-> pkg_elements[$phpDocumentor_DefaultPackageName])) $start =
455     $phpDocumentor_DefaultPackageName;
456     $this-> package = $start;
457     $this-> subpackage = '';
458     $setalready = false;
459     if (isset($this-> tutorials[$start]['']['pkg']))
460     {
461         foreach($this-> tutorials[$start]['']['pkg'] as $tute)
462         {
463             if ($tute-> name == $start . '.pkg')
464             {
465                 $setalready = true;
466                 $this-> addTOC("Start page" , $start . '/tutorial_' . $tute-
467             > name, "Index" );
468         }
469     if (!$setalready)
470     {
471         if (isset($this-> package_pages[$start]))
472         {
473             $this-> addTOC("Start
474             page" , 'package_' . $start, "Index" );
475         else
476         {
477             $index-> assign("blank" , "blank" );
478             $blank = & $this-> newSmarty();
479             $blank-> assign('package' , $phpDocumentor_DefaultPackageName);
480             $this-> addTOC("Start page" , 'blank' , "Index" );
481             $this-> writefile("blank.html" , $blank-> fetch('blank.tpl'));
482             Converter::writefile('index.html' , $blank-> fetch('tutorial.tpl'));
483         }
484     }
485     phpDocumentor_out("\n" );
486     flush();
487
488     unset($index);
489 }
490
491     function writeNewPPage($key)
492     {
493         return;
494         $template = & $this-> newSmarty();
495         $this-> package = $key;
496         $this-> subpackage = '';
497         $template-> assign("date" ,date("r" ,time()));
498         $template-> assign("title" ,$this-> title);
499         $template-> assign("package" , $key);
500         $template-> register_outputfilter('CHMdefault_outputfilter');
501         phpDocumentor_out("\n" );
502         flush();
503         $this-> setTargetDir($this-> base_dir);
504
505         $this-> addTOC(" $key Index" , " li_$key" , $key, '');
506         $this-> writefile(" li_$key.html" , $template-> fetch('index.tpl'));
507         unset($template);
508     }
509
510 /**

```

```

511     * Generate indexes for li_package.html and classtree output files
512     *
513     * This function generates the li_package.html files from the template file left.html. It
514     * does this by
515     * iterating through each of the $page_elements, $class_elements and $function_elements
516     * arrays to retrieve
517     * the pre-sorted {@link abstractLink} descendants needed for index generation. Conversion
518     * of these links to
519     * text is done by {@link returnSee()}. The {@link $local} parameter is set to false to
520     * ensure that paths are correct.
521     *
522     * Then it uses {@link generateFormattedClassTrees()} to create class trees from the
523     * template file classtrees.html. Output
524     * filename is classtrees_packageName.html. This function also unsets {@link $elements}
525     * and {@link $pkg_elements} to free
526     * up the considerable memory these two class vars use
527     * @see $page_elements, $class_elements, $function_elements
528     */
529     function formatLeftIndex()
530     {
531         phpDocumentor_out("\n");
532         flush();
533         $this-> setTargetDir($this-> base_dir);
534         if (0)//!isset($this->left))
535         {
536             debug("Nothing parsed, check the command-line");
537             die();
538         }
539         foreach($this-> all_packages as $package => $rest)
540         {
541             if (!isset($this-> pkg_elements[$package])) continue;
542
543             // Create class tree page
544             $template = & $this-> newSmarty();
545             $template-> assign("classtrees" , $this-
546             > generateFormattedClassTrees($package));
547             $template-> assign("package" , $package);
548             $template-> assign("date" , date("r" , time()));
549             $template-> register_outputfilter('CHMdefault_outputfilter');
550             $this-> addTOC(" $package Class
551             Trees" , " classtrees_$package" , $package, '');
552             $this-> writefile(" classtrees_$package.html" , $template-
553             > fetch('classtrees.tpl'));
554             phpDocumentor_out("\n");
555             flush();
556
557             // free up considerable memory
558             unset($this-> elements);
559             unset($this-> pkg_elements);
560         }
561
562         /**
563         * This function takes an {@link abstractLink} descendant and returns an html link
564         *
565         * @param abstractLink a descendant of abstractlink should be passed, and never text
566         * @param string text to display in the link
567         * @param boolean this parameter is not used, and is deprecated
568         * @param boolean determines whether the returned text is enclosed in an <a> tag
569         */
570         function returnSee(& $element, $eltext = false, $with_a = true)
571         {
572             if (!$element) return false;
573             if (!$with_a) return $this-> getId($element, false);
574             if (!$eltext)
575             {
576                 $eltext = '';
577                 switch($element-> type)
578                 {
579                     case 'tutorial' :
580                         $eltext = strip_tags($element-> title);
581                         break;
582                     case 'method' :
583                     case 'var' :
584                     case 'const' :
585                         $eltext .= $element-> class.'::';
586                     case 'page' :
587                     case 'define' :
588                     case 'class' :
589                     case 'function' :
590                     case 'global' :

```

```

582         default :
583             $eltext .= $element-> name;
584             if ($element-> type == 'function' || $element-> type == 'method') $eltext
585             .= '()';
586         }
587     }
588     return '<a href="" .' . $this-> getId($element) . '">' . $eltext. '</a>' ;
589 }
590
591 function getId($element, $fullpath = true)
592 {
593     if (phpDocumentor_get_class($element) == 'parserdata')
594     {
595         $element = $this-> addLink($element-> parent);
596         $elp = $element-> parent;
597     } elseif (is_a($element, 'parserbase'))
598     {
599         $elp = $element;
600         $element = $this-> addLink($element);
601     }
602     $c = '';
603     if (!empty($element-> subpackage))
604     {
605         $c = '/' . $element-> subpackage;
606     }
607     $b = "{$subdir}";
608     switch ($element-> type)
609     {
610         case 'page' :
611             if ($fullpath)
612                 return $b.$element-> package.$c.'/'.$element-> fileAlias.'.html';
613             return 'top';
614             break;
615         case 'define' :
616         case 'global' :
617         case 'function' :
618             if ($fullpath)
619                 return $b.$element-> package.$c.'/'.$element-> fileAlias.'.html#' . $element-
620             > type.$element-> name;
621             return $element-> type.$element-> name;
622             break;
623         case 'class' :
624             if ($fullpath)
625                 return $b.$element-> package.$c.'/'.$element-> name.'.html';
626             return 'top';
627             break;
628         case 'method' :
629         case 'var' :
630         case 'const' :
631             if ($fullpath)
632                 return $b.$element-> package.$c.'/'.$element-> class.'.html#' . $element-
633             > type.$element-> name;
634             return $element-> type.$element-> name;
635             break;
636         case 'tutorial' :
637             $d = '';
638             if ($element-> section)
639             {
640                 $d = '#'.$element-> section;
641             }
642             return $b.$element-> package.$c.'/tutorial_'.$element-> name.'.html' . $d;
643     }
644
645     function ConvertTodoList()
646     {
647         $todolist = array();
648         foreach($this-> todoList as $package => $alltodos)
649         {
650             foreach($alltodos as $todos)
651             {
652                 $converted = array();
653                 $converted['link'] = $this-> returnSee($todos[0]);
654                 if (!is_array($todos[1]))
655                 {
656                     $converted['todos'][] = $todos[1]-> Convert($this);
657                 } else
658                 {
659                     foreach($todos[1] as $todo)
660                     {
661                         $converted['todos'][] = $todo-> Convert($this);
662                     }
663                 }
664             }
665             $todolist[$package] = $converted;
666         }
667         return $todolist;
668     }

```

```

659             {
660                 $converted['todos'][] = $todo-> Convert($this);
661             }
662         $todolist[$package][] = $converted;
663     }
664 }
665 $templ = & $this-> newSmarty();
666 $templ-> assign('todos',$todolist);
667 $templ-> register_outputfilter('CHMdefault_outputfilter');
668 $this-> setTargetDir($this-> base_dir);
669 $this-> addTOC('Todo List','todolist','Index','','false,true');
670 $this-> addKLink('Todo List', 'todolist', '', 'Development');
671 $this-> writefile('todolist.html',$templ-> fetch('todolist.tpl'));
672 }
673 /**
674 * Convert README/INSTALL/CHANGELOG file contents to output format
675 * @param README/INSTALL/CHANGELOG
676 * @param string contents of the file
677 */
678 function Convert_RIC($name, $contents)
679 {
680     $template = & $this-> newSmarty();
681     $template-> assign('contents',$contents);
682     $template-> assign('name',$name);
683     $this-> setTargetDir($this-> base_dir);
684     $this-> addTOC($name,'ric_'.$name,'Index','','false,true');
685     $this-> addKLink($name, 'ric_'.$name, '', 'Development');
686     $this-> writefile('ric_'.$name . '.html',$template-> fetch('ric.tpl'));
687     $this-> ric_set[$name] = true;
688 }
689 /**
690 * Create errors.html template file output
691 *
692 * This method takes all parsing errors and warnings and spits them out ordered by file and
693 * line number.
694 * @global ErrorTracker We'll be using it's output facility
695 */
696 function ConvertErrorLog()
697 {
698     global $phpDocumentor_errors;
699     $allfiles = array();
700     $files = array();
701     $warnings = $phpDocumentor_errors-> returnWarnings();
702     $errors = $phpDocumentor_errors-> returnErrors();
703     $template = & $this-> newSmarty();
704     foreach($warnings as $warning)
705     {
706         $file = '##none';
707         $linenum = 'Warning';
708         if ($warning-> file)
709         {
710             $file = $warning-> file;
711             $allfiles[$file] = 1;
712             $linenum .= ' on line ' . $warning-> linenum;
713         }
714         $files[$file]['warnings'][] = array('name' => $linenum, 'listing' => $warning-
715 > data);
716     }
717     foreach($errors as $error)
718     {
719         $file = '##none';
720         $linenum = 'Error';
721         if ($error-> file)
722         {
723             $file = $error-> file;
724             $allfiles[$file] = 1;
725             $linenum .= ' on line ' . $error-> linenum;
726         }
727         $files[$file]['errors'][] = array('name' => $linenum, 'listing' => $error-
728 > data);
729     }
730     $i=1;
731     $af = array();
732     foreach($allfiles as $file => $num)
733     {
734         $af[$i++] = $file;
735     }

```

```

736     $allfiles = $af;
737     usort($allfiles,'strnatcasecmp');
738     $allfiles[0] = "Post-parsing" ;
739     foreach($allfiles as $i =>    $a)
740     {
741         $allfiles[$i] = array('file' =>    $a);
742     }
743     $out = array();
744     foreach($files as $file =>    $data)
745     {
746         if ($file == '##none') $file = 'Post-parsing';
747         $out[$file] = $data;
748     }
749     $template-> assign("files" , $allfiles);
750     $template-> assign("all" , $out);
751     $template-> assign("title" , "phpDocumentor Parser Errors and
752     Warnings");
753     $this-> setTargetDir($this-> base_dir);
754     $this-> writefile("errors.html" , $template-> fetch('errors.tpl'));
755     unset($template);
756     phpDocumentor_out("\n\nTo view errors and warnings, look at "
757     . $this-> PATH_DELIMITER . "errors.html\n");
758     flush();
759 }
760
761     function getCData($value)
762     {
763         return '<pre>' . htmlentities($value) . '</pre>';
764     }
765
766     function getTutorialId($package,$subpackage,$tutorial,$id)
767     {
768         return $id;
769     }
770
771     /**
772      * Converts package page and sets its package as used in {@link $package_pages}
773      * @param parserPackagePage
774      */
775     function convertPackagepage(&    $element)
776     {
777         phpDocumentor_out("\n");
778         flush();
779         $this-> package = $element-> package;
780         $this-> subpackage = '';
781         $contents = $element-> Convert($this);
782         $this-> package_pages[$element-> package] =
783         str_replace('{subdir}', '../', $contents);
784         phpDocumentor_out("\n");
785         flush();
786         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $element-> package);
787         $this-> addTOC($element-> package." Tutorial" , 'package_'.$element-
788 > package,$element-> package,'');
789         $this-> writefile('package_'.$element-
790 > package.'.html',str_replace('{subdir}', '../', $contents));
791         $this-> setTargetDir($this-> base_dir);
792         Converter::writefile('index.html',str_replace('{subdir}', '', $contents));
793         $this-> addKLink($element-> package." Tutorial" , 'package_'.$element-
794 > package, '' , 'Tutorials');
795     }
796
797     /**
798      * @param parserTutorial
799      */
800     function convertTutorial(&    $element)
801     {
802         phpDocumentor_out("\n");
803         flush();
804         $template = & parent::convertTutorial($element);
805         $a = '../';
806         if ($element-> subpackage) $a .= '../';
807         $template-> assign('subdir',$a);
808         $template-> register_outputfilter('CHMdefault_outputfilter');
809         $contents = $template-> fetch('tutorial.tpl');
810         if ($element-> package == $GLOBALS['phpDocumentor_DefaultPackageName'] &&
811 empty($element-> subpackage) && (    $element-> name == $element-> package . '.pkg'))
812         {
813             $template-> assign('subdir','');
814             $this-> setTargetDir($this-> base_dir);
815             Converter::writefile('index.html',$template-> fetch('tutorial.tpl'));

```

```

809     }
810     $a = '';
811     if ($element-> subpackage) $a = PATH_DELIMITER . $element-> subpackage;
812     phpDocumentor_out("\n");
813     flush();
814     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $element-> package . $a);
815     $this-> addTOC($a = strip_tags($element-> getTitle($this)), 'tutorial_' . $element-
> name,
816                   $element-> package, $element-> subpackage, false, true);
817     $this-> writeFile('tutorial_' . $element-> name . '.html', $contents);
818     $this-> addKLink($element-> getTitle($this), $element-> package . $a .
PATH_DELIMITER . 'tutorial_' . $element-> name,
819                   '', 'Tutorials');
820   }
821
822 /**
823  * Converts class for template output
824  * @see prepareDocBlock(), generateChildClassList(), generateFormattedClassTree(),
825  * getFormattedConflicts()
826  * @see getFormattedInheritedMethods(), getFormattedInheritedVars()
827  * @param parserClass
828  */
829 function convertClass(& $element)
830 {
  parent::convertClass($element);
831   $this-> class_dir = $element-> docblock-> package;
832   if (!empty($element-> docblock-> subpackage)) $this-> class_dir .= PATH_DELIMITER
. $element->
docblock-> subpackage;
833   $a = './';
834   if ($element-> docblock-> subpackage != '') $a = "./$a";
835
836   $this-> class_data-> assign('subdir', $a);
837   $this-> class_data-> assign("title" , "Docs For Class " .
838   $element-> getName());
839   $this-> class_data-> assign("page" , $element-> getName() . '.html');
840   $this-> addKLink($element-> name, $this-> class_dir . PATH_DELIMITER . $this-
> class, '' , 'Classes');
841 }
842
843 /**
844  * Converts class variables for template output
845  * @see prepareDocBlock(), getFormattedConflicts()
846  * @param parserDefine
847  */
848 function convertVar(& $element)
849 {
  parent::convertVar($element, array('var_dest' => $this-> getId($element, false)));
850   $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> class_dir);
851   $this-> addKLink($element-> name, $this-> class_dir . PATH_DELIMITER . $this-
> class, $this-> getId($element, false), $element-> class.' Properties');
852 }
853
854 /**
855  * Converts class constants for template output
856  * @see prepareDocBlock(), getFormattedConflicts()
857  * @param parserDefine
858  */
859 function convertConst(& $element)
860 {
  parent::convertConst($element, array('const_dest' => $this-
> getId($element, false)));
861   $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> class_dir);
862   $this-> addKLink($element-> name, $this-> class_dir . PATH_DELIMITER . $this-
> class, $this-> getId($element, false), $element-> class.' Constants');
863 }
864
865 /**
866  * Converts class methods for template output
867  * @see prepareDocBlock(), getFormattedConflicts()
868  * @param parserDefine
869  */
870 function convertMethod(& $element)
871 {
  parent::convertMethod($element, array('method_dest' => $this-
> getId($element, false)));
872   $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> class_dir);
873   $this-> addKLink($element-> name, $this-> class_dir . PATH_DELIMITER . $this-
> class, $this-> getId($element, false), $element-> class.' Methods');
874 }

```

```

878
879     /**
880      * Converts function for template output
881      * @see prepareDocBlock(), parserFunction::getFunctionCall(), getFormattedConflicts()
882      * @param parserFunction
883      */
884     function convertFunction(& $element)
885     {
886         $funcloc = $this-> getId($this-> addLink($element));
887         parent::convertFunction($element, array('function_dest' => $this-
> getId($element, false)));
888         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
889         $this-> addKLink($element-> name, $this-> page_dir . PATH_DELIMITER . $this-
> page, $this-> getId($element, false), 'Functions');
890     }
891
892     /**
893      * Converts include elements for template output
894      * @see prepareDocBlock()
895      * @param parserInclude
896      */
897     function convertInclude(& $element)
898     {
899         parent::convertInclude($element, array('include_file' => '_'.strtr($element-
> getValue(), array("'" => "'", '"' => '"', '.' => '_'))));
900         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
901         $this-> addKLink(str_replace("'", '"', $element-> getValue()), $this-
> page_dir . PATH_DELIMITER . $this-> page, '_', ucfirst($element-> name));
902     }
903
904     /**
905      * Converts defines for template output
906      * @see prepareDocBlock(), getFormattedConflicts()
907      * @param parserDefine
908      */
909     function convertDefine(& $element)
910     {
911         parent::convertDefine($element, array('define_link' => $this-
> getId($element, false)));
912         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
913         $this-> addKLink($element-> name, $this-> page_dir . PATH_DELIMITER . $this-
> page, $this-> getId($element, false), 'Constants');
914     }
915
916     /**
917      * Converts global variables for template output
918      * @param parserGlobal
919      */
920     function convertGlobal(& $element)
921     {
922         parent::convertGlobal($element, array('global_link' => $this-
> getId($element, false)));
923         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
924         $this-> addKLink($element-> name, $this-> page_dir . PATH_DELIMITER . $this-
> page, $this-> getId($element, false), 'Global Variables');
925     }
926
927     /**
928      * converts procedural pages for template output
929      * @see prepareDocBlock(), getClassessOnPage()
930      * @param parserData
931      */
932     function convertPage(& $element)
933     {
934         parent::convertPage($element);
935         $this-> juststarted = true;
936         $this-> page_dir = $element-> parent-> package;
937         if (!empty($element-> parent-> subpackage)) $this-> page_dir .= PATH_DELIMITER .
$element-> parent-> subpackage;
938         // registering stuff on the template
939         $this-> page_data-> assign("page" , $this-> getPageName($element) .
'.html');
940         $this-> page_data-> assign("title" , "Docs for page " . $element-
> parent-> getFile());
941         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
942         $this-> addKLink($element-> parent-> file, $this-> page_dir . PATH_DELIMITER .
$this-> page, '' , 'Files');
943     }
944
945     function getPageName(& $element)

```

```

946     {
947         if (phpDocumentor_get_class($element) == 'parserpage') return '_'.$element-
> getName();
948         return '_'.$element-> parent-> getName();
949     }
950
951 /**
952 * returns an array containing the class inheritance tree from the root object to the class
953 *
954 * @param parserClass    class variable
955 * @return array Format: array(root,child,child,child,...,$class)
956 * @uses parserClass::getParentClassTree()
957 */
958
959 function generateFormattedClassTree($class)
960 {
961     $tree = $class-> getParentClassTree($this);
962     $out = '';
963     if (count($tree) - 1)
964     {
965         $result = array($class-> getName());
966         $parent = $tree[$class-> getName()];
967         $distance[] = '';
968         while ($parent)
969         {
970             $x = $parent;
971             if (is_object($parent))
972             {
973                 $subpackage = $parent-> docblock-> subpackage;
974                 $package = $parent-> docblock-> package;
975                 $x = $parent;
976                 $x = $parent-> getLink($this);
977                 if (!$x) $x = $parent-> getName();
978             }
979             $result[] =
980             $x;
981             $distance[] =
982             "\n$x|\n";
983             "%s--";
984             if (is_object($parent))
985                 $parent = $tree[$parent-> getName()];
986             elseif (isset($tree[$parent]))
987                 $parent = $tree[$parent];
988             }
989             $nbsp = ' ';
990             for($i=count($result) - 1;$i>= 0;$i--)
991             {
992                 $my_nbsp = '';
993                 for($j=0;$j< count($result) - $i;$j++) $my_nbsp .= $nbsp;
994                 $distance[$i] = sprintf($distance[$i],$my_nbsp,$my_nbsp);
995             }
996             return
array('classes'=> array_reverse($result), 'distance'=> array_reverse($distance));
997         } else
998         {
999             return array('classes'=> $class-> getName(), 'distance'=>array( '' ));
1000         }
1001     }
1002
1003 /**
1004 function sortVar($a, $b)
1005 {
1006     return strnatcasecmp($a-> getName(),$b-> getName());
1007 }
1008
1009 /**
1010 function sortMethod($a, $b)
1011 {
1012     if ($a-> isConstructor) return -1;
1013     if ($b-> isConstructor) return 1;
1014     return strnatcasecmp($a-> getName(),$b-> getName());
1015 }
1016
1017 /**
1018 * returns a template-enabled array of class trees
1019 *
1020 * @param string $package    package to generate a class tree for
1021 * @see $roots, HTMLConverter::getRootTree()
1022 */
1023 function generateFormattedClassTrees($package)

```

```

1024     {
1025         if (!isset($this-> roots['normal'][$package]) &&
1026             !isset($this-> roots['special'][$package])) {
1027             return array();
1028         }
1029         $trees = array();
1030         if (isset($this-> roots['normal'][$package])) {
1031             $roots = $this-> roots['normal'][$package];
1032             for($i=0;$i< count($roots);$i++)
1033             {
1034                 $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1035                 if ($root && $root-> isInterface()) {
1036                     continue;
1037                 }
1038                 $trees[] = array('class' => $roots[$i], 'class_tree' =>
1039                     . $this-> getRootTree($this-
1040                     > getSortedClassTreeFromClass($roots[$i],$package,''),$package). "</ul>\n"
1041                 );
1042             if (isset($this-> roots['special'][$package])) {
1043                 $roots = $this-> roots['special'][$package];
1044                 foreach ($roots as $parent => $classes) {
1045                     $thistree = '';
1046                     foreach ($classes as $classinfo) {
1047                         $root = $this-> classes-> getClassByPackage($classinfo, $package);
1048                         if ($root && $root-> isInterface()) {
1049                             continue;
1050                         }
1051                         $thistree .=
1052                             $this-> getRootTree(
1053                                 $this-> getSortedClassTreeFromClass(
1054                                     $classinfo,
1055                                     $package,
1056                                     ''),
1057                                     $package,
1058                                     true);
1059                     if (!$thistree) {
1060                         continue;
1061                     }
1062                     $trees[] = array(
1063                         'class' => $parent,
1064                         'class_tree' => "<ul>\n" . $thistree .
1065                     );
1066                 }
1067             }
1068             return $trees;
1069         }
1070     /**
1071      * returns a template-enabled array of interface inheritance trees
1072      *
1073      * @param string $package package to generate a class tree for
1074      * @see $roots, HTMLConverter::getRootTree()
1075      */
1076     function generateFormattedInterfaceTrees($package)
1077     {
1078         if (!isset($this-> roots['normal'][$package]) &&
1079             !isset($this-> roots['special'][$package])) {
1080             return array();
1081         }
1082         $trees = array();
1083         if (isset($this-> roots['normal'][$package])) {
1084             $roots = $this-> roots['normal'][$package];
1085             for($i=0;$i< count($roots);$i++)
1086             {
1087                 $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1088                 if ($root && !$root-> isInterface()) {
1089                     continue;
1090                 }
1091                 $trees[] = array('class' => $roots[$i], 'class_tree' =>
1092                     . $this-> getRootTree($this-
1093                     > getSortedClassTreeFromClass($roots[$i],$package,''),$package). "</ul>\n"
1094                 );
1095             if (isset($this-> roots['special'][$package])) {
1096                 $roots = $this-> roots['special'][$package];
1097                 foreach ($roots as $parent => $classes) {
1098                     $thistree = '';

```

```

1099     foreach ($classes as $classinfo) {
1100         $root = $this-> classes-> getClassByPackage($classinfo, $package);
1101         if ($root && ! $root-> isInterface()) {
1102             continue;
1103         }
1104         $thistree .=
1105             $this-> getRootTree(
1106                 $this-> getSortedClassTreeFromClass(
1107                     $classinfo,
1108                     $package,
1109                     ''),
1110                     $package,
1111                     true);
1112     }
1113     if (!$thistree) {
1114         continue;
1115     }
1116     $trees[] = array(
1117         'class' => $parent,
1118         'class_tree' => "<ul>\n"
1119             . $thistree .
1120             );
1121 }
1122 return $trees;
1123 }
1124 /**
1125 * return formatted class tree for the Class Trees page
1126 *
1127 * @param array $tree output from {@link getSortedClassTreeFromClass()}
1128 * @param string $package package
1129 * @param boolean $nounknownparent if true, an object's parent will not be checked
1130 * @see Classes:::$definitechild, generateFormattedClassTrees()
1131 * @return string
1132 */
1133 function getRootTree($tree, $package, $noparent = false)
1134 {
1135     if (!$tree) return '';
1136     $my_tree = '';
1137     $cur = '#root';
1138     $lastcur = array(false);
1139     $kids = array();
1140     $dopar = false;
1141     if (!$noparent && $tree[$cur]['parent'])
1142     {
1143         $dopar = true;
1144         if (!is_object($tree[$cur]['parent']))
1145         {
1146             // debug("parent ".$tree[$cur]['parent']." not found");
1147             $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['parent']);
1148         }
1149         else
1150         {
1151             // debug("parent ".$this-
1152             >returnSee($tree[$cur]['parent'])." in other package");
1153             $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['parent']);
1154             if ($tree[$cur]['parent']-> package != $package) $my_tree .= '
1155 <b>(Different package)</b><ul>';
1156         }
1157     do
1158     {
1159         // fancy_debug($cur,$lastcur,$kids);
1160         if (count($tree[$cur]['children']))
1161         {
1162             // debug("$cur has children");
1163             if (!isset($kids[$cur]))
1164             {
1165                 // debug("set $cur kids");
1166                 $kids[$cur] = 1;
1167                 $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link']);
1168                 $my_tree .= '<ul>' . "\n" ;
1169             }
1170             array_push($lastcur,$cur);
1171             list($cur) = each($tree[$cur]['children']);
1172             var_dump('listed',$cur);
1173             if ($cur)
1174             {
1175                 $cur = $cur['package'] . '#' . $cur['class'];

```

```

1176 // debug("set cur to child $cur");
1177 // $my_tree .= '<li>' . $this->returnSee($tree[$cur]['link']);
1178 continue;
1179 } else
1180 {
1181 // debug("end of children for $cur");
1182 $cur = array_pop($lastcur);
1183 $cur = array_pop($lastcur);
1184 $my_tree .= '</ul></li>' . "\n";
1185 if ($dopar && ($cur == '#root' || !$cur)) $my_tree .=
1186 '</ul></li>';
1187 }
1188 {
1189 // debug("$cur has no children");
1190 $my_tree .= '<li>' . $this-
1191 > returnSee($tree[$cur]['link']) . "</li>" ;
1192 if ($dopar && ($cur == '#root')) $my_tree .= '</ul></li>' ;
1193 $cur = array_pop($lastcur);
1194 } while ($cur);
1195 return $my_tree;
1196 }
1197 /**
1198 * Generate indexing information for given element
1199 *
1200 * @param parserElement descendant of parserElement
1201 * @see generateElementIndex()
1202 * @return array
1203 */
1204 function getIndexInformation($elt)
1205 {
1206     $Result['type'] = $elt-> type;
1207     $Result['file_name'] = $elt-> file;
1208     $Result['path'] = $elt-> getPath();
1209
1210     if (isset($elt-> docblock))
1211     {
1212         $Result['description'] = $elt-> docblock-> getSDesc($this);
1213
1214         if ($elt-> docblock-> hasaccess)
1215             $Result['access'] = $elt-> docblock-> tags['access'][0]-
1216 > value;
1217
1218         else
1219             $Result['abstract'] = isset ($elt-> docblock-
1220 > tags['abstract'][0]);
1221
1222         else
1223             $Result['description'] = '';
1224
1225         $aa = $Result['description'];
1226         if (!empty($aa)) $aa =
1227             "<br>&nbsp;&nbsp;&nbsp;" . $aa . "<br>&nbsp;&nbsp;&nbsp;";
1228
1229         switch($elt-> type)
1230         {
1231             case 'class':
1232                 $Result['name'] = $elt-> getName();
1233                 $Result['title'] = 'Class';
1234                 $Result['link'] = $this-> getClassLink($elt-> getName(),
1235
1236                                         $elt-> docblock-> package,
1237                                         $elt-> getPath(),
1238                                         $elt-> getName());
1239                 $Result['listing'] = 'in file ' . $elt-> file . ', class
1240
1241                 $aa" ;
1242                 break;
1243             case 'define':
1244                 $Result['name'] = $elt-> getName();
1245                 $Result['title'] = 'Constant';
1246                 $Result['link'] = $this-> getDefineLink($elt-> getName(),
1247
1248                                         $elt-> docblock-
1249 > package,
1250                                         $elt-> getPath(),
1251                                         $elt-> getName());
1252                 $Result['listing'] = 'in file ' . $elt-> file . ', constant
1253
1254                 $aa" ;
1255                 break;
1256             case 'global':

```

```

1248     $Result['name'] = $elt-> getName();
1249     $Result['title'] = 'Global';
1250     $Result['link'] = $this-> getGlobalLink($elt-> getName(),
1251                                              $elt-> docblock-
1252                                              $elt-> getPath(),
1253                                              $elt-> getName());
1254     $Result['listing'] = 'in file '.$elt-> file.', global variable
1255     $aa" ;
1256     break;
1257     case 'function':
1258         $Result['name'] = $elt-> getName();
1259         $Result['title'] = 'Function';
1260         $Result['link'] = $this-> getFunctionLink($elt-> getName(),
1261                                              $elt-> docblock-
1262                                              $elt-> getPath(),
1263                                              $elt-> getName().'()');
1264     $Result['listing'] = 'in file '.$elt-> file.', function
1265     $aa" ;
1266     break;
1267     case 'method':
1268         $Result['name'] = $elt-> getName();
1269         $Result['title'] = 'Method';
1270         $Result['link'] = $this-> getMethodLink($elt-> getName(),
1271                                              $elt-> class,
1272                                              $elt-> docblock-
1273                                              $elt-> getPath(),
1274                                              $elt-> class.'::'.$elt-
1275                                              );
1276         if ($elt-> isConstructor)
1277             $Result['listing'] = 'in file '.$elt-> file.', method
1278     $aa" ;
1279     break;
1280     case 'var':
1281         $Result['name'] = $elt-> getName();
1282         $Result['title'] = 'Variable';
1283         $Result['link'] = $this-> getVarLink($elt-> getName(),
1284                                              $elt-> class,
1285                                              $elt-> docblock-> package,
1286                                              $elt-> getPath(),
1287                                              $elt-> class.'::'.$elt-
1288             );
1289         $Result['listing'] = 'in file '.$elt-> file.', variable
1290     $aa" ;
1291     break;
1292     case 'const':
1293         $Result['name'] = $elt-> getName();
1294         $Result['title'] = 'Variable';
1295         $Result['link'] = $this-> getConstLink($elt-> getName(),
1296                                              $elt-> class,
1297                                              $elt-> docblock-> package,
1298                                              $elt-> getPath(),
1299                                              $elt-> class.'::'.$elt-
1300             );
1301         $Result['listing'] = 'in file '.$elt-> file.', class constant
1302     $aa" ;
1303     break;
1304     case 'page':
1305         $Result['name'] = $elt-> getFile();
1306         $Result['title'] = 'Page';
1307         $Result['link'] = $this-> getPageLink($elt-> getFile(),
1308                                              $elt-> package,
1309                                              $elt-> getPath(),
1310                                              $elt-> getFile());
1311         $Result['listing'] = 'procedural page '.$Result['link'];
1312     break;
1313     case 'include':
1314         $Result['name'] = $elt-> getName();
1315         $Result['title'] = 'Include';
1316         $Result['link'] = $elt-> getValue();
1317         $Result['listing'] = 'include '.$Result['name'];
1318     break;
1319 }
1320 return $Result;
1321 }

```

```

1316 /**
1317 * Generate alphabetical index of all elements
1318 *
1319 * @see $elements, walk()
1320 */
1321 function generateElementIndex()
1322 {
1323     $elementindex = array();
1324     $letters = array();
1325     $used = array();
1326     foreach($this-> elements as $letter =>    $nutoh)
1327     {
1328         foreach($this-> elements[$letter] as $i =>    $yuh)
1329         {
1330             if ($this-> elements[$letter][$i]-> type != 'include')
1331             {
1332                 if (!isset($used[$letter]))
1333                 {
1334                     $letters[]['letter'] = $letter;
1335                     $elindex['letter'] = $letter;
1336                     $used[$letter] = 1;
1337                 }
1338
1339                 $elindex['index'][] = $this-> getIndexInformation($this-
> elements[$letter][$i]);
1340             }
1341         }
1342         if (isset($elindex['index']))
1343         {
1344             $elementindex[] = $elindex;
1345         } else
1346         {
1347             unset($letters[count($letters) - 1]);
1348         }
1349         $elindex = array();
1350     }
1351     return array($elementindex,$letters);
1352 }
1353
1354 function setTemplateDir($dir)
1355 {
1356     Converter::setTemplateDir($dir);
1357     $this-> smarty_dir = $this-> templateDir;
1358 }
1359
1360 function copyMediaRecursively($media,$targetdir,$subdir = '')
1361 {
1362     $versionControlDirectories = array ('CVS', 'media/CVS', 'media\\CVS', '.svn',
1363     'media/.svn', 'media\\.svn');
1364     if (!is_array($media)) {
1365         return;
1366     }
1367     foreach($media as $dir =>    $files)
1368     {
1369         if ($dir === '/')
1370         {
1371             $this-> copyMediaRecursively($files,$targetdir);
1372         } else
1373         {
1374             if (!is_numeric($dir))
1375             {
1376                 if (in_array($dir, $versionControlDirectories))
1377                 {
1378                     // skip it entirely
1379                 } else
1380                 {
1381                     // create the subdir
1382                     phpDocumentor_out("      creating $targetdir" . PATH_DELIMITER .
1383                     "$dir\n" );
1384                     Converter::setTargetDir($targetdir . PATH_DELIMITER . $dir);
1385                     if (!empty($subdir))
1386                     {
1387                         $subdir .= PATH_DELIMITER;
1388                     }
1389                     $this-
1390                     > copyMediaRecursively($files,"      $targetdir/$dir" , $subdir . $dir);
1391                 }
1392             }
1393         }
1394     }
}

```

```

1392             {
1393                 // copy the file
1394                 phpDocumentor_out("      copying $targetdir" . PATH_DELIMITER .
1395                 );
1396                 $this-> copyFile($files['file'],$subdir);
1397             }
1398         }
1399     }
1400 }
1401 /**
1402 * calls the converter setTargetDir, and then copies any template images and the stylesheet
1403 if they haven't been copied
1404 * @see Converter::setTargetDir()
1405 */
1406 function setTargetDir($dir)
1407 {
1408     Converter::setTargetDir($dir);
1409     if ($this-> wrote) return;
1410     $this-> wrote = true;
1411     $template_images = array();
1412     $stylesheets = array();
1413     $dir = $dir;
1414     $dir = $this-> templateDir;
1415     $this-> templateDir = $this-> templateDir.'templates/';
1416     $info = new Io;
1417     $this-> copyMediaRecursively($info-> getDirTree($this-> templateDir.'media',$this-
1418 > templateDir),$dir);
1419 }
1420 /**
1421 * Generate alphabetical index of all elements by package and subpackage
1422 *
1423 * @param string $package name of a package
1424 * @see $pkg_elements, walk(), generatePkgElementIndexes()
1425 */
1426 function generatePkgElementIndex($package)
1427 {
1428     var_dump($this->pkg_elements[$package]);
1429     $elementindex = array();
1430     $letters = array();
1431     $letterind = array();
1432     $used = array();
1433     $subp = '';
1434     foreach($this-> pkg_elements[$package] as $subpackage => $els)
1435     {
1436         if (empty($els)) continue;
1437         if (!empty($subpackage)) $subp = "      <b>subpackage:</b>".
1438 $subpackage;
1439         ; else $subp = '';
1440         foreach($els as $letter => $yuh)
1441         {
1442             foreach($els[$letter] as $i => $yuh)
1443             {
1444                 if ($els[$letter][$i]-> type != 'include')
1445                 {
1446                     if (!isset($used[$letter]))
1447                     {
1448                         $letters[]['letter'] = $letter;
1449                         $letterind[$letter] = count($letters) - 1;
1450                         $used[$letter] = 1;
1451                         $selindex[$letter]['letter'] = $letter;
1452                         $selindex[$letter]['index'][] = $this-
1453 > getIndexInformation($els[$letter][$i]);
1454                     }
1455                 }
1456             }
1457             ksort($selindex);
1458             usort($letters,'CHMdefault_lettersort');
1459             if (isset($selindex))
1460             {
1461                 while(list($letter,$tempel) = each($selindex))
1462                 {
1463                     if (!isset($tempel))
1464                     {
1465                         unset($letters[$letterind[$tempel['letter']]]);
1466                     } else
1467                         $elementindex[] = $tempel;
1468             }
1469         }
1470     }
1471 }

```

```

1467         }
1468     } else $letters = array();
1469     return array($elementindex,$letters);
1470 }
1471 /**
1472 *
1473 * @see generatePkgElementIndex()
1474 */
1475 function generatePkgElementIndexes()
1476 {
1477     $packages = array();
1478     $package_names = array();
1479     $pkg = array();
1480     $letters = array();
1481     foreach($this-> pkg_elements as $package => $trash)
1482     {
1483         $pkgs['package'] = $package;
1484         $pkg['package'] = $package;
1485         list($pkg['pindex'],$letters[$package]) = $this-
1486 > generatePkgElementIndex($package);
1487         if (count($pkg['pindex']))
1488         {
1489             $packages[] = $pkg;
1490             $package_names[] = $pkgs;
1491         }
1492         unset($pkgs);
1493         unset($pkg);
1494     }
1495     foreach($packages as $i => $package)
1496     {
1497         $pnames = array();
1498         for($j=0;$j< count($package_names);$j++)
1499         {
1500             if ($package_names[$j]['package'] != $package['package']) $pnames[] =
1501             $package_names[$j];
1502         }
1503         $packages[$i]['packageindexes'] = $pnames;
1504     }
1505     return array($packages,$package_names,$letters);
1506 }
1507 /**
1508 * @param string name of class
1509 * @param string package name
1510 * @param string full path to look in (used in index generation)
1511 * @param boolean deprecated
1512 * @param boolean return just the URL, or enclose it in an html a tag
1513 * @return mixed false if not found, or an html a link to the class's documentation
1514 * @see parent::getClassLink()
1515 */
1516 function getClassLink($expr,$package, $file = false,$text = false, $with_a = true)
1517 {
1518     $a = Converter::getClassLink($expr,$package,$file);
1519     if (!$a) return false;
1520     return $this-> returnSee($a, $text, $with_a);
1521 }
1522 /**
1523 * @param string name of function
1524 * @param string package name
1525 * @param string full path to look in (used in index generation)
1526 * @param boolean deprecated
1527 * @param boolean return just the URL, or enclose it in an html a tag
1528 * @return mixed false if not found, or an html a link to the function's documentation
1529 * @see parent::getFunctionLink()
1530 */
1531 function getFunctionLink($expr,$package, $file = false,$text = false)
1532 {
1533     $a = Converter::getFunctionLink($expr,$package,$file);
1534     if (!$a) return false;
1535     return $this-> returnSee($a, $text);
1536 }
1537 /**
1538 * @param string name of define
1539 * @param string package name
1540 * @param string full path to look in (used in index generation)
1541 * @param boolean deprecated
1542 * @param boolean return just the URL, or enclose it in an html a tag
1543 */

```

```

1545 * @return mixed false if not found, or an html a link to the define's documentation
1546 * @see parent::getDefineLink()
1547 */
1548 function getDefineLink($expr,$package, $file = false,$text = false)
1549 {
1550     $a = Converter::getDefineLink($expr,$package,$file);
1551     if (!$a) return false;
1552     return $this->  returnSee($a, $text);
1553 }
1554 /**
1555 * @param string name of global variable
1556 * @param string package name
1557 * @param string full path to look in (used in index generation)
1558 * @param boolean deprecated
1559 * @param boolean return just the URL, or enclose it in an html a tag
1560 * @param mixed false if not found, or an html a link to the global variable's
1561 documentation
1562 * @see parent::getGlobalLink()
1563 */
1564 function getGlobalLink($expr,$package, $file = false,$text = false)
1565 {
1566     $a = Converter::getGlobalLink($expr,$package,$file);
1567     if (!$a) return false;
1568     return $this->  returnSee($a, $text);
1569 }
1570 /**
1571 * @param string name of procedural page
1572 * @param string package name
1573 * @param string full path to look in (used in index generation)
1574 * @param boolean deprecated
1575 * @param boolean return just the URL, or enclose it in an html a tag
1576 * @param mixed false if not found, or an html a link to the procedural page's
1577 documentation
1578 * @see parent::getPageLink()
1579 */
1580 function getPageLink($expr,$package, $path = false,$text = false)
1581 {
1582     $a = Converter::getPageLink($expr,$package,$path);
1583     if (!$a) return false;
1584     return $this->  returnSee($a, $text);
1585 }
1586 /**
1587 * @param string name of method
1588 * @param string class containing method
1589 * @param string package name
1590 * @param string full path to look in (used in index generation)
1591 * @param boolean deprecated
1592 * @param boolean return just the URL, or enclose it in an html a tag
1593 * @param mixed false if not found, or an html a link to the method's documentation
1594 * @see parent::getMethodLink()
1595 */
1596 function getMethodLink($expr,$class,$package, $file = false,$text = false)
1597 {
1598     $a = Converter::getMethodLink($expr,$class,$package,$file);
1599     if (!$a) return false;
1600     return $this->  returnSee($a, $text);
1601 }
1602 /**
1603 * @param string name of var
1604 * @param string class containing var
1605 * @param string package name
1606 * @param string full path to look in (used in index generation)
1607 * @param boolean deprecated
1608 * @param boolean return just the URL, or enclose it in an html a tag
1609 * @param mixed false if not found, or an html a link to the var's documentation
1610 * @see parent::getVarLink()
1611 */
1612 function getVarLink($expr,$class,$package, $file = false,$text = false)
1613 {
1614     $a = Converter::getVarLink($expr,$class,$package,$file);
1615     if (!$a) return false;
1616     return $this->  returnSee($a, $text);
1617 }
1618 /**
1619 * @param string name of class constant

```

```

1623 * @param string class containing class constant
1624 * @param string package name
1625 * @param string full path to look in (used in index generation)
1626 * @param boolean deprecated
1627 * @param boolean return just the URL, or enclose it in an html a tag
1628 * @return mixed false if not found, or an html a link to the var's documentation
1629 * @see parent::getVarLink()
1630 */
1631 function getConstLink($expr,$class,$package, $file = false,$text = false)
1632 {
1633     $a = Converter::getConstLink($expr,$class,$package,$file);
1634     if (!$a) return false;
1635     return $this->  returnSee($a, $text);
1636 }
1637 /**
1638 * does a nat case sort on the specified second level value of the array
1639 *
1640 * @param mixed $a
1641 * @param mixed $b
1642 * @return int
1643 */
1644 function rcNatCmp ($a, $b)
1645 {
1646     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1647     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1648
1649     return strnatcasecmp($aa, $bb);
1650 }
1651 /**
1652 * does a nat case sort on the specified second level value of the array.
1653 * this one puts constructors first
1654 *
1655 * @param mixed $a
1656 * @param mixed $b
1657 * @return int
1658 */
1659 function rcNatCmpl ($a, $b)
1660 {
1661     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1662     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1663
1664     if (strpos($aa,'CONSTRUCTOR') === 0)
1665     {
1666         return -1;
1667     }
1668     if (strpos($bb,'CONSTRUCTOR') === 0)
1669     {
1670         return 1;
1671     }
1672     if (strpos($aa,strtoupper($this->  class)) === 0)
1673     {
1674         return -1;
1675     }
1676     if (strpos($bb,strtoupper($this->  class)) === 0)
1677     {
1678         return -1;
1679     }
1680     return strnatcasecmp($aa, $bb);
1681 }
1682 /**
1683 * Write a file to disk, and add it to the {@link $hhp_files} list of files
1684 * to include in the generated CHM
1685 *
1686 * {@source}
1687 */
1688 function writefile($file,$contents)
1689 {
1690     $this->  addHHP($this->  targetDir . PATH_DELIMITER . $file);
1691     Converter::writefile($file,$contents);
1692 }
1693 /**
1694 * @uses $hhp_files creates the array by adding parameter $file
1695 */
1696 function addHHP($file)
1697 {
1698     $file = str_replace($this->  base_dir . PATH_DELIMITER, '', $file);
1699
1700 }
1701
1702

```

```

1703     $file = str_replace('\\\\', PATH_DELIMITER, $file);
1704     $file = str_replace('/', PATH_DELIMITER, $file);
1705     $file = str_replace(PATH_DELIMITER, '\\\\', $file);
1706     $this-> hhp_files[]['name'] = $file;
1707 }
1708
1709 function generateTOC()
1710 {
1711     $comppack = '';
1712     $templ = & $this-> newSmarty();
1713     foreach($this-> TOC as $package => $TOC1)
1714     {
1715         $comp_subs = '';
1716         $comp_subs1 = false;
1717         foreach($TOC1 as $subpackage => $types)
1718         {
1719             $comp_types = '';
1720             foreach($types as $type => $files)
1721             {
1722                 $comp = '';
1723                 $templ1 = & $this-> newSmarty();
1724                 $templ1-> assign('entry', array());
1725                 foreach($files as $file)
1726                 {
1727                     // use book icon for classes
1728                     if ($type == 'Classes') {
1729                         $templ1-> append('entry', array('paramname' => $file[0], 'outputfile' => $file[1], 'isclass' => 1));
1730                     } else {
1731                         $templ1-> append('entry', array('paramname' => $file[0], 'outputfile' => $file[1]));
1732                     }
1733                 }
1734                 $templ = & $this-> newSmarty();
1735                 $templ-> assign('tocsubentries', $templ1-> fetch('tocentry.tpl'));
1736                 $templ-> assign('entry', array(array('paramname' => $type)));
1737                 $comp_types .= $templ-> fetch('tocentry.tpl');
1738             }
1739             if (!empty($subpackage))
1740             {
1741                 $templ = & $this-> newSmarty();
1742                 $templ-> assign('tocsubentries', $comp_types);
1743                 $templ-> assign('entry', array(array('paramname' => $subpackage)));
1744                 $comp_subs .= $templ-> fetch('tocentry.tpl');
1745             } else
1746             {
1747                 $comp_subs1 = $comp_types;
1748             }
1749             if ($comp_subs1)
1750                 $templ-> assign('tocsubentries', $comp_subs1);
1751             if (!empty($comp_subs))
1752                 $templ-> assign('entry', array(array('paramname' => $package, 'tocsubentries' => $comp_subs)));
1753             else
1754                 $templ-> assign('entry', array(array('paramname' => $package)));
1755             $comppack .= $templ-> fetch('tocentry.tpl');
1756         }
1757     }
1758     return $comppack;
1759 }
1760
1761 function addSourceTOC($name, $file, $package, $subpackage, $source = false)
1762 {
1763     $file = str_replace($this-> base_dir . PATH_DELIMITER, '', $this-> targetDir)
1764     . PATH_DELIMITER . $file . '.html';
1765     $file = str_replace('\\\\', PATH_DELIMITER, $file);
1766     $file = str_replace('/', PATH_DELIMITER, $file);
1767     $file = str_replace(PATH_DELIMITER, '\\\\', $file);
1768     $sub = $source ? 'Source Code' : 'Examples';
1769     $this-> TOC[$package][$subpackage][$sub][] = array($name, $file);
1770 }
1771
1772 function addTOC($name, $file, $package, $subpackage, $class = false, $tutorial = false)
1773 {
1774     $file = str_replace($this-> base_dir . PATH_DELIMITER, '', $this-> targetDir . PATH_DELIMITER)
1775     . $file . '.html';
1776     $file = str_replace('\\\\', PATH_DELIMITER, $file);
1777     $file = str_replace('/', PATH_DELIMITER, $file);
1778     $file = str_replace(PATH_DELIMITER, '\\\\', $file);

```

```

1779     $file = str_replace($this-> base_dir . '\\', '', $file);
1780     $sub = $class ? 'Classes' : 'Files';
1781     if ($tutorial) $sub = 'Manual';
1782     $this-> TOC[$package][$subpackage][$sub][] = array($name,$file);
1783 }
1784 /**
1785 * Add an item to the index.hhk file
1786 * @param string $name index entry name
1787 * @param string $file filename containing index
1788 * @param string $bookmark html anchor of location in file, if any
1789 * @param string $group group this entry with a string
1790 * @uses $KLinks tracks the index
1791 * @author Andrew Eddie <eddieajau@users.sourceforge.net>
1792 */
1793 function addKLink($name, $file, $bookmark='', $group='')
1794 {
1795     $file = $file . '.html';
1796     $file = str_replace('\\',PATH_DELIMITER,$file);
1797     $file = str_replace('/',PATH_DELIMITER,$file);
1798     $file = str_replace(PATH_DELIMITER,'\\',$file);
1799     // debug("added $name, $file, $bookmark, $group ");
1800     $link = $file;
1801     $link .= $bookmark ? "#$bookmark" : '';
1802     if ($group) {
1803         $this-> KLinks[$group]['grouplink'] = $file;
1804         $this-> KLinks[$group][] = array($name,$link);
1805     }
1806     $this-> KLinks[] = array($name,$link);
1807 }
1808 /**
1809 * Get the table of contents for index.hhk
1810 * @return string contents of tocentry.tpl generated from $KLinks
1811 * @author Andrew Eddie <eddieajau@users.sourceforge.net>
1812 */
1813 function generateKLinks()
1814 {
1815     $stempl = & $this-> newSmarty();
1816     $stempl-> assign('entry', array());
1817     foreach($this-> KLinks as $group=> $link)
1818     {
1819         if (isset($link['grouplink'])) {
1820             $stemplg = & $this-> newSmarty();
1821             $stemplg-> assign('entry', array());
1822             foreach($link as $k=> $sublink)
1823             {
1824                 if ($k != 'grouplink') {
1825                     $stemplg-> append('entry', array('paramname' =>
1826 $sublink[0], 'outputfile' => $sublink[1]));
1827                 }
1828                 $stemplg-> append('entry', array('paramname' => $group, 'outputfile' =>
1829 $link['grouplink'], 'tocsubentries' => $stemplg-> fetch('tocentry.tpl')));
1830             } else {
1831                 $stemplg-> append('entry', array('paramname' => $link[0], 'outputfile' =>
1832 $link[1]));
1833             }
1834         }
1835         return $stempl-> fetch('tocentry.tpl');
1836     }
1837 /**
1838 * Create the phpdoc.hhp, contents.hhc files needed by MS HTML Help Compiler
1839 * to create a CHM file
1840 *
1841 * The output function generates the table of contents (contents.hhc)
1842 * and file list (phpdoc.hhp) files used to create a .CHM by the
1843 * free MS HTML Help compiler.
1844 * {@internal
1845 * Using {@link $hhp_files}, a list of all separate .html files
1846 * is created in CHM format, and written to phpdoc.hhp. This list was
1847 * generated by {@link writefile}.
1848 *
1849 * Next, a call to the table of contents:
1850 *
1851 * {@source 12 2}
1852 *
1853 * finishes things off}}
1854 * @todo use to directly call html help compiler hhc.exe

```

```

1856 * @link http://www.microsoft.com/downloads/release.asp?releaseid=33071
1857 * @uses generateTOC() assigns to the toc template variable
1858 */
1859 function Output()
1860 {
1861     $templ = & $this-> newSmarty();
1862     $templ-> assign('files',$this-> hhp_files);
1863     $this-> setTargetDir($this-> base_dir);
1864     Converter::writefile('phpdoc.hhp',$templ-> fetch('hhp.tpl'));
1865     $templ = & $this-> newSmarty();
1866     $templ-> assign('toc',$this-> generateTOC());
1867     Converter::writefile('contents.hhc',$templ-> fetch('contents.hhc.tpl'));
1868     $templ-> assign('klinks',$this-> generateKLinks());
1869     Converter::writefile('index.hhk',$templ-> fetch('index.hhk.tpl'));
1870     phpDocumentor_out("NOTE: to create the documentation.chm file, you must now run
Microsoft Help Workshop on phpdoc.hhp\n");
1871     phpDocumentor_out("To get the free Microsoft Help Workshop, browse to:
http://go.microsoft.com/fwlink/?LinkId=14188\n");
1872     flush();
1873 }
1874 }
1875 /**
1876 * @access private
1877 * @global string name of the package to set as the first package
1878 */
1880 function CHMdefault_pindexcmp($a, $b)
1881 {
1882     global $phpDocumentor_DefaultPackageName;
1883     if ($a['title'] == $phpDocumentor_DefaultPackageName) return -1;
1884     if ($b['title'] == $phpDocumentor_DefaultPackageName) return 1;
1885     return strnatcasecmp($a['title'],$b['title']);
1886 }
1887 /**
1888 * @access private */
1889 function CHMdefault_lettersort($a, $b)
1890 {
1891     return strnatcasecmp($a['letter'],$b['letter']);
1892 }
1893 /**
1894 * @access private */
1895 function CHMdefault_outputfilter($src, & $smarty)
1896 {
1897     return str_replace('{subdir}',$smarty-> _tpl_vars['subdir'],$src);
1898 }
1899 ?>

```

# File Source for HTMLframesConverter.inc

Documentation for this file is available at [HTMLframesConverter.inc](#)

```
1      <?php
2      /**
3      * HTML original framed output converter, modified to use Smarty Template.
4      * This Converter takes output from the {@link Parser} and converts it to HTML-ready output for
use with {@link Smarty}.
5      *
6      * phpDocumentor :: automatic documentation generator
7      *
8      * PHP versions 4 and 5
9      *
10     * Copyright (c) 2002-2006 Gregory Beaver
11     *
12     * LICENSE:
13     *
14     * This library is free software; you can redistribute it
15     * and/or modify it under the terms of the GNU Lesser General
16     * Public License as published by the Free Software Foundation;
17     * either version 2.1 of the License, or (at your option) any
18     * later version.
19     *
20     * This library is distributed in the hope that it will be useful,
21     * but WITHOUT ANY WARRANTY; without even the implied warranty of
22     * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23     * Lesser General Public License for more details.
24     *
25     * You should have received a copy of the GNU Lesser General Public
26     * License along with this library; if not, write to the Free Software
27     * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28     *
29     * @package Converters
30     * @subpackage HTMLframes
31     * @author Gregory Beaver <cellog@php.net>
32     * @copyright 2002-2006 Gregory Beaver
33     * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34     * @version CVS: $Id: HTMLframesConverter.inc 234145 2007-04-19 20:20:57Z ashnazg $
35     * @filesource
36     * @link http://www.phpdoc.org
37     * @link http://pear.php.net/PhpDocumentor
38     * @see parserDocBlock, parserInclude, parserPage, parserClass
39     * @see parserDefine, parserFunction, parserMethod, parserVar
40     * @since 1.2
41     */
42 /**
43     * HTML output converter.
44     * This Converter takes output from the {@link Parser} and converts it to HTML-ready output for
use with {@link Smarty}.
45     *
46     * @package Converters
47     * @subpackage HTMLframes
48     * @see parserDocBlock, parserInclude, parserPage, parserClass, parserDefine, parserFunction,
parserMethod, parserVar
49     * @author Greg Beaver <cellog@php.net>
50     * @since 1.2
51     * @version $Id: HTMLframesConverter.inc 234145 2007-04-19 20:20:57Z ashnazg $
52     */
53 class HTMLframesConverter extends Converter
54 {
55     /**
56     * This converter knows about the new root tree processing
57     * In order to fix PEAR Bug #6389
58     * @var boolean
59     */
60     var $processSpecialRoots = true;
61 /**
62     * Smarty Converter wants elements sorted by type as well as alphabetically
63     * @see Converter::$sort_page_contents_by_type
64     * @var boolean
```

```

65      */
66  var $sort_page_contents_by_type = true;
67  /** @var string */
68  var $outputformat = 'HTML';
69  /** @var string */
70  var $name = 'frames';
71  /**
72   * indexes of elements of package that need to be generated
73   * @var array
74   */
75  var $leftindex = array('classes' => true, 'pages' => true, 'functions' => true,
76 'defines' => false, 'globals' => false);
77 /**
78  * output directory for the current procedural page being processed
79  * @var string
80  */
81  var $page_dir;
82 /**
83  * target directory passed on the command-line.
84  * {@link $targetDir} is malleable, always adding package/ and package/subpackage/
85  subdirectories onto it.
86  * @var string
87  */
88  var $base_dir;
89 /**
90  * output directory for the current class being processed
91  * @var string
92  */
93  var $class_dir;
94 /**
95  * array of converted package page names.
96  * Used to link to the package page in the left index
97  * @var array Format: array(package => 1)
98  */
99  var $package_pages = array();
100 /**
101  * controls formatting of parser informative output
102  *
103  * Converter prints:
104  * "Converting /path/to/file.php... Procedural Page Elements... Classes..."
105  * Since HTMLdefaultConverter outputs files while converting, it needs to send a \n to
106  start a new line. However, if there
107  * is more than one class, output is messy, with multiple \n's just between class file
108  output. This variable prevents that
109  * and is purely cosmetic
110  * @var boolean
111  */
112  var $juststarted = false;
113 /**
114  * contains all of the template procedural page element loop data needed for the current
115  template
116  * @var array
117  */
118  var $current;
119 /**
120  * contains all of the template class element loop data needed for the current template
121  * @var array
122  */
123  var $currentclass;
124  var $wrote = false;
125  var $ric_set = array();
126 /**
127  * sets {@link $base_dir} to $targetDir
128  * @see Converter()
129  */
130  function HTMLframesConverter(& $allp, & $packp, & $classes, & $procpages, $po,
131  $pp, $qm, $targetDir, $templateDir, $title)
132  {
133      Converter::Converter($allp, $packp, $classes, $procpages, $po, $pp, $qm, $targetDir,
134      $templateDir, $title);
135      $this-> base_dir = $targetDir;
136  }

```

```

138
139 /**
140 * @deprecated in favor of PHP 4.3.0+ tokenizer-based source highlighting
141 */
142 function unmangle($sourcecode)
143 {
144     $sourcecode = str_replace('<code>', '<pre>', $sourcecode);
145     $sourcecode = str_replace('</code>', '</pre>', $sourcecode);
146     $sourcecode = str_replace('<br />', '\n', $sourcecode);
147     $sourcecode = str_replace('&nbsp;', ' ', $sourcecode);
148     $sourcecode = str_replace('&lt;', '<', $sourcecode);
149     $sourcecode = str_replace('&gt;', '>', $sourcecode);
150     $sourcecode = str_replace('&amp;', '&', $sourcecode);
151     return $sourcecode;
152 }
153
154 /**
155 * @param string full path to the source file
156 * @param string fully highlighted source code
157 */
158 function writeSource($path, $value)
159 {
160     $templ = & $this-> newSmarty();
161     $pathinfo = $this-> proceduralpages-> getPathInfo($path, $this);
162     $templ-> assign('source', $value);
163     $templ-> assign('package', $pathinfo['package']);
164     $templ-> assign('subpackage', $pathinfo['subpackage']);
165     $templ-> assign('name', $pathinfo['name']);
166     $templ-> assign('source_loc', $pathinfo['source_loc']);
167     $templ-> assign('docs', $pathinfo['docs']);
168     $templ-> assign("subdir", '../');
169     $templ-> register_outputfilter('HTMLframes_outputfilter');
170     $this-> setTargetDir($this-> getFileSourcePath($this-> base_dir));
171     phpDocumentor_out("\n");
172     $this-> setSourcePaths($path);
173     $this-> writefile($this-> getFileSourceName($path) . '.html', $templ-
> fetch('filesource.tpl'));
174 }
175
176 function writeExample($title, $path, $source)
177 {
178     $templ = & $this-> newSmarty();
179     $templ-> assign('source', $source);
180     if (empty($title))
181     {
182         $title = 'example';
183         addWarning(PDERROR_EMPTY_EXAMPLE_TITLE, $path, $title);
184     }
185     $templ-> assign('title', $title);
186     $templ-> assign('file', $path);
187     $templ-> assign("subdir", '../');
188     $templ-> register_outputfilter('HTMLframes_outputfilter');
189     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . '__examplesource');
190     phpDocumentor_out("\n");
191     $this-> writefile('exsource_' . $path . '.html', $templ-> fetch('examplesource.tpl'));
192 }
193
194 function getExampleLink($path, $title)
195 {
196     return $this-> returnLink('{$subdir}__examplesource' . PATH_DELIMITER .
'__exsource_' . $path . '.html', $title);
197 }
198
199 function getSourceLink($path)
200 {
201     return $this-> returnLink('{$subdir}__filesource/' .
$this-> getFileSourceName($path) . '.html', 'Source Code for this file');
202 }
203
204 /**
205 * Retrieve a Converter-specific anchor to a segment of a source code file
206 * parsed via a {@tutorial tags.filesource.pkg} tag.
207 * @param string full path to source file
208 * @param string name of anchor
209 * @param string link text, if this is a link
210 * @param boolean returns either a link or a destination based on this
211 * parameter
212 * @return string link to an anchor, or the anchor
213 */
214 function getSourceAnchor($sourcefile, $anchor, $text = '', $link = false)

```

```

216     {
217         if ($link) {
218             return $this-> _returnLink('{$subdir}__filesource/' .
219                                         $this-> getFileSourceName($sourcefile) . '.html#a' . $anchor, $text);
220         } else {
221             return '<a name="a" href="' . $anchor . '"></a>';
222         }
223     }
224
225     /**
226      * Return a line of highlighted source code with formatted line number
227      *
228      * If the $path is a full path, then an anchor to the line number will be
229      * added as well
230      * @param integer line number
231      * @param string highlighted source code line
232      * @param false/string full path to @filesource file this line is a part of,
233      *                      if this is a single line from a complete file.
234      * @return string formatted source code line with line number
235      */
236     function sourceLine($linenumber, $line, $path = false)
237     {
238         $extra = '';
239         if (strlen(str_replace("\n", "\n", $line)) == 0) {
240             $extra = ' ';
241         }
242         if ($path)
243         {
244             return '<li><div class="src-line">' . $this-
245             > getSourceAnchor($path, $linenumber) . str_replace("\n", "\n", $line) . $extra .
246             "</div></li>\n";
247         } else
248         {
249             return '<li><div class="src-line">' .
250             str_replace("\n", "\n", $line) .
251             " " . $extra . '</div></li>\n" ' ;
252         }
253     }
254     /**
255      * Used to convert the <<code>> tag in a docblock
256      * @param string
257      * @param boolean
258      * @return string
259      */
260     function ProgramExample($example, $tutorial = false, $inlinesourceparse = null/*false*/,
261                             $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
262                             null/*false*/)
263     {
264         return '<div class="src-code"><ol>' .
265             parent::ProgramExample($example, $tutorial, $inlinesourceparse, $class, $linenum, $filesourcepath) .
266             '</ol></div>' ;
267     }
268     /**
269      * @param string
270      */
271     function TutorialExample($example)
272     {
273         $trans = $this-> template_options['desctranslate'];
274         $this-> template_options['desctranslate'] = array();
275         $example = '<ol>' . parent::TutorialExample($example) .
276             '</ol>' ;
277         $this-> template_options['desctranslate'] = $trans;
278         if (!isset($this-> template_options['desctranslate'])) return $example;
279         if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
280         $example = $this-> template_options['desctranslate'][['code']] . $example;
281         if (!isset($this-> template_options['desctranslate'][[['code']]])) return $example;
282         return $example . $this-> template_options['desctranslate'][[['code']]);
283     }
284     function getCurrentPageLink()
285     {
286         return $this-> curname . '.html';
287     }
288
289     /**
290      * Uses htmlspecialchars() on the input
291      */

```

```

292     function postProcess($text)
293     {
294         if ($this-> highlightingSource) {
295             return str_replace(array(' ', "\t"), array(' ', ' ', 
296 'nbsp;&nbsp;'), htmlspecialchars($text));
297         }
298         return htmlspecialchars($text);
299     }
300
301 /**
302 * Use the template tutorial_toc.tpl to generate a table of contents for HTML
303 * @return string table of contents formatted for use in the current output format
304 * @param array format: array(array('tagname' => section, 'link' => returnsee link,
305 'id' => anchor name, 'title' => from title tag),...)
306 */
307     function formatTutorialTOC($toc)
308     {
309         $template = & $this-> newSmarty();
310         $template-> assign('toc', $toc);
311         return $template-> fetch('tutorial_toc.tpl');
312     }
313
314     function & SmartyInit(& $templ)
315     {
316         if (!isset($this-> package_index))
317             foreach($this-> all_packages as $key => $val)
318             {
319                 if (isset($this-> pkg_elements[$key]))
320                 {
321                     if (!isset($start)) $start = $key;
322                     $this-> package_index[] = array('link' => "li_$key.html"
323 'title' => $key);
324                 }
325                 $templ-> assign("packageindex" , $this-> package_index);
326                 $templ-> assign("subdir" , '');
327                 return $templ;
328             }
329 /**
330 * Writes out the template file of {@link $class_data} and unsets the template to save
331 * memory
332 * @see registerCurrentClass()
333 * @see parent::endClass()
334 */
335     function endClass()
336     {
337         $a = '../';
338         if (!empty($this-> subpackage)) $a .= '../';
339         if ($this-> juststarted)
340         {
341             $this-> juststarted = false;
342             phpDocumentor_out("\n");
343             flush();
344         }
345         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> class_dir);
346         $this-> class_data-> assign("subdir" , $a);
347         $this-> class_data-> register_outputfilter('HTMLframes_outputfilter');
348         $this-> writefile($this-> class . '.html', $this-> class_data-
> fetch('class.tpl'));
349         unset($this-> class_data);
350     }
351 /**
352 * Writes out the template file of {@link $page_data} and unsets the template to save memory
353 * @see registerCurrent()
354 * @see parent::endPage()
355 */
356     function endPage()
357     {
358         $this-> package = $this-> curpage-> package;
359         $this-> subpackage = $this-> curpage-> subpackage;
360         $a = '../';
361         if (!empty($this-> subpackage)) $a .= '../';
362         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
363         $this-> page_data-> assign("package" , $this-> package);
364         $this-> page_data-> assign("subdir" , $a);
365         $this-> page_data-> register_outputfilter('HTMLframes_outputfilter');
366         $this-> writefile($this-> page . '.html', $this-> page_data-> fetch('page.tpl'));

```

```

367     unset($this-> page_data);
368 }
369
370 /**
371 * @param string
372 * @param string
373 * @return string <a href='".$link."'>'.$text.'</a>';
374 */
375 function returnLink($link,$text)
376 {
377     return '<a href="'.$link.'">'.$text.'</a>';
378 }
379
380 function makeLeft()
381 {
382     foreach($this-> page_elements as $package => $o1)
383     {
384         foreach($o1 as $subpackage => $links)
385         {
386             for($i=0;$i< count($links);$i++)
387             {
388                 $left[$package][$subpackage]['files'][] =
389                     array("link" => $this-> getId($links[$i]),
390 "title" => $links[$i]-> name);
391             }
392         }
393         $interfaces = $classes = false;
394         foreach($this-> class_elements as $package => $o1)
395         {
396             foreach($o1 as $subpackage => $links)
397             {
398                 for($i=0;$i< count($links);$i++)
399                 {
400                     $class = $this-> classes-> getClassByPackage($links[$i]-> name,
$links[$i]-> package);
401                     $isinterface = $isclass = false;
402                     if ($class-> isInterface())
403                         {
404                             $isinterface = true;
405                             $interfaces = true;
406                         } else {
407                             $isclass = true;
408                             $classes = true;
409                         }
410                     if ($class && isset( $class-> docblock) &&
$class-> docblock-> hasaccess) {
411                         $left[$package][$subpackage]['classes'][] =
412                             array("link" => $this-> getId($links[$i]),
413 "title" => $links[$i]-> name,
414 'is_interface' => $isinterface,
415 'is_class' => $isclass,
416 "access" => $class-> docblock-> tags['access'][0]-> value,
417 "abstract" => isset( $class-> docblock-> tags['abstract'][0]));
418                     } else {
419                         $left[$package][$subpackage]['classes'][] =
420                             array("link" => $this-> getId($links[$i]),
421 "title" => $links[$i]-> name,
422 'is_interface' => $isinterface,
423 'is_class' => $isclass,
424 "access" => 'public',
425 "abstract" => isset( $class-> docblock-> tags['abstract'][0]));
426                     }
427                 }
428             }
429         foreach($this-> function_elements as $package => $o1)
430         {
431             foreach($o1 as $subpackage => $links)
432             {
433                 for($i=0;$i< count($links);$i++)
434                 {
435                     $left[$package][$subpackage]['functions'][] =
436                         array("link" => $this-> getId($links[$i]),
437 "title" => $links[$i]-> name);
438                 }
439             }
}

```

```

440     $ret = array();
441     foreach($left as $package =>      $r)
442     {
443         $pd = 'blank';
444         if (isset($this-> package_pages[$package])) $pd =
$package.'/package_'. $package.'.html';
445         if (!isset($r['']))
446         {
447             $pt = false;
448             $ptnoa = false;
449             $ptt = $package;
450             if ($t = $this-> hasTutorial('pkg', $package, $package, ''))
451             {
452                 $pt = $t-> getLink($this);
453                 $ptnoa = $this-> getId($t-> getLink($this, true));
454                 $ptt = $t-> getTitle($this);
455             }
456             $stutes = array();
457             foreach($this-> tutorial_tree as $root =>      $tr)
458             {
459                 if ($tr['tutorial']-> package == $package &&      $tr['tutorial']-
> subpackage == '') {
460                     $stutes[$tr['tutorial']-> tutorial_type][] =
$tr-> getTutorialTree($tr['tutorial']);
461                 }
462             }
463             if (isset($this-> childless_tutorials[$package][$subpackage]))
464             {
465                 foreach($this-> childless_tutorials[$package][$subpackage] as $ext =>
$other)
466                 {
467                     foreach($other as $tutorial)
468                     {
469                         $stutes[$tutorial-> tutorial_type][] = $this-
> getTutorialTree($tutorial);
470                     }
471                 }
472             }
473         }
474         $ret[$package][] =
475             array(
476                 'package' =>      $package,
477                 'subpackage' =>      '',
478                 'packagedoc' =>      $pd,
479                 'packagetutorial' =>      $pt,
480                 'packagetutorialnoa' =>      $ptnoa,
481                 'packagetutorialtitle' =>      $ptt,
482                 'files' => array(),
483                 'functions' => array(),
484                 'classes' => array(),
485                 'tutorials' =>      $stutes,
486             );
487     }
488     foreach($r as $subpackage =>      $info)
489     {
490         $my = array();
491         $my['package'] = $package;
492         if (isset($this-> package_pages[$package]))
493             $my['packagedoc'] = $pd;
494         else
495             $my['packagedoc'] = 'blank';
496         $my['subpackage'] = $subpackage;
497         if (empty($subpackage))
498         {
499             if ($t = $this-> hasTutorial('pkg', $package, $package, $subpackage))
500             {
501                 $my['packagetutorial'] = $t-> getLink($this);
502                 $my['packagetutorialnoa'] = $this-> getId($t-> getLink($this, true));
503                 $my['packagetutorialtitle'] = $t-> getTitle($this);
504             } else
505             {
506                 $my['packagetutorial'] = '<a href="blank.html">No
Package-Level Tutorial</a>';
507                 ;
508                 $my['packagetutorialnoa'] = 'blank.html';
509                 $my['packagetutorialtitle'] = $package;
510             }
511         }
512         if ($t = $this-> hasTutorial('pkg', $subpackage, $package, $subpackage))
513         {
514             $my['subpackagetutorial'] = $this-> returnSee($this-

```

```

>     getTutorialLink($t));
515     $my['subpackagetutorialnoa'] = $this-> getId($t-
>     getLink($this,true));
516     $my['subpackagetutorialtitle'] = $t-> getTitle($this);
517     } else
518     {
519         $my['subpackagetutorial'] = false;
520         $my['subpackagetutorialnoa'] = false;
521         $my['subpackagetutorialtitle'] = $subpackage;
522     }
523     $stutes = array();
524     foreach($this-> tutorial_tree as $root => $str)
525     {
526         if ($str['tutorial']-> package == $package && $str['tutorial']-
>     subpackage == $subpackage)
527         {
528             $stutes[$str['tutorial']]-> tutorial_type[] = $this-
>     getTutorialTree($str['tutorial']);
529         }
530     }
531     $my['tutorials'] = $stutes;
532     $my['files'] = $my['classes'] = $my['functions'] = array();
533     if (isset($info['files']))
534     {
535         $my['files'] = $info['files'];
536         if (isset($info['classes']))
537         {
538             $my['hasclasses'] = $classes;
539             $my['hasinterfaces'] = $interfaces;
540             if (isset($info['functions']))
541             {
542                 $my['functions'] = $info['functions'];
543                 $ret[$package][] = $my;
544             }
545         }
546     }
547     return $ret;
548 }
549 function getTutorialTree($tutorial,$k = false)
550 {
551     $ret = '';
552     if (is_object($tutorial)) $tree = parent::getTutorialTree($tutorial); else $tree =
553     Tutorial;
554     // debug($this->vardump_tree($tree));exit;
555     if (!$tree)
556     {
557         $template = & $this-> newSmarty();
558         $template-> assign('subtree',false);
559         $template-> assign('name',str_replace('.','',$tutorial-> name));
560         $template-> assign('parent',false);
561         $template-> assign('haskids',false);
562         $template-> assign('kids','');
563         $link = new tutorialLink;
564         $t = $tutorial;
565         $link-> addLink('',$t-> path,$t-> name,$t-> package,$t-> subpackage,$t-
>     getTitle($this));
566         $main = array('link' => $this-> getId($link), 'title' => $link-> title);
567         $template-> assign('main',$main);
568         return $template-> fetch('tutorial_tree.tpl');
569     }
570     if (isset($tree['kids']))
571     {
572         foreach($tree['kids'] as $subtree)
573         {
574             $ret .= $this-> getTutorialTree($subtree, true);
575         }
576         $template = & $this-> newSmarty();
577         $template-> assign('subtree',$k);
578         $template-> assign('name',str_replace('.','',$tree['tutorial']-> name));
579         $template-> assign('parent',($k ? str_replace('.','',$tree['tutorial']-> parent-
>     name) : false));
580         $template-> assign('haskids',strlen($ret));
581         $template-> assign('kids',$ret);
582         $link = new tutorialLink;
583         $t = $tree['tutorial'];
584         $link-> addLink('',$t-> path,$t-> name,$t-> package,$t-> subpackage,$t-
>     getTitle($this));
585         $main = array('link' => $this-> getId($link), 'title' => $link-> title);
586         $ret = $template-> fetch('tutorial_tree.tpl');

```

```

587             return $ret;
588         }
589
590         /**
591          * HTMLDefaultConverter chooses to format both package indexes and the complete index here
592          *
593          * This function formats output for the elementindex.html and pkgelementindex.html template
594          * files. It then
595          * writes them to the target directory
596          * @see generateElementIndex(), generatePkgElementIndex()
597         */
598         function formatPkgIndex()
599     {
600         list($package_indexes,$packages,$mletters) = $this-> generatePkgElementIndexes();
601         for($i=0;$i< count($package_indexes);$i++)
602     {
603             $template = & $this-> newSmarty();
604             $this-> package = $package_indexes[$i]['package'];
605             $this-> subpackage = '';
606             $template-> assign("index" ,,$package_indexes[$i]['pindex']);
607             $template-> assign("package" ,,$package_indexes[$i]['package']);
608             $template-
609             > assign("letters" ,,$mletters[$package_indexes[$i]['package']]);
610             $template-> register_outputfilter('HTMLframes_outputfilter');
611             $this-> setTargetDir($this-> base_dir);
612             $this-
613             > writefile('elementindex_'.$package_indexes[$i]['package'].'.html',$template-
614             > fetch('pkgelementindex.tpl'));
615             }
616             /**
617              * HTMLDefaultConverter uses this function to format template index.html and packages.html
618              *
619              * This function generates the package list from {@link $all_packages}, eliminating any
620              * packages that don't have any entries in their package index (no files at all, due to
621              * @ignore
622              * or other factors). Then it uses the default package name as the first package index to
623              * display.
624              * It sets the right pane to be either a blank file with instructions on making package-
625              * level docs,
626              * or the package-level docs for the default package.
627              * @global string Used to set the starting package to display
628             */
629             function formatIndex()
630         {
631             global $phpDocumentor_DefaultPackageName;
632             list($elindex,$mletters) = $this-> generateElementIndex();
633             $template = & $this-> newSmarty();
634             $template-> assign("index" ,,$elindex);
635             $template-> assign("letters" ,,$mletters);
636             $template-> register_outputfilter('HTMLframes_outputfilter');
637             $phpDocumentor_out("\n");
638             $this-> setTargetDir($this-> base_dir);
639             $this-> writefile('elementindex.html',$template-> fetch('elementindex.tpl'));
640             usort($this-> package_index,"HTMLframes_pindexcmp");
641             $index = & $this-> newSmarty();
642             foreach($this-> all_packages as $key => $val)
643             {
644                 if (isset($this-> pkg_elements[$key]))
645                 {
646                     if (!isset($start)) $start = $key;
647                     if (!isset($this-> package_pages[$key])) $this-> writeNewPPage($key);
648                 }
649                 // Created index.html
650                 if (isset($this-> pkg_elements[$phpDocumentor_DefaultPackageName])) $start =
651                 $phpDocumentor_DefaultPackageName;
652                 $this-> package = $start;
653                 $this-> subpackage = '';
654                 $index-> assign("package_count" ,count($this-> pkg_elements));
655                 if (count($this-> ric_set))
656                 $index-> assign("package_count" ,2);
657                 $index-> assign("date" ,date("r" ,time()));
658                 $index-> assign("title" ,$this-> title);
659                 $index-> assign("start" ,"- $start.html");
660                 $index-> register_outputfilter('HTMLframes_outputfilter');

```

```

659     if (isset($this-> tutorials[$start][''][$pkg][$start . '.pkg']))
660     {
661         $index-> assign("blank" , $start . '/tutorial_' . $start . '.pkg');
662     } elseif (isset($this-> package_pages[$start]))
663     {
664         $index-> assign("blank" , $start . '/package_' . $start);
665     }
666     else
667     {
668         $index-> assign("blank" , "blank");
669         $blank = & $this-> newSmarty();
670         $blank-> assign('package' , $this-> package);
671         $this-> setTargetDir($this-> base_dir);
672         $this-> writefile("blank.html" , $blank-> fetch('blank.tpl'));
673     }
674     phpDocumentor_out("\n");
675     flush();
676     $this-> setTargetDir($this-> base_dir);
677     $this-> writefile("index.html" , $index-> fetch('index.tpl'));
678
679     // Create package index
680     $package = & $this-> newSmarty();
681     $package-> assign('ric' , array());
682     if (isset($this-> ric_set))
683     {
684         foreach($this-> ric_set as $name => $u)
685         {
686             $package-> append('ric' , array('file' => 'ric_' . $name . '.html' , 'name' =>
687 $name));
688         }
689     }
690     $package-> assign("packages" , $this-> package_index);
691     $package-> register_outputfilter('HTMLframes_outputfilter');
692     $this-> writefile("packages.html" , $package-> fetch('top_frame.tpl'));
693     unset($index);
694 }
695 function writeNewPPage($key)
696 {
697     return;
698     $template = & $this-> newSmarty();
699     $this-> package = $key;
700     $this-> subpackage = '';
701     $template-> assign("date" , date("r" , time()));
702     $template-> assign("title" , $this-> title);
703     $template-> assign("package" , $key);
704     $template-> register_outputfilter('HTMLframes_outputfilter');
705     phpDocumentor_out("\n");
706     flush();
707     $this-> setTargetDir($this-> base_dir);
708
709     $this-> writefile("li_" . $key . ".html" , $template-> fetch('index.tpl'));
710     unset($template);
711 }
712 /**
713 * Generate indexes for li_package.html and classtree output files
714 *
715 * This function generates the li_package.html files from the template file left.html. It
716 does this by
717 * iterating through each of the $page_elements, $class_elements and $function_elements
718 arrays to retrieve
719 * the pre-sorted {@link abstractLink} descendants needed for index generation. Conversion
720 of these links to
721 * text is done by {@link returnSee()}. The {@link $local} parameter is set to false to
722 ensure that paths are correct.
723 *
724 * Then it uses {@link generateFormattedClassTrees()} to create class trees from the
725 template file classtrees.html. Output
726 * filename is classtrees_packagename.html. This function also unsets {@link $elements}
727 and {@link $pkg_elements} to free
728 * up the considerable memory these two class vars use
729 * @see $page_elements, $class_elements, $function_elements
730 */
731 function formatLeftIndex()
732 {
733     phpDocumentor_out("\n");
734     flush();
735     $this-> setTargetDir($this-> base_dir);
736     if (0) //!isset($this->left))

```

```

732
733     {
734         debug("Nothing parsed, check the command-line");
735         die();
736     }
737     $x = $this-> makeLeft();
738     foreach($this-> all_packages as $package => $rest)
739     {
740         if (!isset($this-> pkg_elements[$package])) continue;
741         $template = & $this-> newSmarty();
742         $template-> assign("info", $x[$package]);
743         $template-> assign('package', $package);
744         $template-> assign("hastutorials", isset($this-> tutorials[$package]));
745         $template-> assign("hastodos", count($this-> todoList));
746         $template-> assign("todolink", 'todolist.html');
747         $template-> assign("classtreepage", "classtrees_$package");
748         $template-> assign("elementindex", "elementindex_$package");
749         $template-> register_outputfilter('HTMLframes_outputfilter');
750         if (isset($this-> package_pages[$package]))
751         {
752             $template-> assign("packagedoc", $package.'/package_'.$package.'.html');
753         }
754         else
755         {
756             $template-> assign("packagedoc", false);
757         }
758         $this-> writefile("li_$package.html", $template-
> fetch('left_frame.tpl'));
759         // Create class tree page
760         $template = & $this-> newSmarty();
761         $template-> assign("classtrees", $this-
> generateFormattedClassTrees($package));
762         $template-> assign("interfaces", $this-
> generateFormattedInterfaceTrees($package));
763         $template-> assign("package", $package);
764         $template-> register_outputfilter('HTMLframes_outputfilter');
765         $this-> writefile("classtrees_$package.html", $template-
> fetch('classtrees.tpl'));
766         phpDocumentor_out("\n");
767         flush();
768     }
769     // free up considerable memory
770     unset($this-> elements);
771     unset($this-> pkg_elements);
772 }
773 /**
774 * This function takes an {@link abstractLink} descendant and returns an html link
775 *
776 * @param abstractLink a descendant of abstractlink should be passed, and never text
777 * @param string text to display in the link
778 * @param boolean this parameter is not used, and is deprecated
779 * @param boolean determines whether the returned text is enclosed in an <a> tag
780 */
781 function returnSee(& $element, $eltext = false, $with_a = true)
782 {
783     if (!is_object($element) || !$element) return false;
784     if (!$with_a) return $this-> getId($element, false);
785     if (!$eltext)
786     {
787         $eltext = '';
788         switch($element-> type)
789         {
790             case 'tutorial' :
791                 $eltext = strip_tags($element-> title);
792                 break;
793             case 'method' :
794                 case 'var' :
795                 case 'const' :
796                     $eltext .= $element-> class.'::';
797             case 'page' :
798                 case 'define' :
799                 case 'class' :
800                 case 'function' :
801                 case 'global' :
802                     default :
803                         $eltext .= $element-> name;
804                         if ($element-> type == 'function' || $element-> type == 'method') $eltext
805                         .= '()';
806                         break;

```

```

806         }
807     }
808     return '<a href=""' . $this-> getId($element) . '">' . $eltext. '</a>' ;
809 }
810
811 function getId($element, $fullpath = true)
812 {
813     if (phpDocumentor_get_class($element) == 'parserdata')
814     {
815         $element = $this-> addLink($element-> parent);
816         $elp = $element-> parent;
817     } elseif (is_a($element, 'parserbase'))
818     {
819         $elp = $element;
820         $element = $this-> addLink($element);
821     }
822     $c = '';
823     if (!empty($element-> subpackage))
824     {
825         $c = '/'. $element-> subpackage;
826     }
827     $b = "{$subdir}";
828     switch ($element-> type)
829     {
830         case 'page' :
831             if ($fullpath)
832                 return $b.$element-> package.$c.'/'.$element-> fileAlias.'.html';
833             return 'top';
834             break;
835         case 'define' :
836         case 'global' :
837         case 'function' :
838             if ($fullpath)
839                 return $b.$element-> package.$c.'/'.$element-> fileAlias.'.html#'.$element-
840             > type.$element-> name;
841             return $element-> type.$element-> name;
842             break;
843         case 'class' :
844             if ($fullpath)
845                 return $b.$element-> package.$c.'/'.$element-> name.'.html';
846             return 'top';
847             break;
848         case 'method' :
849         case 'var' :
850         case 'const' :
851             if ($fullpath)
852                 return $b.$element-> package.$c.'/'.$element-> class.'.html#'.$element-
853             > type.$element-> name;
854             return $element-> type.$element-> name;
855             break;
856         case 'tutorial' :
857             $d = '';
858             if ($element-> section)
859             {
860                 $d = '#'.$element-> section;
861             }
862             return $b.$element-> package.$c.'/tutorial_'.$element-> name.'.html'.$d;
863     }
864 /**
865 * Convert README/INSTALL/CHANGELOG file contents to output format
866 * @param README/INSTALL/CHANGELOG
867 * @param string contents of the file
868 */
869 function Convert_RIC($name, $contents)
870 {
871     $template = & $this-> newSmarty();
872     $template-> assign('contents', $contents);
873     $template-> assign('name', $name);
874     $this-> setTargetDir($this-> base_dir);
875     $this-> writefile('ric_'.$name . '.html', $template-> fetch('ric.tpl'));
876     $this-> ric_set[$name] = true;
877 }
878
879 function ConvertTodoList()
880 {
881     $todolist = array();
882     foreach($this-> todoList as $package => $alltodos)
883     {

```

```

884     foreach($alltodos as $todos)
885     {
886         $converted = array();
887         $converted['link'] = $this-> returnSee($todos[0]);
888         if (!is_array($todos[1]))
889         {
890             $converted['todos'][] = $todos[1]-> Convert($this);
891         } else
892         {
893             foreach($todos[1] as $todo)
894             {
895                 $converted['todos'][] = $todo-> Convert($this);
896             }
897         }
898         $todolist[$package][] = $converted;
899     }
900 }
901 $templ = & $this-> newSmarty();
902 $templ-> assign('todos',$todolist);
903 $templ-> register_outputfilter('HTMLframes_outputfilter');
904 $this-> setTargetDir($this-> base_dir);
905 $this-> writefile('todolist.html',$templ-> fetch('todolist.tpl'));
906 }
907 /**
908 * Create errors.html template file output
909 *
910 * This method takes all parsing errors and warnings and spits them out ordered by file and
911 * line number.
912 * @global ErrorTracker We'll be using it's output facility
913 */
914 function ConvertErrorLog()
915 {
916     global $phpDocumentor_errors;
917     $allfiles = array();
918     $files = array();
919     $warnings = $phpDocumentor_errors-> returnWarnings();
920     $errors = $phpDocumentor_errors-> returnErrors();
921     $template = & $this-> newSmarty();
922     foreach($warnings as $warning)
923     {
924         $file = '##none';
925         $linenum = 'Warning';
926         if ($warning-> file)
927         {
928             $file = $warning-> file;
929             $allfiles[$file] = 1;
930             $linenum .= ' on line '.$warning-> linenum;
931         }
932         $files[$file]['warnings'][] = array('name' => $linenum, 'listing' => $warning-
> data);
933     }
934     foreach($errors as $error)
935     {
936         $file = '##none';
937         $linenum = 'Error';
938         if ($error-> file)
939         {
940             $file = $error-> file;
941             $allfiles[$file] = 1;
942             $linenum .= ' on line '.$error-> linenum;
943         }
944         $files[$file]['errors'][] = array('name' => $linenum, 'listing' => $error-
> data);
945     }
946     $i=1;
947     $af = array();
948     foreach($allfiles as $file => $num)
949     {
950         $af[$i++] = $file;
951     }
952     $allfiles = $af;
953     usort($allfiles,'strnatcasecmp');
954     $allfiles[0] = "Post-parsing";
955     foreach($allfiles as $i => $a)
956     {
957         $allfiles[$i] = array('file' => $a);
958     }
959     $out = array();
960     foreach($files as $file => $data)

```

```

961     {
962         if ($file == '#none') $file = 'Post-parsing';
963         $out[$file] = $data;
964     }
965     $template-> assign("files" , $allfiles);
966     $template-> assign("all" , $out);
967     $template-> assign("title" , "phpDocumentor Parser Errors and
968 Warnings");
969     $this-> setTargetDir($this-> base_dir);
970     $this-> writefile("errors.html" , $template-> fetch('errors.tpl'));
971     unset($template);
972     phpDocumentor_out("\n\nTo view errors and warnings, look at "
973 > base_dir . PATH_DELIMITER . "errors.html\n");
974     flush();
975 }
976
977 function getTutorialId($package,$subpackage,$tutorial,$id)
978 {
979     return $id;
980 }
981
982 function getCData($value)
983 {
984     return '<pre>' . htmlentities($value) . '</pre>';
985 }
986 /**
987 * Converts package page and sets its package as used in {@link $package_pages}
988 * @param parserPackagePage
989 */
990 function convertPackagepage(& $element)
991 {
992     phpDocumentor_out("\n");
993     flush();
994     $this-> package = $element-> package;
995     $this-> subpackage = '';
996     $contents = $element-> Convert($this);
997     $this-> package_pages[$element-> package] =
998 str_replace('{subdir}', '../', $contents);
999     phpDocumentor_out("\n");
1000    flush();
1001    $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $element-> package);
1002    $this-> writeFile('package_' . $element-
1003 > package . '.html', str_replace('{subdir}', '../', $contents));
1004 }
1005 /**
1006 * @param parserTutorial
1007 */
1008 function convertTutorial(& $element)
1009 {
1010     phpDocumentor_out("\n");
1011     flush();
1012     $template = & parent::convertTutorial($element);
1013     $a = '../';
1014     if ($element-> subpackage) $a .= '/';
1015     $template-> assign('subdir', $a);
1016     $template-> register_outputfilter('HTMLframes_outputfilter');
1017     $contents = $template-> fetch('tutorial.tpl');
1018     $a = '';
1019     if ($element-> subpackage) $a = PATH_DELIMITER . $element-> subpackage;
1020     phpDocumentor_out("\n");
1021     flush();
1022     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $element-> package . $a);
1023     $this-> writeFile('tutorial_' . $element-> name . '.html', $contents);
1024 }
1025 /**
1026 * Converts class for template output
1027 * @see prepareDocBlock(), generateChildClassList(), generateFormattedClassTree(),
1028 * getFormattedConflicts()
1029 * @param parserClass
1030 */
1031 function convertClass(& $element)
1032 {
1033     parent::convertClass($element);
1034     $this-> class_dir = $element-> docblock-> package;
1035     if (!empty($element-> docblock-> subpackage)) $this-> class_dir .= PATH_DELIMITER
1036 . $element-> docblock-> subpackage;

```

```

1035     $a = '../';
1036     if ($element-> docblock-> subpackage != '') $a = "      ./{$a}"      ;
1037
1038     $this-> class_data-> assign('subdir', $a);
1039     $this-> class_data-> assign("title"           , "Docs For Class "
1040     $element-> getName());;
1041     $this-> class_data-> assign("page"           , $element-> getName() . '.html');
1042 }
1043 /**
1044 * Converts class variables for template output
1045 * @see prepareDocBlock(), getFormattedConflicts()
1046 * @param parserDefine
1047 */
1048 function convertVar(& $element)
1049 {
1050     parent::convertVar($element, array('var_dest' => $this-> getId($element, false)));
1051 }
1052
1053 /**
1054 * Converts class variables for template output
1055 * @see prepareDocBlock(), getFormattedConflicts()
1056 * @param parserDefine
1057 */
1058 function convertConst(& $element)
1059 {
1060     parent::convertConst($element, array('const_dest' => $this-
> getId($element, false)));
1061 }
1062
1063 /**
1064 * Converts class methods for template output
1065 * @see prepareDocBlock(), getFormattedConflicts()
1066 * @param parserDefine
1067 */
1068 function convertMethod(& $element)
1069 {
1070     parent::convertMethod($element, array('method_dest' => $this-
> getId($element, false)));
1071 }
1072
1073 /**
1074 * Converts function for template output
1075 * @see prepareDocBlock(), parserFunction::getFunctionCall(), getFormattedConflicts()
1076 * @param parserFunction
1077 */
1078 function convertFunction(& $element)
1079 {
1080     $funcloc = $this-> getId($this-> addLink($element));
1081     parent::convertFunction($element, array('function_dest' => $this-
> getId($element, false)));
1082 }
1083
1084 /**
1085 * Converts include elements for template output
1086 * @see prepareDocBlock()
1087 * @param parserInclude
1088 */
1089 function convertInclude(& $element)
1090 {
1091     parent::convertInclude($element, array('include_file' => '_'.strtr($element-
> getValue(), array('"' => '"', '"' => "'", ',' => '_'))));
1092 }
1093
1094 /**
1095 * Converts defines for template output
1096 * @see prepareDocBlock(), getFormattedConflicts()
1097 * @param parserDefine
1098 */
1099 function convertDefine(& $element)
1100 {
1101     parent::convertDefine($element, array('define_link' => $this-
> getId($element, false)));
1102 }
1103
1104 /**
1105 * Converts global variables for template output
1106 * @param parserGlobal
1107 */
1108 function convertGlobal(& $element)

```

```

1109     {
1110         parent::convertGlobal($element, array('global_link' => $this-
1111 > _getId($element, false)));
1112     }
1113 
1114     /**
1115      * converts procedural pages for template output
1116      * @see prepareDocBlock(), getClassessOnPage()
1117      * @param parserData
1118      */
1119     function convertPage(& $element)
1120     {
1121         parent::convertPage($element);
1122         $this-> juststarted = true;
1123         $this-> page_dir = $element-> parent-> package;
1124         if (!empty($element-> parent-> subpackage)) $this-> page_dir .= PATH_DELIMITER .
1125         $element-> parent-> subpackage;
1126         // registering stuff on the template
1127         $this-> page_data-> assign("page" , $this-> getPageName($element) .
1128         '.html');
1129         $this-> page_data-> assign("title" , "Docs for page " . $element-
1130 > parent-> getFile());
1131 
1132         function getPageName(& $element)
1133     {
1134         if (phpDocumentor_get_class($element) == 'parserpage') return '_'. $element-
1135 > getName();
1136         return '_'. $element-> parent-> getName();
1137     }
1138 
1139     /**
1140      * returns an array containing the class inheritance tree from the root object to the class
1141      *
1142      * @param parserClass    class variable
1143      * @return array Format: array(root,child,child,child,...,$class)
1144      * @uses parserClass::getParentClassTree()
1145      */
1146 
1147     function generateFormattedClassTree($class)
1148     {
1149         $tree = $class-> getParentClassTree($this);
1150         $out = '';
1151         if (count($tree) - 1)
1152         {
1153             $result = array($class-> getName());
1154             $parent = $tree[$class-> getName()];
1155             $distance[] = '';
1156             while ($parent)
1157             {
1158                 $x = $parent;
1159                 if (is_object($parent))
1160                 {
1161                     $subpackage = $parent-> docblock-> subpackage;
1162                     $package = $parent-> docblock-> package;
1163                     $x = $parent;
1164                     $x = $parent-> getLink($this);
1165                     if (!$x) $x = $parent-> getName();
1166                 }
1167                 $result[] =
1168                 $x;
1169                 $distance[] =
1170                 "\n$%|\n"
1171                 "%$--";
1172                 if (is_object($parent))
1173                 $parent = $tree[$parent-> getName()];
1174                 elseif (isset($tree[$parent]))
1175                 $parent = $tree[$parent];
1176             }
1177             $nbsp = ' ';
1178             for($i=count($result) - 1;$i>= 0;$i--)
1179             {
1180                 $my_nbsp = '';
1181                 for($j=0;$j< count($result) - $i;$j++) $my_nbsp .= $nbsp;
1182                 $distance[$i] = sprintf($distance[$i], $my_nbsp, $my_nbsp);
1183             }
1184             return
array('classes'=> array_reverse($result), 'distance'=> array_reverse($distance));
1185         } else
1186     {

```

```

1183         return array('classes' => $class-> getName(), 'distance' => array(    ''));
1184     }
1185 }
1186
1187 /** @access private */
1188 function sortVar($a, $b)
1189 {
1190     return strnatcasecmp($a-> getName(), $b-> getName());
1191 }
1192
1193 /** @access private */
1194 function sortMethod($a, $b)
1195 {
1196     if ($a-> isConstructor) return -1;
1197     if ($b-> isConstructor) return 1;
1198     return strnatcasecmp($a-> getName(), $b-> getName());
1199 }
1200
1201 /**
1202 * returns a template-enabled array of class trees
1203 *
1204 * @param string $package package to generate a class tree for
1205 * @see $roots, HTMLConverter::getRootTree()
1206 */
1207 function generateFormattedClassTrees($package)
1208 {
1209     if (!isset($this-> roots['normal'][$package]) &&
1210         !isset($this-> roots['special'][$package])) {
1211         return array();
1212     }
1213     $trees = array();
1214     if (isset($this-> roots['normal'][$package])) {
1215         $roots = $this-> roots['normal'][$package];
1216         for ($i=0;$i< count($roots);$i++)
1217     {
1218         $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1219         if ($root && $root-> isInterface()) {
1220             continue;
1221         }
1222         $trees[] = array('class' => $roots[$i], 'class_tree' =>
1223             $this-> getRootTree($this-
1224             > getSortedClassTreeFromClass($roots[$i], $package, '') , $package). "</ul>\n"
1225         );
1226     }
1227     if (isset($this-> roots['special'][$package])) {
1228         $roots = $this-> roots['special'][$package];
1229         foreach ($roots as $parent => $classes) {
1230             $thistree = '';
1231             foreach ($classes as $classinfo) {
1232                 $root = $this-> classes-> getClassByPackage($classinfo, $package);
1233                 if ($root && $root-> isInterface()) {
1234                     continue;
1235                 }
1236                 $thistree .=
1237                     $this-> getRootTree(
1238                         $this-> getSortedClassTreeFromClass(
1239                             $classinfo,
1240                             $package,
1241                             true);
1242             }
1243             if (!$thistree) {
1244                 continue;
1245             }
1246             $trees[] = array(
1247                 'class' => $parent,
1248                 'class_tree' => "<ul>\n" . $thistree .
1249             );
1250         }
1251     }
1252     return $trees;
1253 }
1254
1255 /**
1256 * returns a template-enabled array of interface inheritance trees
1257 *
1258 * @param string $package package to generate a class tree for
1259 * @see $roots, HTMLConverter::getRootTree()

```

```

1260      */
1261     function generateFormattedInterfaceTrees($package)
1262     {
1263         if (!isset($this->roots['normal'][$package]) &&
1264             !isset($this->roots['special'][$package])) {
1265             return array();
1266         }
1267         $trees = array();
1268         if (isset($this->roots['normal'][$package])) {
1269             $roots = $this->roots['normal'][$package];
1270             for ($i=0;$i<count($roots);$i++)
1271             {
1272                 $root = $this->classes->getClassByPackage($roots[$i], $package);
1273                 if ($root && !$root->isInterface()) {
1274                     continue;
1275                 }
1276                 $trees[] = array('class' => $roots[$i], 'class_tree' =>
1277                     . $this->getRootTree($this-
1278                     >getSortedClassTreeFromClass($roots[$i], $package, ''), $package). "</ul>\n");
1279             }
1280             if (isset($this->roots['special'][$package])) {
1281                 $roots = $this->roots['special'][$package];
1282                 foreach ($roots as $parent => $classes) {
1283                     $thistree = '';
1284                     foreach ($classes as $classinfo) {
1285                         $root = $this->classes->getClassByPackage($classinfo, $package);
1286                         if ($root && !$root->isInterface()) {
1287                             continue;
1288                         }
1289                         $thistree .=
1290                             $this->getRootTree(
1291                                 $this->getSortedClassTreeFromClass(
1292                                     $classinfo,
1293                                     $package,
1294                                     ''),
1295                                     $package,
1296                                     true);
1297                     if (!$thistree) {
1298                         continue;
1299                     }
1300                     $trees[] = array(
1301                         'class' => $parent,
1302                         'class_tree' => "<ul>\n" . $thistree .
1303                         );
1304                 }
1305             }
1306             return $trees;
1307         }
1308     /**
1309      * return formatted class tree for the Class Trees page
1310      *
1311      * @param array $tree output from {@link getSortedClassTreeFromClass()}
1312      * @param string $package package
1313      * @param boolean $nouknownparent if true, an object's parent will not be checked
1314      * @see Classes:::$definitechild, generateFormattedClassTrees()
1315      * @return string
1316      */
1317     function getRootTree($tree, $package, $noperent = false)
1318     {
1319         if (!$tree) return '';
1320         $my_tree = '';
1321         $cur = '#root';
1322         $lastcur = array(false);
1323         $kids = array();
1324         $dopar = false;
1325         if (!$noperent && $tree[$cur]['parent'])
1326         {
1327             $dopar = true;
1328             if (!is_object($tree[$cur]['parent']))
1329             {
1330                 debug("parent ".$tree[$cur]['parent']." not found");
1331                 $my_tree .= '<li>' . $tree[$cur]['parent'] . '<ul>' ;
1332             }
1333             else
1334             {
1335                 debug("parent ".$this-
1336             //
```

```

>returnSee($tree[$cur]['parent'])." in other package");
1337     $root = $this-> classes-> getClassByPackage($tree[$cur]['parent']-> name,
1338                                         $package);
1339     $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['parent']);
1340     if ($tree[$cur]['parent']-> package != $package) $my_tree .= '
1341 <b>(Different package)</b><ul>';
1342     }
1343     do
1344     {
1345         // fancy_debug($cur,$lastcur,$kids);
1346         if (count($tree[$cur]['children']))
1347         {
1348             // debug("$cur has children");
1349             if (!isset($kids[$cur]))
1350             {
1351                 // debug("set $cur kids");
1352                 $kids[$cur] = 1;
1353                 $root = $this-> classes-> getClassByPackage(
1354                     $tree[$cur]['link']-> name,
1355                     $tree[$cur]['link']-> package);
1356                 if ($implements = $root-> getImplements())
1357                 {
1358                     $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link']) .
1359                     ' (implements ';
1360                     foreach ($implements as $i => $interface) {
1361                         if ($i && $i != count($implements) - 1) $my_tree .= ', ';
1362                         if ($link = $this-> getLink('object' . $interface)) {
1363                             $my_tree .= $this-> returnSee($link);
1364                         } else {
1365                             $my_tree .= $interface;
1366                         }
1367                     $my_tree .= ')';
1368                 } else {
1369                     $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link']);
1370                 }
1371                 $my_tree .= '<ul>' . "\n";
1372             }
1373             array_push($lastcur,$cur);
1374             list($cur) = each($tree[$cur]['children']);
1375             var_dump('listed',$cur);
1376             if ($cur)
1377             {
1378                 $cur = $cur['package'] . '#' . $cur['class'];
1379                 // debug("set cur to child $cur");
1380                 $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link']);
1381                 continue;
1382             } else
1383             {
1384                 // debug("end of children for $cur");
1385                 $cur = array_pop($lastcur);
1386                 $cur = array_pop($lastcur);
1387                 $my_tree .= '</ul></li>' . "\n";
1388                 if ($dopar && ($cur == '#root' || !$cur)) $my_tree .=
1389 '</ul></li>';
1390             }
1391         }
1392     }
1393     // debug("$cur has no children");
1394     $my_tree .= '<li>' . $this-
> returnSee($tree[$cur]['link'])."</li>";
1395     if ($dopar && ($cur == '#root')) $my_tree .= '</ul></li>';
1396     $cur = array_pop($lastcur);
1397     } while ($cur);
1398     return $my_tree;
1399 }
1400 /**
1401 * Generate indexing information for given element
1402 *
1403 * @param parserElement descendant of parserElement
1404 * @see generateElementIndex()
1405 * @return array
1406 */
1407
1408 function getIndexInformation($elt)
1409 {
1410     $Result['type'] = $elt-> type;
1411     $Result['file_name'] = $elt-> file;
1412     $Result['path'] = $elt-> getPath();

```

```

1413
1414     if (isset($elt-> docblock))
1415     {
1416         $Result['description'] = $elt-> docblock-> getSDesc($this);
1417
1418         if ($elt-> docblock-> hasaccess)
1419             $Result['access'] = $elt-> docblock-> tags['access'][0]-
1420             > value;
1421
1422         else
1423             $Result['access'] = 'public';
1424
1425     }
1426
1427     else
1428         $Result['description'] = '';
1429
1430     $aa = $Result['description'];
1431     if (!empty($aa)) $aa =
1432     "<br>&nbsp;&nbsp;&nbsp;" . $aa" ;
1433
1434     switch($elt-> type)
1435     {
1436         case 'class':
1437             $Result['name'] = $elt-> getName();
1438             $Result['title'] = 'Class';
1439             $Result['link'] = $this-> getClassLink($elt-> getName(),
1440                                         $elt-> docblock-> package,
1441                                         $elt-> getPath(),
1442                                         $elt-> getName());
1443
1444             $Result['listing'] = 'in file '.$elt-> file.', class
1445             $aa" ;
1446             break;
1447         case 'define':
1448             $Result['name'] = $elt-> getName();
1449             $Result['title'] = 'Constant';
1450             $Result['link'] = $this-> getDefineLink($elt-> getName(),
1451                                         $elt-> docblock-> package,
1452                                         $elt-> getPath(),
1453                                         $elt-> getName());
1454
1455             $Result['listing'] = 'in file '.$elt-> file.', constant
1456             $aa" ;
1457             break;
1458         case 'global':
1459             $Result['name'] = $elt-> getName();
1460             $Result['title'] = 'Global';
1461             $Result['link'] = $this-> getGlobalLink($elt-> getName(),
1462                                         $elt-> docblock-> package,
1463                                         $elt-> getPath(),
1464                                         $elt-> getName());
1465
1466             $Result['listing'] = 'in file '.$elt-> file.', global variable
1467             $aa" ;
1468             break;
1469         case 'function':
1470             $Result['name'] = $elt-> getName();
1471             $Result['title'] = 'Function';
1472             $Result['link'] = $this-> getFunctionLink($elt-> getName(),
1473                                         $elt-> docblock-> package,
1474                                         $elt-> getPath(),
1475                                         $elt-> getName().'()');
1476
1477             $Result['listing'] = 'in file '.$elt-> file.', function
1478             $aa" ;
1479             break;
1480         case 'method':
1481             $Result['name'] = $elt-> getName();
1482             $Result['title'] = 'Method';
1483             $Result['link'] = $this-> getMethodLink($elt-> getName(),
1484                                         $elt-> class,
1485                                         $elt-> docblock-> package,
1486                                         $elt-> getPath(),
1487                                         $elt-> class.'::'.$elt-
1488             > getName().'()'
1489             );
1490
1491             if ($elt-> isConstructor)
1492                 $Result['listing'] = 'in file '.$elt-> file.', method
1493             $aa" ;
1494             break;
1495         case 'var':
1496             $Result['name'] = $elt-> getName();
1497             $Result['title'] = 'Variable';

```

```

1484             $Result['link'] = $this-> getVarLink($elt-> getName(),
1485                                         $elt-> class,
1486                                         $elt-> docblock-> package,
1487                                         $elt-> getPath(),
1488                                         $elt-> class.'::'.$elt-
1489             > getName());
1490             $Result['listing'] = 'in file '.$elt-> file.', variable
1491             '$aa" ;
1492             break;
1493             case 'const':
1494                 $Result['name'] = $elt-> getName();
1495                 $Result['title'] = 'Class Constant';
1496                 $Result['link'] = $this-> getConstLink($elt-> getName(),
1497                                         $elt-> class,
1498                                         $elt-> docblock-> package,
1499                                         $elt-> getPath(),
1500                                         $elt-> class.'::'.$elt-
1501             > getName());
1502             $Result['listing'] = 'in file '.$elt-> file.', class constant
1503             '$aa" ;
1504             break;
1505             case 'page':
1506                 $Result['name'] = $elt-> getFile();
1507                 $Result['title'] = 'Page';
1508                 $Result['link'] = $this-> getPageLink($elt-> getFile(),
1509                                         $elt-> package,
1510                                         $elt-> getPath(),
1511                                         $elt-> getFile());
1512                 $Result['listing'] = 'procedural page '.$Result['link'];
1513             break;
1514             case 'include':
1515                 $Result['name'] = $elt-> getName();
1516                 $Result['title'] = 'Include';
1517                 $Result['link'] = $elt-> getValue();
1518                 $Result['listing'] = 'include '.$Result['name'];
1519             break;
1520         }
1521     }
1522     return $Result;
1523 }
1524 /**
1525 * Generate alphabetical index of all elements
1526 *
1527 * @see $elements, walk()
1528 */
1529 function generateElementIndex()
1530 {
    $elementindex = array();
    $letters = array();
    $used = array();
    foreach($this-> elements as $letter => $nutoh)
    {
        foreach($this-> elements[$letter] as $i => $yuh)
        {
            if ($this-> elements[$letter][$i]-> type != 'include')
            {
                if (!isset($used[$letter]))
                {
                    $letters[]['letter'] = $letter;
                    $selindex['letter'] = $letter;
                    $used[$letter] = 1;
                }
                $selindex['index'][] = $this-> getIndexInformation($this-
> elements[$letter][$i]);
            }
            if (isset($selindex['index']))
            {
                $elementindex[] = $selindex;
            } else
            {
                unset($letters[count($letters) - 1]);
            }
            $selindex = array();
        }
        return array($elementindex,$letters);
    }
}
1554 function copyMediaRecursively($media,$targetdir,$subdir = '')

```

```

1559     {
1560         $versionControlDirectories = array ('CVS', 'media/CVS', 'media\\CVS', '.svn',
1561         'media/.svn', 'media\\.svn');
1562         if (!is_array($media)) {
1563             return;
1564         }
1565         foreach($media as $dir =>    $files)
1566         {
1567             if ($dir === '/')
1568             {
1569                 $this->  copyMediaRecursively($files,$targetdir);
1570             }
1571             if (!is_numeric($dir))
1572             {
1573                 if (in_array($dir, $versionControlDirectories))
1574                 {
1575                     // skip it entirely
1576                 }
1577                 else
1578                 {
1579                     // create the subdir
1580                     phpDocumentor_out("      creating $targetdir" . PATH_DELIMITER .
1581 " $dir\n" );
1582                     Converter::setTargetDir($targetdir . PATH_DELIMITER . $dir);
1583                     if (!empty($subdir))
1584                     {
1585                         $subdir .= PATH_DELIMITER;
1586                     }
1587                     $this-
1588 >  copyMediaRecursively($files,"      $targetdir/$dir" , $subdir . $dir);
1589                 }
1590             }
1591             else
1592             {
1593                 // copy the file
1594                 phpDocumentor_out("      copying $targetdir" . PATH_DELIMITER .
1592 $files['file']. "\n");
1593                 $this->  copyFile($files['file'], $subdir);
1594             }
1595         }
1596     }
1597 }
1598 /**
1599  * calls the converter setTargetDir, and then copies any template images and the stylesheet
1600  * if they haven't been copied
1601  * @see Converter::setTargetDir()
1602  */
1603 function setTargetDir($dir)
1604 {
1605     Converter::setTargetDir($dir);
1606     if ($this-> wrote) return;
1607     $this-> wrote = true;
1608     $template_images = array();
1609     $stylesheets = array();
1610     $stdir = $dir;
1611     $dir = $this-> templateDir;
1612     $this-> templateDir = $this-> templateDir.'templates/';
1613     $info = new Io;
1614     $this-> copyMediaRecursively($info-> getDirTree($this-> templateDir.'media', $this-
1614 > templateDir), $stdir);
1615 }
1616 /**
1617  * Generate alphabetical index of all elements by package and subpackage
1618  *
1619  * @param string $package name of a package
1620  * @see $pkg_elements, walk(), generatePkgElementIndexes()
1621  */
1622 function generatePkgElementIndex($package)
1623 {
1624     var_dump($this->pkg_elements[$package]);
1625     $elementindex = array();
1626     $letters = array();
1627     $letterind = array();
1628     $used = array();
1629     $subp = '';
1630     foreach($this-> pkg_elements[$package] as $subpackage =>    $els)
1631     {

```

```

1633         if (empty($els)) continue;
1634         if (!empty($subpackage)) $subp = "      (<b>subpackage:</b>
$subpackage)" ;
1635         else $subp = '';
1636         foreach($els as $letter =>    $yuh)
1637         {
1638             foreach($els[$letter] as $i =>    $yuh)
1639             {
1640                 if ($els[$letter][$i]-> type != 'include')
1641                 {
1642                     if (!isset($used[$letter]))
1643                     {
1644                         $letters[]['letter'] = $letter;
1645                         $letterind[$letter] = count($letters) - 1;
1646                         $used[$letter] = 1;
1647                     }
1648                     $elindex[$letter]['letter'] = $letter;
1649                     $elindex[$letter]['index'][] = $this-
1650 >     getIndexInformation($els[$letter][$i]);
1651             }
1652         }
1653     }
1654     ksort($elindex);
1655     usort($letters, 'HTMLframes_lettersort');
1656     if (isset($elindex))
1657     {
1658         while(list($letter,$tempel) = each($elindex))
1659         {
1660             if (!isset($tempel))
1661             {
1662                 unset($letters[$letterind[$tempel['letter']]]);
1663             }
1664             $elementindex[] = $tempel;
1665         }
1666     } else $letters = array();
1667     return array($elementindex,$letters);
1668 }
1669 /**
1670 *
1671 * @see generatePkgElementIndex()
1672 */
1673 function generatePkgElementIndexes()
1674 {
1675     $packages = array();
1676     $package_names = array();
1677     $pkg = array();
1678     $letters = array();
1679     foreach($this-> pkg_elements as $package =>    $trash)
1680     {
1681         $pkgs['package'] = $package;
1682         $pkg['package'] = $package;
1683         list($pkg['pindex'],$letters[$package]) = $this-
1684 >     generatePkgElementIndex($package);
1685         if (count($pkg['pindex']))
1686         {
1687             $packages[] = $pkg;
1688             $package_names[] = $pkgs;
1689         }
1690         unset($pkgs);
1691         unset($pkg);
1692     }
1693     foreach($packages as $i =>    $package)
1694     {
1695         $pnames = array();
1696         for($j=0;$j< count($package_names);$j++)
1697         {
1698             if ($package_names[$j]['package'] != $package['package']) $pnames[] =
1699             $package_names[$j];
1700         }
1701         $packages[$i]['packageindexes'] = $pnames;
1702     }
1703     return array($packages,$package_names,$letters);
1704 }
1705 /**
1706 * @param string name of class
1707 * @param string package name
1708 * @param string full path to look in (used in index generation)

```

```

1709 * @param boolean deprecated
1710 * @param boolean return just the URL, or enclose it in an html a tag
1711 * @return mixed false if not found, or an html a link to the class's documentation
1712 * @see parent::getClassLink()
1713 */
1714 function getClassLink($expr,$package, $file = false,$text = false, $with_a = true)
1715 {
1716     $a = Converter::getClassLink($expr,$package,$file);
1717     if (!$a) return false;
1718     return $this->  returnSee($a, $text, $with_a);
1719 }
1720
1721 /**
1722 * @param string name of function
1723 * @param string package name
1724 * @param string full path to look in (used in index generation)
1725 * @param boolean deprecated
1726 * @param boolean return just the URL, or enclose it in an html a tag
1727 * @return mixed false if not found, or an html a link to the function's documentation
1728 * @see parent::getFunctionLink()
1729 */
1730 function getFunctionLink($expr,$package, $file = false,$text = false)
1731 {
1732     $a = Converter::getFunctionLink($expr,$package,$file);
1733     if (!$a) return false;
1734     return $this->  returnSee($a, $text);
1735 }
1736
1737 /**
1738 * @param string name of define
1739 * @param string package name
1740 * @param string full path to look in (used in index generation)
1741 * @param boolean deprecated
1742 * @param boolean return just the URL, or enclose it in an html a tag
1743 * @return mixed false if not found, or an html a link to the define's documentation
1744 * @see parent::getDefineLink()
1745 */
1746 function getDefineLink($expr,$package, $file = false,$text = false)
1747 {
1748     $a = Converter::getDefineLink($expr,$package,$file);
1749     if (!$a) return false;
1750     return $this->  returnSee($a, $text);
1751 }
1752
1753 /**
1754 * @param string name of global variable
1755 * @param string package name
1756 * @param string full path to look in (used in index generation)
1757 * @param boolean deprecated
1758 * @param boolean return just the URL, or enclose it in an html a tag
1759 * @return mixed false if not found, or an html a link to the global variable's
documentation
1760 * @see parent::getGlobalLink()
1761 */
1762 function getGlobalLink($expr,$package, $file = false,$text = false)
1763 {
1764     $a = Converter::getGlobalLink($expr,$package,$file);
1765     if (!$a) return false;
1766     return $this->  returnSee($a, $text);
1767 }
1768
1769 /**
1770 * @param string name of procedural page
1771 * @param string package name
1772 * @param string full path to look in (used in index generation)
1773 * @param boolean deprecated
1774 * @param boolean return just the URL, or enclose it in an html a tag
1775 * @return mixed false if not found, or an html a link to the procedural page's
documentation
1776 * @see parent::getPageLink()
1777 */
1778 function getPageLink($expr,$package, $path = false,$text = false)
1779 {
1780     $a = Converter::getPageLink($expr,$package,$path);
1781     if (!$a) return false;
1782     return $this->  returnSee($a, $text);
1783 }
1784
1785 /**
1786 * @param string name of method

```

```

1787 * @param string class containing method
1788 * @param string package name
1789 * @param string full path to look in (used in index generation)
1790 * @param boolean deprecated
1791 * @param boolean return just the URL, or enclose it in an html a tag
1792 * @return mixed false if not found, or an html a link to the method's documentation
1793 * @see parent::getMethodLink()
1794 */
1795 function getMethodLink($expr,$class,$package, $file = false,$text = false)
1796 {
1797     $a = Converter::getMethodLink($expr,$class,$package,$file);
1798     if (!$a) return false;
1799     return $this->  returnSee($a, $text);
1800 }
1801 /**
1802 * @param string name of var
1803 * @param string class containing var
1804 * @param string package name
1805 * @param string full path to look in (used in index generation)
1806 * @param boolean deprecated
1807 * @param boolean return just the URL, or enclose it in an html a tag
1808 * @return mixed false if not found, or an html a link to the var's documentation
1809 * @see parent::getVarLink()
1810 */
1811 function getVarLink($expr,$class,$package, $file = false,$text = false)
1812 {
1813     $a = Converter::getVarLink($expr,$class,$package,$file);
1814     if (!$a) return false;
1815     return $this->  returnSee($a, $text);
1816 }
1817 /**
1818 * @param string name of class constant
1819 * @param string class containing class constant
1820 * @param string package name
1821 * @param string full path to look in (used in index generation)
1822 * @param boolean deprecated
1823 * @param boolean return just the URL, or enclose it in an html a tag
1824 * @return mixed false if not found, or an html a link to the var's documentation
1825 * @see parent::getVarLink()
1826 */
1827 function getConstLink($expr,$class,$package, $file = false,$text = false)
1828 {
1829     $a = Converter::getConstLink($expr,$class,$package,$file);
1830     if (!$a) return false;
1831     return $this->  returnSee($a, $text);
1832 }
1833 /**
1834 * does a nat case sort on the specified second level value of the array
1835 *
1836 * @param mixed $a
1837 * @param mixed $b
1838 * @return int
1839 */
1840 function rcNatCmp ($a, $b)
1841 {
1842     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1843     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1844
1845     return strnatcasecmp($aa, $bb);
1846 }
1847 /**
1848 * does a nat case sort on the specified second level value of the array.
1849 * this one puts constructors first
1850 *
1851 * @param mixed $a
1852 * @param mixed $b
1853 * @return int
1854 */
1855 function rcNatCmpl ($a, $b)
1856 {
1857     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1858     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1859
1860     if (strpos($aa,'CONSTRUCTOR') === 0)
1861     {
1862         return -1;
1863     }
1864
1865     if (strpos($bb,'CONSTRUCTOR') === 0)
1866     {
1867         return 1;
1868     }
1869
1870     if ($aa < $bb)
1871     {
1872         return -1;
1873     }
1874
1875     if ($aa > $bb)
1876     {
1877         return 1;
1878     }
1879
1880     return 0;
1881 }
1882
1883 /**
1884 * does a nat case sort on the specified second level value of the array.
1885 * this one puts constructors first
1886 *
1887 * @param mixed $a
1888 * @param mixed $b
1889 * @return int
1890 */
1891 function rcNatCmp2 ($a, $b)
1892 {
1893     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1894     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1895
1896     if (strpos($aa,'CONSTRUCTOR') === 0)
1897     {
1898         return -1;
1899     }
1900
1901     if (strpos($bb,'CONSTRUCTOR') === 0)
1902     {
1903         return 1;
1904     }
1905
1906     if ($aa < $bb)
1907     {
1908         return -1;
1909     }
1910
1911     if ($aa > $bb)
1912     {
1913         return 1;
1914     }
1915
1916     return 0;
1917 }
1918
1919 /**
1920 * does a nat case sort on the specified second level value of the array.
1921 * this one puts constructors first
1922 *
1923 * @param mixed $a
1924 * @param mixed $b
1925 * @return int
1926 */
1927 function rcNatCmp3 ($a, $b)
1928 {
1929     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1930     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1931
1932     if (strpos($aa,'CONSTRUCTOR') === 0)
1933     {
1934         return -1;
1935     }
1936
1937     if (strpos($bb,'CONSTRUCTOR') === 0)
1938     {
1939         return 1;
1940     }
1941
1942     if ($aa < $bb)
1943     {
1944         return -1;
1945     }
1946
1947     if ($aa > $bb)
1948     {
1949         return 1;
1950     }
1951
1952     return 0;
1953 }
1954
1955 /**
1956 * does a nat case sort on the specified second level value of the array.
1957 * this one puts constructors first
1958 *
1959 * @param mixed $a
1960 * @param mixed $b
1961 * @return int
1962 */
1963 function rcNatCmp4 ($a, $b)
1964 {
1965     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1966     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1967
1968     if (strpos($aa,'CONSTRUCTOR') === 0)
1969     {
1970         return -1;
1971     }
1972
1973     if (strpos($bb,'CONSTRUCTOR') === 0)
1974     {
1975         return 1;
1976     }
1977
1978     if ($aa < $bb)
1979     {
1980         return -1;
1981     }
1982
1983     if ($aa > $bb)
1984     {
1985         return 1;
1986     }
1987
1988     return 0;
1989 }
1990
1991 /**
1992 * does a nat case sort on the specified second level value of the array.
1993 * this one puts constructors first
1994 *
1995 * @param mixed $a
1996 * @param mixed $b
1997 * @return int
1998 */
1999 function rcNatCmp5 ($a, $b)
2000 {
2001     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2002     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2003
2004     if (strpos($aa,'CONSTRUCTOR') === 0)
2005     {
2006         return -1;
2007     }
2008
2009     if (strpos($bb,'CONSTRUCTOR') === 0)
2010     {
2011         return 1;
2012     }
2013
2014     if ($aa < $bb)
2015     {
2016         return -1;
2017     }
2018
2019     if ($aa > $bb)
2020     {
2021         return 1;
2022     }
2023
2024     return 0;
2025 }
2026
2027 /**
2028 * does a nat case sort on the specified second level value of the array.
2029 * this one puts constructors first
2030 *
2031 * @param mixed $a
2032 * @param mixed $b
2033 * @return int
2034 */
2035 function rcNatCmp6 ($a, $b)
2036 {
2037     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2038     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2039
2040     if (strpos($aa,'CONSTRUCTOR') === 0)
2041     {
2042         return -1;
2043     }
2044
2045     if (strpos($bb,'CONSTRUCTOR') === 0)
2046     {
2047         return 1;
2048     }
2049
2050     if ($aa < $bb)
2051     {
2052         return -1;
2053     }
2054
2055     if ($aa > $bb)
2056     {
2057         return 1;
2058     }
2059
2060     return 0;
2061 }
2062
2063 /**
2064 * does a nat case sort on the specified second level value of the array.
2065 * this one puts constructors first
2066 *
2067 * @param mixed $a
2068 * @param mixed $b
2069 * @return int
2070 */
2071 function rcNatCmp7 ($a, $b)
2072 {
2073     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2074     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2075
2076     if (strpos($aa,'CONSTRUCTOR') === 0)
2077     {
2078         return -1;
2079     }
2080
2081     if (strpos($bb,'CONSTRUCTOR') === 0)
2082     {
2083         return 1;
2084     }
2085
2086     if ($aa < $bb)
2087     {
2088         return -1;
2089     }
2090
2091     if ($aa > $bb)
2092     {
2093         return 1;
2094     }
2095
2096     return 0;
2097 }
2098
2099 /**
2100 * does a nat case sort on the specified second level value of the array.
2101 * this one puts constructors first
2102 *
2103 * @param mixed $a
2104 * @param mixed $b
2105 * @return int
2106 */
2107 function rcNatCmp8 ($a, $b)
2108 {
2109     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2110     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2111
2112     if (strpos($aa,'CONSTRUCTOR') === 0)
2113     {
2114         return -1;
2115     }
2116
2117     if (strpos($bb,'CONSTRUCTOR') === 0)
2118     {
2119         return 1;
2120     }
2121
2122     if ($aa < $bb)
2123     {
2124         return -1;
2125     }
2126
2127     if ($aa > $bb)
2128     {
2129         return 1;
2130     }
2131
2132     return 0;
2133 }
2134
2135 /**
2136 * does a nat case sort on the specified second level value of the array.
2137 * this one puts constructors first
2138 *
2139 * @param mixed $a
2140 * @param mixed $b
2141 * @return int
2142 */
2143 function rcNatCmp9 ($a, $b)
2144 {
2145     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2146     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2147
2148     if (strpos($aa,'CONSTRUCTOR') === 0)
2149     {
2150         return -1;
2151     }
2152
2153     if (strpos($bb,'CONSTRUCTOR') === 0)
2154     {
2155         return 1;
2156     }
2157
2158     if ($aa < $bb)
2159     {
2160         return -1;
2161     }
2162
2163     if ($aa > $bb)
2164     {
2165         return 1;
2166     }
2167
2168     return 0;
2169 }
2170
2171 /**
2172 * does a nat case sort on the specified second level value of the array.
2173 * this one puts constructors first
2174 *
2175 * @param mixed $a
2176 * @param mixed $b
2177 * @return int
2178 */
2179 function rcNatCmp10 ($a, $b)
2180 {
2181     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2182     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2183
2184     if (strpos($aa,'CONSTRUCTOR') === 0)
2185     {
2186         return -1;
2187     }
2188
2189     if (strpos($bb,'CONSTRUCTOR') === 0)
2190     {
2191         return 1;
2192     }
2193
2194     if ($aa < $bb)
2195     {
2196         return -1;
2197     }
2198
2199     if ($aa > $bb)
2200     {
2201         return 1;
2202     }
2203
2204     return 0;
2205 }
2206
2207 /**
2208 * does a nat case sort on the specified second level value of the array.
2209 * this one puts constructors first
2210 *
2211 * @param mixed $a
2212 * @param mixed $b
2213 * @return int
2214 */
2215 function rcNatCmp11 ($a, $b)
2216 {
2217     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2218     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2219
2220     if (strpos($aa,'CONSTRUCTOR') === 0)
2221     {
2222         return -1;
2223     }
2224
2225     if (strpos($bb,'CONSTRUCTOR') === 0)
2226     {
2227         return 1;
2228     }
2229
2230     if ($aa < $bb)
2231     {
2232         return -1;
2233     }
2234
2235     if ($aa > $bb)
2236     {
2237         return 1;
2238     }
2239
2240     return 0;
2241 }
2242
2243 /**
2244 * does a nat case sort on the specified second level value of the array.
2245 * this one puts constructors first
2246 *
2247 * @param mixed $a
2248 * @param mixed $b
2249 * @return int
2250 */
2251 function rcNatCmp12 ($a, $b)
2252 {
2253     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2254     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2255
2256     if (strpos($aa,'CONSTRUCTOR') === 0)
2257     {
2258         return -1;
2259     }
2260
2261     if (strpos($bb,'CONSTRUCTOR') === 0)
2262     {
2263         return 1;
2264     }
2265
2266     if ($aa < $bb)
2267     {
2268         return -1;
2269     }
2270
2271     if ($aa > $bb)
2272     {
2273         return 1;
2274     }
2275
2276     return 0;
2277 }
2278
2279 /**
2280 * does a nat case sort on the specified second level value of the array.
2281 * this one puts constructors first
2282 *
2283 * @param mixed $a
2284 * @param mixed $b
2285 * @return int
2286 */
2287 function rcNatCmp13 ($a, $b)
2288 {
2289     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2290     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2291
2292     if (strpos($aa,'CONSTRUCTOR') === 0)
2293     {
2294         return -1;
2295     }
2296
2297     if (strpos($bb,'CONSTRUCTOR') === 0)
2298     {
2299         return 1;
2300     }
2301
2302     if ($aa < $bb)
2303     {
2304         return -1;
2305     }
2306
2307     if ($aa > $bb)
2308     {
2309         return 1;
2310     }
2311
2312     return 0;
2313 }
2314
2315 /**
2316 * does a nat case sort on the specified second level value of the array.
2317 * this one puts constructors first
2318 *
2319 * @param mixed $a
2320 * @param mixed $b
2321 * @return int
2322 */
2323 function rcNatCmp14 ($a, $b)
2324 {
2325     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2326     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2327
2328     if (strpos($aa,'CONSTRUCTOR') === 0)
2329     {
2330         return -1;
2331     }
2332
2333     if (strpos($bb,'CONSTRUCTOR') === 0)
2334     {
2335         return 1;
2336     }
2337
2338     if ($aa < $bb)
2339     {
2340         return -1;
2341     }
2342
2343     if ($aa > $bb)
2344     {
2345         return 1;
2346     }
2347
2348     return 0;
2349 }
2350
2351 /**
2352 * does a nat case sort on the specified second level value of the array.
2353 * this one puts constructors first
2354 *
2355 * @param mixed $a
2356 * @param mixed $b
2357 * @return int
2358 */
2359 function rcNatCmp15 ($a, $b)
2360 {
2361     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2362     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2363
2364     if (strpos($aa,'CONSTRUCTOR') === 0)
2365     {
2366         return -1;
2367     }
2368
2369     if (strpos($bb,'CONSTRUCTOR') === 0)
2370     {
2371         return 1;
2372     }
2373
2374     if ($aa < $bb)
2375     {
2376         return -1;
2377     }
2378
2379     if ($aa > $bb)
2380     {
2381         return 1;
2382     }
2383
2384     return 0;
2385 }
2386
2387 /**
2388 * does a nat case sort on the specified second level value of the array.
2389 * this one puts constructors first
2390 *
2391 * @param mixed $a
2392 * @param mixed $b
2393 * @return int
2394 */
2395 function rcNatCmp16 ($a, $b)
2396 {
2397     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2398     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2399
2400     if (strpos($aa,'CONSTRUCTOR') === 0)
2401     {
2402         return -1;
2403     }
2404
2405     if (strpos($bb,'CONSTRUCTOR') === 0)
2406     {
2407         return 1;
2408     }
2409
2410     if ($aa < $bb)
2411     {
2412         return -1;
2413     }
2414
2415     if ($aa > $bb)
2416     {
2417         return 1;
2418     }
2419
2420     return 0;
2421 }
2422
2423 /**
2424 * does a nat case sort on the specified second level value of the array.
2425 * this one puts constructors first
2426 *
2427 * @param mixed $a
2428 * @param mixed $b
2429 * @return int
2430 */
2431 function rcNatCmp17 ($a, $b)
2432 {
2433     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2434     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2435
2436     if (strpos($aa,'CONSTRUCTOR') === 0)
2437     {
2438         return -1;
2439     }
2440
2441     if (strpos($bb,'CONSTRUCTOR') === 0)
2442     {
2443         return 1;
2444     }
2445
2446     if ($aa < $bb)
2447     {
2448         return -1;
2449     }
2450
2451     if ($aa > $bb)
2452     {
2453         return 1;
2454     }
2455
2456     return 0;
2457 }
2458
2459 /**
2460 * does a nat case sort on the specified second level value of the array.
2461 * this one puts constructors first
2462 *
2463 * @param mixed $a
2464 * @param mixed $b
2465 * @return int
2466 */
2467 function rcNatCmp18 ($a, $b)
2468 {
2469     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2470     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2471
2472     if (strpos($aa,'CONSTRUCTOR') === 0)
2473     {
2474         return -1;
2475     }
2476
2477     if (strpos($bb,'CONSTRUCTOR') === 0)
2478     {
2479         return 1;
2480     }
2481
2482     if ($aa < $bb)
2483     {
2484         return -1;
2485     }
2486
2487     if ($aa > $bb)
2488     {
2489         return 1;
2490     }
2491
2492     return 0;
2493 }
2494
2495 /**
2496 * does a nat case sort on the specified second level value of the array.
2497 * this one puts constructors first
2498 *
2499 * @param mixed $a
2500 * @param mixed $b
2501 * @return int
2502 */
2503 function rcNatCmp19 ($a, $b)
2504 {
2505     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2506     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2507
2508     if (strpos($aa,'CONSTRUCTOR') === 0)
2509     {
2510         return -1;
2511     }
2512
2513     if (strpos($bb,'CONSTRUCTOR') === 0)
2514     {
2515         return 1;
2516     }
2517
2518     if ($aa < $bb)
2519     {
2520         return -1;
2521     }
2522
2523     if ($aa > $bb)
2524     {
2525         return 1;
2526     }
2527
2528     return 0;
2529 }
2530
2531 /**
2532 * does a nat case sort on the specified second level value of the array.
2533 * this one puts constructors first
2534 *
2535 * @param mixed $a
2536 * @param mixed $b
2537 * @return int
2538 */
2539 function rcNatCmp20 ($a, $b)
2540 {
2541     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2542     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2543
2544     if (strpos($aa,'CONSTRUCTOR') === 0)
2545     {
2546         return -1;
2547     }
2548
2549     if (strpos($bb,'CONSTRUCTOR') === 0)
2550     {
2551         return 1;
2552     }
2553
2554     if ($aa < $bb)
2555     {
2556         return -1;
2557     }
2558
2559     if ($aa > $bb)
2560     {
2561         return 1;
2562     }
2563
2564     return 0;
2565 }
2566
2567 /**
2568 * does a nat case sort on the specified second level value of the array.
2569 * this one puts constructors first
2570 *
2571 * @param mixed $a
2572 * @param mixed $b
2573 * @return int
2574 */
2575 function rcNatCmp21 ($a, $b)
2576 {
2577     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2578     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2579
2580     if (strpos($aa,'CONSTRUCTOR') === 0)
2581     {
2582         return -1;
2583     }
2584
2585     if (strpos($bb,'CONSTRUCTOR') === 0)
2586     {
2587         return 1;
2588     }
2589
2590     if ($aa < $bb)
2591     {
2592         return -1;
2593     }
2594
2595     if ($aa > $bb)
2596     {
2597         return 1;
2598     }
2599
2600     return 0;
2601 }
2602
2603 /**
2604 * does a nat case sort on the specified second level value of the array.
2605 * this one puts constructors first
2606 *
2607 * @param mixed $a
2608 * @param mixed $b
2609 * @return int
2610 */
2611 function rcNatCmp22 ($a, $b)
2612 {
2613     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2614     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2615
2616     if (strpos($aa,'CONSTRUCTOR') === 0)
2617     {
2618         return -1;
2619     }
2620
2621     if (strpos($bb,'CONSTRUCTOR') === 0)
2622     {
2623         return 1;
2624     }
2625
2626     if ($aa < $bb)
2627     {
2628         return -1;
2629     }
2630
2631     if ($aa > $bb)
2632     {
2633         return 1;
2634     }
2635
2636     return 0;
2637 }
2638
2639 /**
2640 * does a nat case sort on the specified second level value of the array.
2641 * this one puts constructors first
2642 *
2643 * @param mixed $a
2644 * @param mixed $b
2645 * @return int
2646 */
2647 function rcNatCmp23 ($a, $b)
2648 {
2649     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2650     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2651
2652     if (strpos($aa,'CONSTRUCTOR') === 0)
2653     {
2654         return -1;
2655     }
2656
2657     if (strpos($bb,'CONSTRUCTOR') === 0)
2658     {
2659         return 1;
2660     }
2661
2662     if ($aa < $bb)
2663     {
2664         return -1;
2665     }
2666
2667     if ($aa > $bb)
2668     {
2669         return 1;
2670     }
2671
2672     return 0;
2673 }
2674
2675 /**
2676 * does a nat case sort on the specified second level value of the array.
2677 * this one puts constructors first
2678 *
2679 * @param mixed $a
2680 * @param mixed $b
2681 * @return int
2682 */
2683 function rcNatCmp24 ($a, $b)
2684 {
2685     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2686     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2687
2688     if (strpos($aa,'CONSTRUCTOR') === 0)
2689     {
2690         return -1;
2691     }
2692
2693     if (strpos($bb,'CONSTRUCTOR') === 0)
2694     {
2695         return 1;
2696     }
2697
2698     if ($aa < $bb)
2699     {
2700         return -1;
2701     }
2702
2703     if ($aa > $bb)
2704     {
2705         return 1;
2706     }
2707
2708     return 0;
2709 }
2710
2711 /**
2712 * does a nat case sort on the specified second level value of the array.
2713 * this one puts constructors first
2714 *
2715 * @param mixed $a
2716 * @param mixed $b
2717 * @return int
2718 */
2719 function rcNatCmp25 ($a, $b)
2720 {
2721     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2722     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2723
2724     if (strpos($aa,'CONSTRUCTOR') === 0)
2725     {
2726         return -1;
2727     }
2728
2729     if (strpos($bb,'CONSTRUCTOR') === 0)
2730     {
2731         return 1;
2732     }
2733
2734     if ($aa < $bb)
2735     {
2736         return -1;
2737     }
2738
2739     if ($aa > $bb)
2740     {
2741         return 1;
2742     }
2743
2744     return 0;
2745 }
2746
2747 /**
2748 * does a nat case sort on the specified second level value of the array.
2749 * this one puts constructors first
2750 *
2751 * @param mixed $a
2752 * @param mixed $b
2753 * @return int
2754 */
2755 function rcNatCmp26 ($a, $b)
2756 {
2757     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2758     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2759
2760     if (strpos($aa,'CONSTRUCTOR') === 0)
2761     {
2762         return -1;
2763     }
2764
2765     if (strpos($bb,'CONSTRUCTOR') === 0)
2766     {
2767         return 1;
2768     }
2769
2770     if ($aa < $bb)
2771     {
2772         return -1;
2773     }
2774
2775     if ($aa > $bb)
2776     {
2777         return 1;
2778     }
2779
2780     return 0;
2781 }
2782
2783 /**
2784 * does a nat case sort on the specified second level value of the array.
2785 * this one puts constructors first
2786 *
2787 * @param mixed $a
2788 * @param mixed $b
2789 * @return int
2790 */
2791 function rcNatCmp27 ($a, $b)
2792 {
2793     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2794     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2795
2796     if (strpos($aa,'CONSTRUCTOR') === 0)
2797     {
2798         return -1;
2799     }
2800
2801     if (strpos($bb,'CONSTRUCTOR') === 0)
2802     {
2803         return 1;
2804     }
2805
2806     if ($aa < $bb)
2807     {
2808         return -1;
2809     }
2810
2811     if ($aa > $bb)
2812     {
2813         return 1;
2814     }
2815
2816     return 0;
2817 }
2818
2819 /**
2820 * does a nat case sort on the specified second level value of the array.
2821 * this one puts constructors first
2822 *
2823 * @param mixed $a
2824 * @param mixed $b
2825 * @return int
2826 */
2827 function rcNatCmp28 ($a, $b)
2828 {
2829     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2830     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2831
2832     if (strpos($aa,'CONSTRUCTOR') === 0)
2833     {
2834         return -1;
2835     }
2836
2837     if (strpos($bb,'CONSTRUCTOR') === 0)
2838     {
2839         return 1;
2840     }
2841
2842     if ($aa < $bb)
2843     {
2844         return -1;
2845     }
2846
2847     if ($aa > $bb)
2848     {
2849         return 1;
2850     }
2851
2852     return 0;
2853 }
2854
2855 /**
2856 * does a nat case sort on the specified second level value of the array.
2857 * this one puts constructors first
2858 *
2859 * @param mixed $a
2860 * @param mixed $b
2861 * @return int
2862 */
2863 function rcNatCmp29 ($a, $b)
2864 {
2865     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2866     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2867
2868     if (strpos($aa,'CONSTRUCTOR') === 0)
2869     {
2870         return -1;
2871     }
2872
2873     if (strpos($bb,'CONSTRUCTOR') === 0)
2874     {
2875         return 1;
2876     }
2877
2878     if ($aa < $bb)
2879     {
2880         return -1;
2881     }
2882
2883     if ($aa > $bb)
2884     {
2885         return 1;
2886     }
2887
2888     return 0;
2889 }
2890
2891 /**
2892 * does a nat case sort on the specified second level value of the array.
2893 * this one puts constructors first
2894 *
2895 * @param mixed $a
2896 * @param mixed $b
2897 * @return int
2898 */
2899 function rcNatCmp30 ($a, $b)
2900 {
2901     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2902     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2903
2904     if (strpos($aa,'CONSTRUCTOR') === 0)
2905     {
2906         return -1;
2907     }
2908
2909     if (strpos($bb,'CONSTRUCTOR') === 0)
2910     {
2911         return 1;
2912     }
2913
2914     if ($aa < $bb)
2915     {
2916         return -1;
2917     }
2918
2919     if ($aa > $bb)
2920     {
2921         return 1;
2922     }
2923
2924     return 0;
2925 }
2926
2927 /**
2928 * does a nat case sort on the specified second level value of the array.
2929 * this one puts constructors first
2930 *
2931 * @param mixed $a
2932 * @param mixed $b
2933 * @return int
2934 */
2935 function rcNatCmp31 ($a, $b)
2936 {
2937     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2938     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2939
2940     if (strpos($aa,'CONSTRUCTOR') === 0)
2941     {
2942         return -1;
2943     }
2944
2945     if (strpos($bb,'CONSTRUCTOR') === 0)
2946     {
2947         return 1;
2948     }
2949
2950     if ($aa < $bb)
2951     {
2952         return -1;
2953     }
2954
2955     if ($aa > $bb)
2956     {
2957         return 1;
2958     }
2959
2960     return 0;
2961 }
2962
2963 /**
2964 * does a nat case sort on the specified second level value of the array.
2965 * this one puts constructors first
2966 *
2967 * @param mixed $a
2968 * @param mixed $b
2969 * @return int
2970 */
2971 function rcNatCmp32 ($a, $b)
2972 {
2973     $aa = strtoupper($a[$this->  rcnatcmpkey]);
2974     $bb = strtoupper($b[$this->  rcnatcmpkey]);
2975
2976     if (strpos($aa,'CONSTRUCTOR') === 0)
2977     {
2978         return -1;
2979     }
29
```

```

1867     }
1868     if (strpos($bb,'CONSTRUCTOR') === 0)
1869     {
1870         return 1;
1871     }
1872     if (strpos($aa,strtoupper($this-> class)) === 0)
1873     {
1874         return -1;
1875     }
1876     if (strpos($bb,strtoupper($this-> class)) === 0)
1877     {
1878         return -1;
1879     }
1880     return strnatcasecmp($aa, $bb);
1881 }
1882 /**
1883 * This function is not used by HTMLdefaultConverter, but is required by Converter
1884 */
1885 function Output()
1886 {
1887 }
1888 }
1889 */
1890 /**
1891 * @access private
1892 * @global string name of the package to set as the first package
1893 */
1894
1895 function HTMLframes_pindexcmp($a, $b)
1896 {
1897     global $phpDocumentor_DefaultPackageName;
1898     if ($a['title'] == $phpDocumentor_DefaultPackageName) return -1;
1899     if ($b['title'] == $phpDocumentor_DefaultPackageName) return 1;
1900     return strnatcasecmp($a['title'],$b['title']);
1901 }
1902 /**
1903 * @access private */
1904 function HTMLframes_lettersort($a, $b)
1905 {
1906     return strnatcasecmp($a['letter'],$b['letter']);
1907 }
1908
1909 /**
1910 * @access private */
1911 function HTMLframes_outputfilter($src, & $smarty)
1912 {
1913     return str_replace('{subdir}',$smarty-> _tpl_vars['subdir'],$src);
1914 ?>

```

# File Source for HTMLSmartyConverter.inc

Documentation for this file is available at [HTMLSmartyConverter.inc](#)

```
1      <?php
2      /**
3      * HTML output converter for Smarty Template.
4      * This Converter takes output from the {@link Parser} and converts it to HTML-ready output for
use with {@link Smarty}.
5      *
6      * phpDocumentor :: automatic documentation generator
7      *
8      * PHP versions 4 and 5
9      *
10     * Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver
11     *
12     * LICENSE:
13     *
14     * This library is free software; you can redistribute it
15     * and/or modify it under the terms of the GNU Lesser General
16     * Public License as published by the Free Software Foundation;
17     * either version 2.1 of the License, or (at your option) any
18     * later version.
19     *
20     * This library is distributed in the hope that it will be useful,
21     * but WITHOUT ANY WARRANTY; without even the implied warranty of
22     * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23     * Lesser General Public License for more details.
24     *
25     * You should have received a copy of the GNU Lesser General Public
26     * License along with this library; if not, write to the Free Software
27     * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28     *
29     * @package Converters
30     * @subpackage HTMLframes
31     * @author Joshua Eichorn <jeichorn@phpdoc.org>
32     * @author Greg Beaver <cellog@php.net>
33     * @copyright 2000-2006 Joshua Eichorn, Gregory Beaver
34     * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
35     * @version CVS: $Id: HTMLSmartyConverter.inc 234145 2007-04-19 20:20:57Z ashnazg $
36     * @filesource
37     * @link http://www.phpdoc.org
38     * @link http://pear.php.net/PhpDocumentor
39     * @see parserDocBlock, parserInclude, parserPage, parserClass
40     * @see parserDefine, parserFunction, parserMethod, parserVar
41     * @since 1.0rc1
42     */
43 /**
44 * HTML output converter.
45 * This Converter takes output from the {@link Parser} and converts it to HTML-ready output for
use with {@link Smarty}.
46 *
47 * @package Converters
48 * @subpackage HTMLSmarty
49 * @see parserDocBlock, parserInclude, parserPage, parserClass, parserDefine, parserFunction,
parserMethod, parserVar
50 * @author Greg Beaver <cellog@php.net>
51 * @since 1.0rc1
52 * @version $Revision: 234145 $
53 */
54 class HTMLSmartyConverter extends Converter
55 {
56 /**
57 * This converter knows about the new root tree processing
58 * In order to fix PEAR Bug #6389
59 * @var boolean
60 */
61 var $processSpecialRoots = true;
62 /**
63 * Smarty Converter wants elements sorted by type as well as alphabetically
64 * @see Converter::$sort_page_contents_by_type
```

```

65      * @var boolean
66      */
67  var $sort_page_contents_by_type = true;
68  /** @var string */
69  var $outputformat = 'HTML';
70  /** @var string */
71  var $name = 'Smarty';
72  /**
73   * indexes of elements of package that need to be generated
74   * @var array
75   */
76  var $leftindex = array('classes' => true, 'pages' => true, 'functions' => true,
'defines' => false, 'globals' => false);
77
78  /**
79   * output directory for the current procedural page being processed
80   * @var string
81   */
82  var $page_dir;
83
84  /**
85   * target directory passed on the command-line.
86   * {@link $targetDir} is malleable, always adding package/ and package/subpackage/
87   * subdirectories onto it.
88   * @var string
89   */
90  var $base_dir;
91
92  /**
93   * output directory for the current class being processed
94   * @var string
95   */
96  var $class_dir;
97
98  /**
99   * array of converted package page names.
100  * Used to link to the package page in the left index
101  * @var array Format: array(package => 1)
102  */
103  var $package_pages = array();
104
105 /**
106  * controls formatting of parser informative output
107  *
108  * Converter prints:
109  * "Converting /path/to/file.php... Procedural Page Elements... Classes...""
110  * Since HTMLDefaultConverter outputs files while converting, it needs to send a \n to
111  * start a new line. However, if there
112  * is more than one class, output is messy, with multiple \n's just between class file
113  * output. This variable prevents that
114  * and is purely cosmetic
115  * @var boolean
116  */
117  var $juststarted = false;
118
119 /**
120  * contains all of the template procedural page element loop data needed for the current
121  * template
122  * @var array
123  */
124  var $current;
125
126 /**
127  * contains all of the template class element loop data needed for the current template
128  * @var array
129  */
130  var $currentclass;
131  var $wrote = false;
132  var $ric_set = array();
133
134 /**
135  * sets {@link $base_dir} to $targetDir
136  * @see Converter()
137  */
138 /**
139  * @access private
140  */
141  var $_classleft_cache = false;
142  var $_classcontents_cache = false;

```

```

140     var $_pagecontents_cache = false;
141     var $_pageleft_cache = false;
142     var $_done_package_index = false;
143     var $_ric_done = false;
144     var $_wrote_tdir = false;
145     var $ric_contents = array();
146     /**#@-*/
147
148     function HTMLSmartyConverter(& $allp, & $packp, & $classes, & $procpages, $po,
149     $pp, $qm, $targetDir, $templateDir, $title)
150     {
151         Converter::Converter($allp, $packp, $classes, $procpages, $po, $pp, $qm, $targetDir,
152         $templateDir, $title);
153         $this-> base_dir = $targetDir;
154     }
155
156     function writeSource($path, $value)
157     {
158         $templ = & $this-> newSmarty();
159         $pathinfo = $this-> proceduralpages-> getPathInfo($path, $this);
160         $templ-> assign('source', $value);
161         $templ-> assign('package', $pathinfo['package']);
162         $templ-> assign('subpackage', $pathinfo['subpackage']);
163         $templ-> assign('name', $pathinfo['name']);
164         $templ-> assign('source_loc', $pathinfo['source_loc']);
165         $templ-> assign('docs', $pathinfo['docs']);
166         $templ-> assign("subdir", ".../");
167         $templ-> register_outputfilter('HTMLSmarty_outputfilter');
168         $this-> setTargetDir($this-> getFileSourcePath($this-> base_dir));
169         phpDocumentor_out("\n");
170         $this-> setSourcePaths($path);
171         $this-> writefile($this-> getFileSourceName($path) . '.html', $templ-
> fetch('filesource.tpl'));
172     }
173
174     function writeExample($title, $path, $source)
175     {
176         $templ = & $this-> newSmarty();
177         $templ-> assign('source', $source);
178         if (empty($title))
179         {
180             $title = 'example';
181             addWarning(PDERROR_EMPTY_EXAMPLE_TITLE, $path, $title);
182         }
183         $templ-> assign('title', $title);
184         $templ-> assign('file', $path);
185         $templ-> assign("subdir", ".../");
186         $templ-> register_outputfilter('HTMLSmarty_outputfilter');
187         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . '__examplesource');
188         phpDocumentor_out("\n");
189         $this-> writefile('exsource_' . $path . '.html', $templ-> fetch('examplesource.tpl'));
190     }
191
192     function getExampleLink($path, $title)
193     {
194         return $this-> returnLink('{$subdir}__examplesource' . PATH_DELIMITER .
195         '__exsource_' . $path . '.html', $title);
196     }
197
198     function getSourceLink($path)
199     {
200         return $this-> returnLink('{$subdir}__filesource/' .
201         $this-> getFileSourceName($path) . '.html', 'Source Code for this file');
202     }
203
204     /**
205      * Retrieve a Converter-specific anchor to a segment of a source code file
206      * parsed via a {@tutorial tags.filesource.pkg} tag.
207      * @param string full path to source file
208      * @param string name of anchor
209      * @param string link text, if this is a link
210      * @param boolean returns either a link or a destination based on this
211      *          parameter
212      * @return string link to an anchor, or the anchor
213      */
214     function getSourceAnchor($sourcefile, $anchor, $text = '', $link = false)
215     {
216         if ($link) {
217             return $this-> returnLink('{$subdir}__filesource/' .
218             $this-> getFileSourceName($sourcefile) . '.html#a' . $anchor, $text);
219     }

```

```

216     } else {
217         return '<a name="a"           .' . $anchor . '"></a>' ;
218     }
219 }
220 /**
221 * Return a line of highlighted source code with formatted line number
222 *
223 * If the $path is a full path, then an anchor to the line number will be
224 * added as well
225 * @param integer line number
226 * @param string highlighted source code line
227 * @param false|string full path to @filesource file this line is a part of,
228 * if this is a single line from a complete file.
229 * @return string formatted source code line with line number
230 */
231 function sourceLine($linenumber, $line, $path = false)
232 {
233     $extra = '';
234     if (strlen(str_replace("\n" , ' ', $line)) == 0) {
235         $extra = ' ';
236     }
237     if ($path)
238     {
239         return '<li><div class="src-line">' . $this->
240         getSourceAnchor($path, $linenumber) .
241             str_replace("\n" , ' ', $line) . $extra .
242             "</div></li>\n";
243     }
244     else
245     {
246         return '<li><div class="src-line">' .
247             str_replace("\n" , ' ', $line) . " " . $extra . '</div></li>\n';
248     }
249 /**
250 * Used to convert the <<code>> tag in a docblock
251 * @param string
252 * @param boolean
253 * @return string
254 */
255 function ProgramExample($example, $tutorial = false, $inlinesourceparse = null/*false*/,
256                         $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
257 null/*false*/)
258 {
259     $trans = $this-> template_options['desctranslate'];
260     $this-> template_options['desctranslate'] = array();
261     $example = '<ol>' . parent::ProgramExample($example, $tutorial,
262     $inlinesourceparse, $class, $linenum, $filesourcepath)
263     . '</ol>';
264     $this-> template_options['desctranslate'] = $trans;
265     if (!isset($this-> template_options['desctranslate'])) return $example;
266     if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
267     $example = $this-> template_options['desctranslate'][['code']] . $example;
268     if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
269     return $example . $this-> template_options['desctranslate'][['code']];
270 }
271 /**
272 * @param string
273 */
274 function TutorialExample($example)
275 {
276     $trans = $this-> template_options['desctranslate'];
277     $this-> template_options['desctranslate'] = array();
278     $example = '<ol>' . parent::TutorialExample($example)
279     . '</ol>';
280     $this-> template_options['desctranslate'] = $trans;
281     if (!isset($this-> template_options['desctranslate'])) return $example;
282     if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
283     $example = $this-> template_options['desctranslate'][['code']] . $example;
284     if (!isset($this-> template_options['desctranslate'][['code']])) return $example;
285     return $example . $this-> template_options['desctranslate'][['code']];
286 }
287 function getCurrentPageLink()
288 {
289     return $this-> curname . '.html';
290 }
291

```

```

292     function unmangle($sourcecode)
293     {
294         $sourcecode = str_replace('&nbsp;', ' ', $sourcecode);
295         $sourcecode = str_replace('&amp;', '&', $sourcecode);
296         $sourcecode = str_replace('<br />', "<br>", $sourcecode);
297         $sourcecode = str_replace('<code>', '<pre>', $sourcecode);
298         $sourcecode = str_replace('</code>', '</pre>', $sourcecode);
299         $sourcecode = str_replace('&lt;', '<', $sourcecode);
300         $sourcecode = str_replace('&gt;', '>', $sourcecode);
301         return $sourcecode;
302     }
303
304     /**
305      * Uses htmlspecialchars() on the input
306     */
307     function postProcess($text)
308     {
309         if ($this-> highlightingSource) {
310             return str_replace(array(' ', "\t"),
311                               array('&nbsp;&nbsp;', ),
312                               htmlspecialchars($text));
313         }
314         return htmlspecialchars($text);
315     }
316
317     /**
318      * Use the template tutorial_toc.tpl to generate a table of contents for HTML
319      * @return string table of contents formatted for use in the current output format
320      * @param array format: array(array('tagname' => section, 'link' => returnsee link,
321      * 'id' => anchor name, 'title' => from title tag),...)
322     */
323     function formatTutorialTOC($toc)
324     {
325         $template = & $this-> newSmarty();
326         $template-> assign('toc', $toc);
327         return $template-> fetch('tutorial_toc.tpl');
328     }
329
330     function & SmartyInit(& $templ)
331     {
332         $this-> makeLeft();
333         $templ-> assign("ric" , $this-> ric_set);
334         $templ-> assign("packageindex" , $this-> package_index);
335         $templ-> assign('hastodos', count($this-> todolist));
336         $templ-> assign('todolink' , 'todolist.html');
337         $templ-> assign("subdir" , '');
338         return $templ;
339     }
340
341     /**
342      * Writes out the template file of {@link $class_data} and unsets the template to save
343      * memory
344      * @see registerCurrentClass()
345      * @see parent::endClass()
346     */
347     function endClass()
348     {
349         $a = '../';
350         if (!empty($this-> subpackage)) $a .= '../';
351         if ($this-> juststarted)
352         {
353             $this-> juststarted = false;
354             phpDocumentor_out("\n");
355             flush();
356         }
357         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> class_dir);
358         $classleft = $this-> getClassLeft();
359         $this-> class_data-> assign("compiledfileindex" , $this-> getPageLeft());
360         $this-> class_data-> assign("compiledclassindex" , $classleft['class']);
361         $this-> class_data-> assign("compiledinterfaceindex" , $classleft['interface']);
362         $this-> class_data-> assign("tutorials" , $this-> getTutorialList());
363         $this-> class_data-> assign("contents" , $this-> getClassContents());
364         $this-> class_data-> assign("packageindex" , $this-> package_index);
365         $this-> class_data-> assign("package" , $this-> package);
366         $this-> class_data-> assign("subdir" , $a);
367         $this-> class_data-> register_outputfilter('HTMLSmarty_outputfilter');
368         $this-> writefile($this-> class . '.html', $this-> class_data-
369     }
370     fetch('class.tpl'));
371     unset($this-> class_data);

```

```

367     }
368
369     function getTutorialList()
370     {
371         static $cache = false;
372         if ($cache)
373         {
374             if (isset($cache[$this-> package])) return $cache[$this-> package];
375         }
376         $package = $this-> package;
377         if (!isset($this-> tutorials[$package])) return false;
378         foreach($this-> tutorials[$package] as $subpackage => $blah)
379         {
380             $subpackages[] = $subpackage;
381         }
382         $tutes = array();
383         foreach($subpackages as $subpackage)
384         {
385             if (isset($this-> tutorial_tree) && is_array($this-> tutorial_tree))
386                 foreach($this-> tutorial_tree as $root => $tr)
387                 {
388                     if ($tr['tutorial']-> package == $package && $tr['tutorial']-> subpackage == $subpackage)
389                         $tutes[$tr['tutorial']-> tutorial_type][] = $this-> getTutorialTree($tr['tutorial']);
390                 }
391             $cache[$this-> package] = $tutes;
392             return $tutes;
393         }
394     }
395
396     function getTutorialTree($tutorial,$k = false)
397     {
398         $ret = '';
399         if (is_object($tutorial)) $tree = parent::getTutorialTree($tutorial); else $tree =
400 $tutorial;
401         if (!$tree)
402         {
403             $template = & $this-> newSmarty();
404             $template-> assign('subtree',false);
405             $template-> assign('name',str_replace('.','', $tutorial-> name));
406             $template-> assign('parent',false);
407             $template-> assign('haskids',false);
408             $template-> assign('kids','');
409             $link = new tutorialLink;
410             $t = $tutorial;
411             $link-> addLink('', $t-> path, $t-> name, $t-> package, $t-> subpackage, $t->
412 getTitle($this));
413             $main = array('link' => $this-> getId($link), 'title' => $link-> title);
414             $template-> assign('main',$main);
415             return $template-> fetch('tutorial_tree.tpl');
416         }
417         if (isset($tree['kids']))
418         {
419             foreach($tree['kids'] as $subtree)
420             {
421                 $ret .= $this-> getTutorialTree($subtree, true);
422             }
423             $template = & $this-> newSmarty();
424             $template-> assign('subtree',$k);
425             $template-> assign('name',str_replace('.','', $tree['tutorial']-> name));
426             $template-> assign('parent',($k ? str_replace('.','', $tree['tutorial']-> parent-
427 name) : false));
428             $template-> assign('haskids',strlen($ret));
429             $template-> assign('kids',$ret);
430             $link = new tutorialLink;
431             $t = $tree['tutorial'];
432             $link-> addLink('', $t-> path, $t-> name, $t-> package, $t-> subpackage, $t->
433 getTitle($this));
434             $main = array('link' => $this-> getId($link), 'title' => $link-> title);
435             $template-> assign('main',$main);
436             return $template-> fetch('tutorial_tree.tpl');
437         }
438         function getClassLeft()
439         {
440             if ($this-> _classleft_cache)
441             {
442                 if (isset($this-> _classleft_cache[$this-> package][$this-> subpackage]))
443

```

```

return $this-> _classleft_cache[$this-> package][$this-> subpackage];
441     }
442     $arr = $classarr = $interfacearr = array();
443     if (isset($this-> left['#class'][$this-> package]))
444     foreach($this-> left['#class'][$this-> package] as $subpackage => $pages)
445     {
446         for ($i = 0; $i < count($pages); $i++) {
447             if ($pages[$i]['is_interface']) {
448                 $interfacearr[$subpackage][] = $pages[$i];
449             } else {
450                 $classarr[$subpackage][] = $pages[$i];
451             }
452         }
453     }
454     $templ = & $this-> newSmarty();
455     $templ-> assign('classleftindex',$classarr);
456     $classarr = $templ-> fetch('classleft.tpl');
457     $this-> _classleft_cache[$this-> package][$this-> subpackage]['class'] =
458     $classarr;
459     $templ = & $this-> newSmarty();
460     $templ-> assign('classleftindex',$interfacearr);
461     $interfacearr = $templ-> fetch('classleft.tpl');
462     $this-> _classleft_cache[$this-> package][$this-> subpackage]['interface'] =
463     $interfacearr;
464     return $this-> _classleft_cache[$this-> package][$this-> subpackage];
465 }
466 function getClassContents()
467 {
468     if ($this-> _classcontents_cache)
469     {
470         if (isset($this-> _classcontents_cache[$this-> package][$this-
471 > subpackage][$this-> class])) return $this-> _classcontents_cache[$this-> package][$this-
472 > subpackage][$this-> class];
473         $arr = array();
474         foreach($this-> class_contents[$this-> package][$this-> subpackage][$this-
475 > class] as $i => $link)
476         {
477             if (is_object($link))
478                 $arr[$link-> type][] = $this-> returnSee($link,$link-> name);
479         }
480         $this-> _classcontents_cache[$this-> package][$this-> subpackage][$this-
481 > class] = $arr;
482         return $arr;
483     }
484     function getPageContents()
485     {
486         if (!isset($this-> path)) $this-> path = '#####';
487         if ($this-> _pagecontents_cache)
488         {
489             if (isset($this-> _pagecontents_cache[$this-> package][$this-
490 > subpackage][$this-> path])) return $this-> _pagecontents_cache[$this-> package][$this-
491 > subpackage][$this-> path];
492             $arr = array();
493             foreach($this-> page_contents[$this-> curpage-> package][$this-> curpage-
494 > subpackage] as $i => $link)
495             {
496                 if (is_object($link))
497                     $arr[$link-> type][$i] = $this-> returnSee($link);
498             }
499             $this-> _pagecontents_cache[$this-> package][$this-> subpackage][$this-> path]
500 = $arr;
501             return $arr;
502     }
503     function getPageLeft()
504     {
505         if ($this-> _pageleft_cache)
506         {
507             if (isset($this-> _pageleft_cache[$this-> package][$this-> subpackage]))
508             return $this-> _pageleft_cache[$this-> package][$this-> subpackage];
509         }
510         $arr = array();
511         if (isset($this-> left[$this-> package]))
512         foreach($this-> left[$this-> package] as $subpackage => $pages)
513         {
514             $arr[$subpackage] = $pages;

```

```

510 }
511 $templ = & $this-> newSmarty();
512 $templ-> assign('fileleftindex',$arr);
513 $arr = $templ-> fetch('fileleft.tpl');
514 $this-> _pageleft_cache[$this-> package][$this-> subpackage] = $arr;
515 return $arr;
516 }
517
518 /**
519 * Writes out the template file of {@link $page_data} and unsets the template to save memory
520 * @see registerCurrent()
521 * @see parent::endPage()
522 */
523 function endPage()
524 {
525     $this-> package = $this-> curpage-> package;
526     $this-> subpackage = $this-> curpage-> subpackage;
527     $a = '../';
528     if (!empty($this-> subpackage)) $a .= '../';
529     $classleft = $this-> getClassLeft();
530     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $this-> page_dir);
531     $this-> page_data-> assign("contents" , $this-> getPageContents());
532     $this-> page_data-> assign("compiledfileindex" , $this-> getPageLeft());
533     $this-> page_data-> assign("compiledclassindex" , $classleft['class']);
534     $this-> page_data-
> assign("compiledinterfaceindex" , $classleft['interface']);
535     $this-> page_data-> assign("tutorials" , $this-> getTutorialList());
536     $this-> page_data-> assign("packageindex" , $this-> package_index);
537     $this-> page_data-> assign("package" , $this-> package);
538     $this-> page_data-> assign("subdir" , $a);
539     $this-> page_data-> register_outputfilter('HTMLSmarty_outputfilter');
540     $this-> writefile($this-> page . '.html' , $this-> page_data-> fetch('page.tpl'));
541     unset($this-> page_data);
542 }
543
544 /**
545 * @param string
546 * @param string
547 * @return string <a href='".$link."'>' . $text . '</a>';
548 */
549 function returnLink($link,$text)
550 {
551     return '<a href="" . $link. '">' . $text . '</a>';
552 }
553
554 function makeLeft()
555 {
556     if ($this-> _done_package_index) return;
557     $this-> _done_package_index = true;
558     if (!isset($this-> package_index))
559     foreach($this-> all_packages as $key => $val)
560     {
561         if (isset($this-> pkg_elements[$key]))
562         {
563             if (!isset($start)) $start = $key;
564             $this-> package_index[] = array('link' => " li_$key.html",
565 'title' => $key);
566         }
567         foreach($this-> page_elements as $package => $o1)
568         {
569             foreach($o1 as $subpackage => $links)
570             {
571                 for($i=0;$i< count($links);$i++)
572                 {
573                     $this-> left[$package][$subpackage][] =
574                         array("link" => $this-> getId($links[$i]),
575 => $links[$i]-> name);
576                 }
577             }
578             foreach($this-> class_elements as $package => $o1)
579             {
580                 foreach($o1 as $subpackage => $links)
581                 {
582                     for($i=0;$i< count($links);$i++)
583                     {
584                         $isinterface = false;
585                         if ($links[$i]-> type == 'class') {
586                             $class = $this-> classes-> getClass($links[$i]-> name,

```

```

587                     $links[$i]-> path);
588                     if ($class) {
589                         $isinterface = $class-> isInterface();
590                     }
591                 }
592             => $this-> left['#class'][$package][$subpackage][] =
593                 array("link" => $this-> getId($links[$i]),
594 "title" => $links[$i]-> name, 'is_interface' => $isinterface),
595         }
596     }
597 }
598 /**
599  * HTMLDefaultConverter chooses to format both package indexes and the complete index here
600  *
601  * This function formats output for the elementindex.html and pkgelementindex.html template
602  * files. It then
603  * writes them to the target directory
604  * @see generateElementIndex(), generatePkgElementIndex()
605  */
606 function formatPkgIndex()
607 {
608     list($package_indexes, $packages, $mletters) = $this-> generatePkgElementIndexes();
609     for($i=0;$i< count($package_indexes);$i++)
610     {
611         $template = & $this-> newSmarty();
612         $this-> package = $package_indexes[$i]['package'];
613         $this-> subpackage = '';
614         $classleft = $this-> getClassLeft();
615         $template-> assign("compiledfileindex" , $this-> getPageLeft());
616         $template-> assign("compiledclassindex" , $classleft['class']);
617         $template-> assign("compiledinterfaceindex" , $classleft['interface']);
618         $template-> assign("tutorials" , $this-> getTutorialList());
619         $template-> assign("index" , $package_indexes[$i]['pindex']);
620         $template-> assign("package" , $package_indexes[$i]['package']);
621         $template-
622 > assign("letters" , $mletters[$package_indexes[$i]['package']]);
623         $template-> assign("title" , "Package
" . $package_indexes[$i]['package'] . " Element Index" );
624         $template-> assign("date" , date("r" , time()));
625         $template-> register_outputfilter('HTMLSmarty_outputfilter');
626         $this-> setTargetDir($this-> base_dir);
627         $this-
628 > writefile('elementindex_' . $package_indexes[$i]['package'] . '.html' , $template-
629 > fetch('pkgelementindex.tpl'));
630     }
631 /**
632  * HTMLDefaultConverter uses this function to format template index.html and packages.html
633  *
634  * This function generates the package list from {@link $all_packages}, eliminating any
635  * packages that don't have any entries in their package index (no files at all, due to
636  * @ignore
637  * or other factors). Then it uses the default package name as the first package index to
638  * display.
639  * It sets the right pane to be either a blank file with instructions on making package-
640  * level docs,
641  * or the package-level docs for the default package.
642  * @global string Used to set the starting package to display
643  */
644 function formatIndex()
645 {
646     global $phpDocumentor_DefaultPackageName;
647     if (!isset($this-> package_index))
648     {
649         debug("\nERROR: Nothing parsed, check the command-line");
650         die();
651     }
652     list($elindex, $mletters) = $this-> generateElementIndex();
653     $template = & $this-> newSmarty();
654     $template-> assign("index" , $elindex);
655     $template-> assign("letters" , $mletters);
656     $template-> assign("title" , "Element Index" );
657     $template-> assign("package" , false);
658     $template-> assign("date" , date("r" , time()));
659     $phpDocumentor_out("\n");

```

```

658     flush();
659     $this-> setTargetDir($this-> base_dir);
660     $template-> register_outputfilter('HTMLSmarty_outputfilter');
661     $this-> writefile('elementindex.html',$template-> fetch('elementindex.tpl'));
662     usort($this-> package_index,"HTMLSmarty_pindexcmp" );
663     $index = & $this-> newSmarty();
664     foreach($this-> all_packages as $key => $val)
665     {
666         if (isset($this-> pkg_elements[$key]))
667         {
668             if (!isset($start)) $start = $key;
669             if (!isset($this-> package_pages[$key])) $this-> writeNewPPage($key);
670         }
671     }
672     // Created index.html
673     $start = $phpDocumentor_DefaultPackageName;
674     if (!isset($this-> pkg_elements[$key]))
675     {
676         // if there are no elements, use a random package as the default
677         $a = array_keys($this-> pkg_elements);
678         $start = array_shift($a);
679     }
680     $this-> package = $start;
681     $this-> subpackage = '';
682     $classleft = $this-> getClassLeft();
683     $index-> assign("compiledfileindex" , $this-> getPageLeft());
684     $index-> assign("compiledclassindex" , $classleft['class']);
685     $index-> assign("compiledinterfaceindex" , $classleft['interface']);
686     $index-> assign('hastodos', count($this-> todoList));
687     $index-> assign('todolink','todolist.html');
688     $index-> assign("tutorials" , $this-> getTutorialList());
689     $index-> assign("date" , date("r" , time()));
690     $index-> assign("package" , $this-> package);
691     $index-> assign("title" , $this-> title);
692     $index-> assign("start" , " li_$start.html" );
693     if (isset($this-> package_pages[$start]))
694     {
695         $index-> assign("contents" , $this-> package_pages[$start]);
696     }
697     $index-> register_outputfilter('HTMLSmarty_outputfilter');
698     phpDocumentor_out("\n");
699     flush();
700     $this-> setTargetDir($this-> base_dir);
701     $this-> writefile("index.html" , $index-> fetch('index.tpl'));
702     unset($index);
703 }
704
705
706 function writeNewPPage($key)
707 {
708     $template = & $this-> newSmarty();
709     $this-> package = $key;
710     $this-> subpackage = '';
711     $classleft = $this-> getClassLeft();
712     $template-> assign("compiledfileindex" , $this-> getPageLeft());
713     $template-> assign("compiledclassindex" , $classleft['class']);
714     $template-> assign("compiledinterfaceindex" , $classleft['interface']);
715     $template-> assign("tutorials" , $this-> getTutorialList());
716     $template-> assign("date" , date("r" , time()));
717     $template-> assign("title" , $this-> title);
718     $template-> assign("package" , $key);
719     $template-> register_outputfilter('HTMLSmarty_outputfilter');
720     phpDocumentor_out("\n");
721     flush();
722     $this-> setTargetDir($this-> base_dir);
723     $this-> writefile(" li_$key.html" , $template-> fetch('index.tpl'));
724     unset($template);
725 }
726
727 /**
728 * Generate indexes for li_package.html and classtree output files
729 *
730 * This function generates the li_package.html files from the template file left.html. It
731 * does this by
732 *   * iterating through each of the $page_elements, $class_elements and $function_elements
733 * arrays to retrieve
734 *   * the pre-sorted {@link abstractLink} descendants needed for index generation. Conversion
735 * of these links to
736 *   * text is done by {@link returnSee()}.
737 */

```

```

735      * Then it uses {@link generateFormattedClassTrees()} to create class trees from the
736      * template file classtrees.html. Output
737      * filename is classtrees_packageName.html. This function also unsets {@link $elements}
738      * and {@link $pkg_elements} to free
739      * up the considerable memory these two class vars use
740      * @see $page_elements, $class_elements, $function_elements
741      */
740      function formatLeftIndex()
741      {
742          phpDocumentor_out("\n");
743          flush();
744          $this-> setTargetDir($this-> base_dir);
745          if (!isset($this-> left))
746          {
747              debug("Nothing parsed, check the command-line");
748              die();
749          }
750          foreach($this-> all_packages as $package => $rest)
751          {
752              if (!isset($this-> pkg_elements[$package])) continue;
753              // Create class tree page
754              $template = & $this-> newSmarty();
755              $classleft = $this-> getClassLeft();
756              $template-> assign("compiledfileindex" , $this-> getPageLeft());
757              $template-> assign("compiledclassindex" , $classleft['class']);
758              $template-> assign("compiledinterfaceindex" , $classleft['interface']);
759              $template-> assign("classtrees" , $this-
> generateFormattedClassTrees($package));
760              $template-> assign("interfaces" , $this-
> generateFormattedInterfaceTrees($package));
761              $template-> assign("package" , $package);
762              $template-> assign("date" , date("r" , time()));
763              $template-> assign("title" , " Class Trees for Package
$package" );
764              $template-> register_outputfilter('HTMLSmarty_outputfilter');
765              $this-> writefile(" classtrees_{$package}.html" , $template-
> fetch('classtrees.tpl'));
766              phpDocumentor_out("\n");
767              flush();
768          }
769          $this-> writeRIC();
770          // free up considerable memory
771          unset($this-> elements);
772          unset($this-> pkg_elements);
773      }

775 /**
776  * This function takes an {@link abstractLink} descendant and returns an html link
777  *
778  * @param abstractLink a descendant of abstractlink should be passed, and never text
779  * @param string text to display in the link
780  * @param boolean this parameter is not used, and is deprecated
781  * @param boolean determines whether the returned text is enclosed in an <a> tag
782  */
783 function returnSee(& $element, $eltext = false, $with_a = true)
784 {
785     if (!is_object($element) || !$element) return false;
786     if (!$with_a) return $this-> getId($element, false);
787     if (!$eltext)
788     {
789         $eltext = '';
790         switch($element-> type)
791         {
792             case 'tutorial' :
793                 $eltext = strip_tags($element-> title);
794                 break;
795             case 'method' :
796             case 'var' :
797             case 'const' :
798                 $eltext .= $element-> class.'::';
799             case 'page' :
800             case 'define' :
801             case 'class' :
802                 $eltext .= $element-> name;
803             case 'function' :
804             case 'global' :
805                 default :
806                     $eltext .= $element-> name;
807                     if ($element-> type == 'function' || $element-> type == 'method') $eltext
.= '()';

```

```

808         break;
809     }
810   }
811   return '<a href="'. $this-> getId($element).'">' . $eltext.'</a>';
812 }
813
814 function getId($element, $fullpath = true)
815 {
816   if (phpDocumentor_get_class($element) == 'parserdata')
817   {
818     $element = $this-> addLink($element-> parent);
819     $elp = $element-> parent;
820   } elseif (is_a($element, 'parserbase'))
821   {
822     $elp = $element;
823     $element = $this-> addLink($element);
824   }
825   $c = '';
826   if (!empty($element-> subpackage))
827   {
828     $c = '/'.$element-> subpackage;
829   }
830   $b = "{$subdir}";
831   switch ($element-> type)
832   {
833     case 'page' :
834       if ($fullpath)
835         return $b.$element-> package.$c.'/'.$element-> fileAlias.'.html';
836       return 'top';
837       break;
838     case 'define' :
839     case 'global' :
840     case 'function' :
841       if ($fullpath)
842         return $b.$element-> package.$c.'/'.$element-> fileAlias.'.html#'.$element-
843 > type.$element-> name;
844       return $element-> type.$element-> name;
845       break;
846     case 'class' :
847       if ($fullpath)
848         return $b.$element-> package.$c.'/'.$element-> name.'.html';
849       return 'top';
850       break;
851     case 'method' :
852     case 'var' :
853     case 'const' :
854       if ($fullpath)
855         return $b.$element-> package.$c.'/'.$element-> class.'.html#'.$element-
856 > type.$element-> name;
857       return $element-> type.$element-> name;
858       break;
859     case 'tutorial' :
860       $d = '';
861       if ($element-> section)
862       {
863         $d = '#'.$element-> section;
864       }
865       return $b.$element-> package.$c.'/tutorial_'.$element-> name.'.html'.$d;
866     }
867   /**
868    * Convert README/INSTALL/CHANGELOG file contents to output format
869    * @param README/INSTALL/CHANGELOG
870    * @param string contents of the file
871    */
872   function Convert_RIC($name, $contents)
873   {
874     $this-> ric_contents[$name] = $contents;
875     $this-> ric_set[] = array('file' => 'ric_'.$name.'.html', 'name' => $name);
876   }
877
878   function writeRIC()
879   {
880     if ($this-> _ric_done) return;
881     $this-> _ric_done = true;
882     foreach($this-> ric_contents as $name => $contents)
883     {
884       $template = & $this-> newSmarty();
885       $template-> assign('contents', $contents);

```

```

886         $template-> assign('name', $name);
887         $template-> assign('title', $name);
888         $this-> setTargetDir($this-> base_dir);
889         $this-> writefile('ric_'.$name . '.html', $template-> fetch('ric.tpl'));
890     }
891 }
892
893 function ConvertTodoList()
894 {
895     $todolist = array();
896     foreach($this-> todoList as $package => $alltodos)
897     {
898         foreach($alltodos as $todos)
899         {
900             $converted = array();
901             $converted['link'] = $this-> returnSee($todos[0]);
902             if (!is_array($todos[1]))
903             {
904                 $converted['todos'][] = $todos[1]-> Convert($this);
905             } else
906             {
907                 foreach($todos[1] as $todo)
908                 {
909                     $converted['todos'][] = $todo-> Convert($this);
910                 }
911             }
912             $todolist[$package][] = $converted;
913         }
914     }
915     $templ = & $this-> newSmarty();
916     $templ-> assign('todos', $todolist);
917     $templ-> register_outputfilter('HTMLSmarty_outputfilter');
918     $this-> setTargetDir($this-> base_dir);
919     $this-> writefile('todolist.html', $templ-> fetch('todolist.tpl'));
920 }
921
922 /**
923 * Create errors.html template file output
924 *
925 * This method takes all parsing errors and warnings and spits them out ordered by file and
926 * line number.
927 * @global ErrorTracker We'll be using it's output facility
928 */
929 function ConvertErrorLog()
930 {
931     global $phpDocumentor_errors;
932     $allfiles = array();
933     $files = array();
934     $warnings = $phpDocumentor_errors-> returnWarnings();
935     $errors = $phpDocumentor_errors-> returnErrors();
936     $template = & $this-> newSmarty();
937     foreach($warnings as $warning)
938     {
939         $file = '##none';
940         $linenum = 'Warning';
941         if ($warning-> file)
942         {
943             $file = $warning-> file;
944             $allfiles[$file] = 1;
945             $linenum .= ' on line ' . $warning-> linenum;
946         }
947         $files[$file]['warnings'][] = array('name' => $linenum, 'listing' => $warning-
948 > data);
949     }
950     foreach($errors as $error)
951     {
952         $file = '##none';
953         $linenum = 'Error';
954         if ($error-> file)
955         {
956             $file = $error-> file;
957             $allfiles[$file] = 1;
958             $linenum .= ' on line ' . $error-> linenum;
959         }
960         $files[$file]['errors'][] = array('name' => $linenum, 'listing' => $error-
961 > data);
962     }
963     $i=1;
964     $af = array();
965     foreach($allfiles as $file => $num)

```

```

963     {
964         $af[$i++] = $file;
965     }
966     $allfiles = $af;
967     usort($allfiles,'strnatcasecmp');
968     $allfiles[0] = "Post-parsing";
969     foreach($allfiles as $i => $a)
970     {
971         $allfiles[$i] = array('file' => $a);
972     }
973     $out = array();
974     foreach($files as $file => $data)
975     {
976         if ($file == '#none') $file = 'Post-parsing';
977         $out[$file] = $data;
978     }
979     $template-> assign("files" , $allfiles);
980     $template-> assign("all" , $out);
981     $template-> assign("title" , "phpDocumentor Parser Errors and
982 Warnings");
983     $template-> register_outputfilter('HTMLSmarty_outputfilter');
984     $this-> setTargetDir($this-> base_dir);
985     $this-> writefile("errors.html" , $template-> fetch('errors.tpl'));
986     unset($template);
987     phpDocumentor_out("\nTo view errors and warnings, look at "
988 > $this-> base_dir. PATH_DELIMITER . "errors.html\n");
989     flush();
990 }
991 function getCDATA($value)
992 {
993     return '<pre>' . htmlentities($value) . '</pre>';
994 }
995 function getTutorialId($package,$subpackage,$tutorial,$id)
996 {
997     return $id;
998 }
999
1000 /**
1001 * Converts package page and sets its package as used in {@link $package_pages}
1002 * @param parserPackagePage
1003 */
1004 function convertPackagepage(& $element)
1005 {
1006     phpDocumentor_out("\n");
1007     flush();
1008     $template = & $this-> newSmarty();
1009     $this-> package = $element-> package;
1010     $this-> subpackage = '';
1011     $classleft = $this-> getClassLeft();
1012     $template-> assign("compiledfileindex" , $this-> getPageLeft());
1013     $template-> assign("compiledclassindex" , $classleft['class']);
1014     $template-> assign("compiledinterfaceindex" , $classleft['interface']);
1015     $template-> assign("tutorials" , $this-> getTutorialList());
1016     $template-> assign("date" , date("r" , time()));
1017     $template-> assign("title" , $this-> title);
1018     $template-> assign("package" , $element-> package);
1019     $x = $element-> Convert($this);
1020     $x = substr($x,strpos($x,'<body' ));
1021     $template-> assign("contents" , trim(substr($x,strpos($x,'>' ) + 1)));
1022     $this-> package_pages[$element-> package] = trim(substr($x,strpos($x,'>' ) + 1));
1023     $template-> register_outputfilter('HTMLSmarty_outputfilter');
1024     phpDocumentor_out("\n");
1025     flush();
1026     $this-> setTargetDir($this-> base_dir);
1027     $this-> writefile("li_" . $element-> package . ".html"
1028 > fetch('index.tpl'));
1029     unset($template);
1030 }
1031 /**
1032 * @param parserTutorial
1033 */
1034 function convertTutorial(& $element)
1035 {
1036     phpDocumentor_out("\n");
1037     flush();
1038     $template = & parent::convertTutorial($element);
1039     $this-> package = $element-> package;

```

```

1040     $this-> subpackage = $element-> subpackage;
1041     $classleft = $this-> getClassLeft();
1042     $template-> assign("compiledfileindex" , $this-> getPageLeft());
1043     $template-> assign("compiledclassindex" , $classleft['class']);
1044     $template-> assign("compiledinterfaceindex" , $classleft['interface']);
1045     $template-> assign("tutorials" , $this-> getTutorialList());
1046     $template-> assign("title" , strip_tags($element-> getTitle($this)));
1047     $contents = $element-> Convert($this);
1048     if ($element-> name == $this-> package . '.pkg')
1049     {
1050         $this-> package_pages[$element-> package] = $contents;
1051     }
1052     $a = './';
1053     if (!empty($element-> subpackage)) $a .= $a;
1054     $template-> assign("subdir" , $a);
1055     $a = '';
1056     if ($element-> subpackage) $a = PATH_DELIMITER . $element-> subpackage;
1057     $template-> register_outputfilter('HTMLSmarty_outputfilter');
1058     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $element-> package . $a);
1059     $this-> writeFile('tutorial_' . $element-> name . '.html' , $template-
> fetch('tutorial.tpl'));
1060     if ($element-> name == $element-> package . '.pkg')
1061     {
1062         phpDocumentor_out("\n");
1063         flush();
1064         // package-level docs
1065         $this-> setTargetDir($this-> base_dir);
1066         $template-> assign("subdir" , '');
1067         $this-> writeFile('li_' . $element-> package . '.html' , $template-
> fetch('tutorial.tpl'));
1068     }
1069     unset($template);
1070 }
1071 /**
1072 * Converts class for template output
1073 * @see prepareDocBlock(), generateChildClassList(), generateFormattedClassTree(),
1074 * getFormattedConflicts()
1075 * @see getFormattedInheritedMethods(), getFormattedInheritedVars()
1076 * @param parserClass
1077 */
1078 function convertClass(& $element)
1079 {
1080     parent::convertClass($element);
1081     $this-> class_dir = $element-> docblock-> package;
1082     if (!empty($element-> docblock-> subpackage)) $this-> class_dir .= PATH_DELIMITER
. $element-> docblock-> subpackage;
1083     $a = './classtrees_';
1084     if ($element-> docblock-> subpackage != '') $a = " ./{$a} ";
1085     $this-> class_data-> assign('subdir' , $a);
1086     $this-> class_data-> assign("title" , "Docs For Class "
. $element-> getName());
1087     $this-> class_data-> assign("page" , $element-> getName() . '.html');
1088 }
1089 /**
1090 * Converts class variables for template output
1091 * @see prepareDocBlock(), getFormattedConflicts()
1092 * @param parserDefine
1093 */
1094 function convertVar(& $element)
1095 {
1096     parent::convertVar($element, array('var_dest' => $this-> getId($element, false)));
1097 }
1098 /**
1099 * Converts class variables for template output
1100 * @see prepareDocBlock(), getFormattedConflicts()
1101 * @param parserDefine
1102 */
1103 function convertConst(& $element)
1104 {
1105     parent::convertConst($element, array('const_dest' => $this-
> getId($element, false)));
1106 }
1107 /**
1108 * Converts class methods for template output
1109 * @see prepareDocBlock(), getFormattedConflicts()
1110 */
1111 /**
1112 * Converts class methods for template output
1113 * @see prepareDocBlock(), getFormattedConflicts()

```

```

1114     * @param parserDefine
1115     */
1116     function convertMethod(& $element)
1117     {
1118         parent::convertMethod($element, array('method_dest' => $this-
1119 > getId($element, false)));
1120     }
1121 
1122     /**
1123      * Converts function for template output
1124      * @see prepareDocBlock(), parserFunction::getFunctionCall(), getFormattedConflicts()
1125      * @param parserFunction
1126      */
1127     function convertFunction(& $element)
1128     {
1129         $funcloc = $this-> getId($this-> addLink($element));
1130         parent::convertFunction($element, array('function_dest' => $this-
1131 > getId($element, false)));
1132     }
1133 
1134     /**
1135      * Converts include elements for template output
1136      * @see prepareDocBlock()
1137      * @param parserInclude
1138      */
1139     function convertInclude(& $element)
1140     {
1141         parent::convertInclude($element, array('include_file' => '_'.strtr($element-
1142 > getValue(), array('"' => '"', "''" => "'", '.' => '_'))));
1143     }
1144 
1145     /**
1146      * Converts defines for template output
1147      * @see prepareDocBlock(), getFormattedConflicts()
1148      * @param parserDefine
1149      */
1150     function convertDefine(& $element)
1151     {
1152         parent::convertDefine($element, array('define_link' => $this-
1153 > getId($element, false)));
1154     }
1155 
1156     /**
1157      * Converts global variables for template output
1158      * @param parserGlobal
1159      * @see prepareDocBlock(), getFormattedConflicts()
1160      */
1161     function convertGlobal(& $element)
1162     {
1163         parent::convertGlobal($element, array('global_link' => $this-
1164 > getId($element, false)));
1165     }
1166 
1167     /**
1168      * converts procedural pages for template output
1169      * @see prepareDocBlock(), getClassessOnPage()
1170      * @param parserData
1171      */
1172     function convertPage(& $element)
1173     {
1174         parent::convertPage($element);
1175         $this-> juststarted = true;
1176         $this-> page_dir = $element-> parent-> package;
1177         if (!empty($element-> parent-> subpackage)) $this-> page_dir .= PATH_DELIMITER .
1178         $element-> parent-> subpackage;
1179         // registering stuff on the template
1180         $a = '../';
1181         if (!empty($element-> docblock-> subpackage)) $a = $a . $a;
1182         $this-> page_data-> assign('subdir', $a);
1183         $this-> page_data-> assign("page" , $this-> getPageName($element) .
1184         '.html');
1185         $this-> page_data-> assign("title" , "Docs for page " . $element-
1186 > parent-> getFile());
1187     }
1188 
1189     function getPageName(& $element)
1190     {
1191         if (phpDocumentor_get_class($element) == 'parserpage') return '_'. $element-
1192 > getName();
1193         return '_'. $element-> parent-> getName();

```

```

1185 }
1186 /**
1187 * returns an array containing the class inheritance tree from the root object to the class
1188 *
1189 * @param parserClass    class variable
1190 * @return array Format: array(root,child,child,child,...,$class)
1191 * @uses parserClass::getParentClassTree()
1192 */
1193
1194
1195 function generateFormattedClassTree($class)
1196 {
1197     $tree = $class-> getParentClassTree($this);
1198     $out = '';
1199     if (count($tree) - 1)
1200     {
1201         $result = array($class-> getName());
1202         $parent = $tree[$class-> getName()];
1203         $distance[] = '';
1204         while ($parent)
1205         {
1206             $x = $parent;
1207             if (is_object($parent))
1208             {
1209                 $subpackage = $parent-> docblock-> subpackage;
1210                 $package = $parent-> docblock-> package;
1211                 $x = $parent;
1212                 $x = $parent-> getLink($this);
1213                 if (!$x) $x = $parent-> getName();
1214             }
1215             $result[] =
1216                 $x;
1217             $distance[] =
1218                 "\n$x|\n"
1219                 "%s--";
1220             if (is_object($parent))
1221                 $parent = $tree[$parent-> getName()];
1222             elseif (isset($tree[$parent]))
1223                 $parent = $tree[$parent];
1224             $nbsp = ' ';
1225             for($i=count($result) - 1;$i>= 0;$i--)
1226             {
1227                 $my_nbsp = '';
1228                 for($j=0;$j< count($result) - $i;$j++) $my_nbsp .= $nbsp;
1229                 $distance[$i] = sprintf($distance[$i],$my_nbsp,$my_nbsp);
1230             }
1231         }
1232         return
array('classes'=> array_reverse($result), 'distance'=> array_reverse($distance));
1233     } else
1234     {
1235         return array('classes'=> $class-> getName(), 'distance'=>array( '' ));
1236     }
1237 }
1238
1239 /** @access private */
1240 function sortVar($a, $b)
1241 {
1242     return strnatcasecmp($a-> getName(),$b-> getName());
1243 }
1244
1245 /** @access private */
1246 function sortMethod($a, $b)
1247 {
1248     if ($a-> isConstructor) return -1;
1249     if ($b-> isConstructor) return 1;
1250     return strnatcasecmp($a-> getName(),$b-> getName());
1251 }
1252
1253 /**
1254 * returns a template-enabled array of class trees
1255 *
1256 * @param string $package package to generate a class tree for
1257 * @see $roots, HTMLConverter::getRootTree()
1258 */
1259 function generateFormattedClassTrees($package)
1260 {
1261     if (!isset($this-> roots['normal'][$package]) &&
1262         !isset($this-> roots['special'][$package])) {
1263         return array();

```

```

1264     }
1265     $trees = array();
1266     if (isset($this-> roots['normal'][$package])) {
1267         $roots = $this-> roots['normal'][$package];
1268         for($i=0;$i< count($roots);$i++)
1269         {
1270             $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1271             if ($root && $root-> isInterface()) {
1272                 continue;
1273             }
1274             $trees[] = array('class' => $roots[$i], 'class_tree' =>
1275 " <ul>\n" . $this-> getRootTree($this-
1276 > getSortedClassTreeFromClass($roots[$i],$package,''),$package). "</ul>\n"
1277         }
1278     if (isset($this-> roots['special'][$package])) {
1279         $roots = $this-> roots['special'][$package];
1280         foreach ($roots as $parent => $classes) {
1281             $thistree = '';
1282             foreach ($classes as $classinfo) {
1283                 $root = $this-> classes-> getClassByPackage($classinfo, $package);
1284                 if ($root && $root-> isInterface()) {
1285                     continue;
1286                 }
1287                 $thistree .=
1288                     $this-> getRootTree(
1289                         $this-> getSortedClassTreeFromClass(
1290                             $classinfo,
1291                             $package,
1292                             ''),
1293                             $package,
1294                             true);
1295             if (!$thistree) {
1296                 continue;
1297             }
1298             $trees[] = array(
1299                 'class' => $parent,
1300                 'class_tree' => "<ul>\n" . $thistree .
1301 " </ul>\n"
1302             );
1303         }
1304     return $trees;
1305 }
1306 /**
1307 * returns a template-enabled array of interface inheritance trees
1308 *
1309 * @param string $package package to generate a class tree for
1310 * @see $roots, HTMLConverter::getRootTree()
1311 */
1312 function generateFormattedInterfaceTrees($package)
1313 {
1314     if (!isset($this-> roots['normal'][$package]) &&
1315         !isset($this-> roots['special'][$package])) {
1316         return array();
1317     }
1318     $trees = array();
1319     if (isset($this-> roots['normal'][$package])) {
1320         $roots = $this-> roots['normal'][$package];
1321         for($i=0;$i< count($roots);$i++)
1322         {
1323             $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1324             if ($root && !$root-> isInterface()) {
1325                 continue;
1326             }
1327             $trees[] = array('class' => $roots[$i], 'class_tree' =>
1328 " <ul>\n" . $this-> getRootTree($this-
1329 > getSortedClassTreeFromClass($roots[$i],$package,''),$package). "</ul>\n"
1330         }
1331     if (isset($this-> roots['special'][$package])) {
1332         $roots = $this-> roots['special'][$package];
1333         foreach ($roots as $parent => $classes) {
1334             $thistree = '';
1335             foreach ($classes as $classinfo) {
1336                 $root = $this-> classes-> getClassByPackage($classinfo, $package);
1337                 if ($root && !$root-> isInterface()) {
1338                     continue;
1339                 }
1340             }
1341             $trees[] = array('class' => $parent, 'class_tree' => "<ul>\n" . $thistree .
1342 " </ul>\n"
1343         }
1344     }
1345     return $trees;
1346 }

```

```

1339     }
1340     $thistree .=
1341     $this-> getRootTree(
1342         $this-> getSortedClassTreeFromClass(
1343             $classinfo,
1344             $package,
1345             ''),
1346             $package,
1347             true);
1348     }
1349     if (!$thistree) {
1350         continue;
1351     }
1352     $trees[] = array(
1353         'class' => $parent,
1354         'class_tree' => "<ul>\n" . $thistree .
1355             );
1356     }
1357 }
1358 return $trees;
1359 }
1360
1361 /**
1362 * return formatted class tree for the Class Trees page
1363 *
1364 * @param array $tree output from {@link getSortedClassTreeFromClass()}
1365 * @param string $package package
1366 * @param boolean $ounknownparent if true, an object's parent will not be checked
1367 * @see Classes:::$definitechild, generateFormattedClassTrees()
1368 * @return string
1369 */
1370 function getRootTree($tree, $package, $noparent = false)
1371 {
1372     if (!$tree) return '';
1373     $my_tree = '';
1374     $cur = '#root';
1375     $lastcur = array(false);
1376     $kids = array();
1377     $dopar = false;
1378     if (!$noparent && $tree[$cur]['parent'])
1379     {
1380         $dopar = true;
1381         if (!is_object($tree[$cur]['parent']))
1382         {
1383             // debug("parent ".$tree[$cur]['parent']."' not found");
1384             $my_tree .= '<li>' . $tree[$cur]['parent'] . '<ul>' ;
1385         }
1386         else
1387         {
1388             // debug("parent ".$this-
1389             >returnSee($tree[$cur]['parent'])." in other package");
1390             $root = $this-> classes-> getClassByPackage($tree[$cur]['parent']-> name,
1391                 $package);
1392             $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['parent']);
1393             if ($tree[$cur]['parent']-> package != $package) $my_tree .= '
1394             <b>(Different package)</b><ul>' ;
1395         }
1396     do
1397     {
1398         fancy_debug($cur,$lastcur,$kids);
1399         if (count($tree[$cur]['children']))
1400         {
1401             // debug("$cur has children");
1402             if (!isset($kids[$cur]))
1403             {
1404                 debug("set $cur kids");
1405                 $kids[$cur] = 1;
1406                 $root = $this-> classes-> getClassByPackage(
1407                     $tree[$cur]['link']-> name,
1408                     $tree[$cur]['link']-> package);
1409                 if ($implements = $root-> getImplements())
1410                 {
1411                     $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link']) .
1412                         ' (implements ';
1413                         foreach ($implements as $i => $interface) {
1414                             if ($i && $i != count($implements) - 1) $my_tree .= ', ';
1415                             if ($link = $this-> getLink('object' . $interface)) {
1416                                 $my_tree .= $this-> returnSee($link);
1417                             } else {
1418                         }
1419                     }
1420                 }
1421             }
1422         }
1423     }
1424     }
1425 }

```

```

1416                     $my_tree .= $interface;
1417                 }
1418             }
1419         } else {
1420             $my_tree .= '>';
1421         }
1422     }
1423     $my_tree .= '<ul>' . "\n" ;
1424 }
1425 array_push($lastcur,$cur);
1426 list($cur) = each($tree[$cur]['children']);
1427 var_dump('listed',$cur);
1428 if ($cur)
1429 {
1430     $cur = $cur['package'] . '#' . $cur['class'];
1431     debug("set cur to child $cur");
1432     $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link']);
1433     continue;
1434 } else
1435 {
1436     debug("end of children for $cur");
1437     $cur = array_pop($lastcur);
1438     $cur = array_pop($lastcur);
1439     $my_tree .= '</ul></li>' . "\n";
1440     if ($dopar && ($cur == '#root' || !$cur)) $my_tree .=
1441     '</ul></li>';
1442 }
1443 }
1444 // debug("$cur has no children");
1445 $my_tree .= '<li>' . $this-
1446 > returnSee($tree[$cur]['link'])."</li>" ;
1447 if ($dopar && ($cur == '#root')) $my_tree .= '</ul></li>';
1448 $cur = array_pop($lastcur);
1449 } while ($cur);
1450 return $my_tree;
1451 }
1452 /**
1453 * Generate indexing information for given element
1454 *
1455 * @param parserElement descendant of parserElement
1456 * @see generateElementIndex()
1457 * @return array
1458 */
1459 function getIndexInformation($elt)
1460 {
1461     $Result['type'] = $elt-> type;
1462     $Result['file_name'] = $elt-> file;
1463     $Result['path'] = $elt-> getPath();
1464
1465     if (isset($elt-> docblock))
1466     {
1467         $Result['description'] = $elt-> docblock-> getSDesc($this);
1468
1469         if ($elt-> docblock-> hasaccess)
1470             $Result['access'] = $elt-> docblock-> tags['access'][0]-
1471             > value;
1472
1473         else
1474             $Result['access'] = 'public';
1475
1476         $Result['abstract'] = isset ($elt-> docblock-> tags['abstract'][0]);
1477     }
1478     else
1479         $Result['description'] = '';
1480
1481     $aa = $Result['description'];
1482     if (!empty($aa)) $aa =
1483     "<br>&nbsp;&nbsp;&nbsp;" . $aa" ;
1484
1485     switch($elt-> type)
1486     {
1487         case 'class':
1488             $Result['name'] = $elt-> getName();
1489             $Result['title'] = 'Class';
1490             $Result['link'] = $this-> getClassLink($elt-> getName(),
1491             $elt-> docblock-> package,
1492             $elt-> getPath(),
1493             $elt-> getName());

```

```

1492     $Result['listing'] = 'in file '.$elt-> file.', class
1493     $aa" ;
1494     break;
1495     case 'define':
1496         $Result['name'] = $elt-> getName();
1497         $Result['title'] = 'Constant';
1498         $Result['link'] = $this-> getDefineLink($elt-> getName(),
1499                                         $elt-> docblock-> package,
1500                                         $elt-> getPath(),
1501                                         $elt-> getName());
1502         $aa" ;
1503     break;
1504     case 'global':
1505         $Result['name'] = $elt-> getName();
1506         $Result['title'] = 'Global';
1507         $Result['link'] = $this-> getGlobalLink($elt-> getName(),
1508                                         $elt-> docblock-> package,
1509                                         $elt-> getPath(),
1510                                         $elt-> getName());
1511         $Result['listing'] = 'in file '.$elt-> file.', global variable
1512         $aa" ;
1513     break;
1514     case 'function':
1515         $Result['name'] = $elt-> getName();
1516         $Result['title'] = 'Function';
1517         $Result['link'] = $this-> getFunctionLink($elt-> getName(),
1518                                         $elt-> docblock-> package,
1519                                         $elt-> getPath(),
1520                                         $elt-> getName().'()');
1521         $Result['listing'] = 'in file '.$elt-> file.', function
1522         $aa" ;
1523     break;
1524     case 'method':
1525         $Result['name'] = $elt-> getName();
1526         $Result['title'] = 'Method';
1527         $Result['link'] = $this-> getMethodLink($elt-> getName(),
1528                                         $elt-> class,
1529                                         $elt-> docblock-> package,
1530                                         $elt-> getPath(),
1531                                         $elt-> class.'::'.$elt-
1532                                         );
1533             if ($elt-> isConstructor)
1534                 $Result['listing'] = 'in file '.$elt-> file.', method
1535             $aa" ;
1536     break;
1537     case 'var':
1538         $Result['name'] = $elt-> getName();
1539         $Result['title'] = 'Variable';
1540         $Result['link'] = $this-> getVarLink($elt-> getName(),
1541                                         $elt-> class,
1542                                         $elt-> docblock-> package,
1543                                         $elt-> getPath(),
1544                                         $elt-> class.'::'.$elt-
1545                                         );
1546         $Result['listing'] = 'in file '.$elt-> file.', variable
1547         $aa" ;
1548     break;
1549     case 'const':
1550         $Result['name'] = $elt-> getName();
1551         $Result['title'] = 'Variable';
1552         $Result['link'] = $this-> getConstLink($elt-> getName(),
1553                                         $elt-> class,
1554                                         $elt-> docblock-> package,
1555                                         $elt-> getPath(),
1556                                         $elt-> class.'::'.$elt-
1557                                         );
1558         $Result['listing'] = 'in file '.$elt-> file.', class constant
1559         $aa" ;
1560     break;
1561     case 'page':
1562         $Result['name'] = $elt-> getFile();
1563         $Result['title'] = 'Page';
1564         $Result['link'] = $this-> getPageLink($elt-> getFile(),
1565                                         $elt-> package,
1566                                         $elt-> getPath(),
1567                                         $elt-> getFile());
1568         $Result['listing'] = 'procedural page '.$Result['link'];

```

```

1561         break;
1562     case 'include':
1563         $Result['name'] = $elt-> getName();
1564         $Result['title'] = 'Include';
1565         $Result['link'] = $elt-> getValue();
1566         $Result['listing'] = 'include ' . $Result['name'];
1567         break;
1568     }
1569 }
1570 return $Result;
1571 }
1572 /**
1573 * Generate alphabetical index of all elements
1574 *
1575 * @see $elements, walk()
1576 */
1577 function generateElementIndex()
1578 {
1579     $elementindex = array();
1580     $letters = array();
1581     $used = array();
1582     foreach($this-> elements as $letter => $nutoh)
1583     {
1584         foreach($this-> elements[$letter] as $i => $yuh)
1585         {
1586             if ($this-> elements[$letter][$i]-> type != 'include')
1587             {
1588                 if (!isset($used[$letter]))
1589                 {
1590                     $letters[]['letter'] = $letter;
1591                     $selindex['letter'] = $letter;
1592                     $used[$letter] = 1;
1593                 }
1594             }
1595             $selindex['index'][] = $this-> getIndexInformation($this-
> elements[$letter][$i]);
1596         }
1597     }
1598     if (isset($selindex['index']))
1599     {
1600         $elementindex[] = $selindex;
1601     } else
1602     {
1603         unset($letters[count($letters) - 1]);
1604     }
1605     $selindex = array();
1606 }
1607 }
1608 return array($elementindex,$letters);
1609 }
1610
1611 function copyMediaRecursively($media,$targetdir,$subdir = '')
1612 {
1613     $versionControlDirectories = array ('CVS', 'media/CVS', 'media\\CVS', '.svn',
1614     'media/.svn', 'media\\.svn');
1615     if (!is_array($media))
1616     {
1617         return;
1618     }
1619     foreach($media as $dir => $files)
1620     {
1621         if ($dir === '/')
1622         {
1623             $this-> copyMediaRecursively($files,$targetdir);
1624         } else
1625         {
1626             if (!is_numeric($dir))
1627             {
1628                 if (in_array($dir, $versionControlDirectories))
1629                 {
1630                     // skip it entirely
1631                 } else
1632                 {
1633                     // create the subdir
1634                     phpDocumentor_out("      creating $targetdir" . PATH_DELIMITER .
1635                     "$dir\n" );
1636                     Converter::setTargetDir($targetdir . PATH_DELIMITER . $dir);
1637                     if (!empty($subdir))
1638                     {
1639                         $subdir .= PATH_DELIMITER;

```

```

1638                     }
1639                     $this-
1640                 }
1641             }
1642         }
1643     }
1644     // copy the file
1645     phpDocumentor_out("      copying $targetdir" . PATH_DELIMITER .
1646     $files['file'] . "\n");
1647 }
1648 }
1649 }
1650 }
1651 /**
1652 * calls the converter setTargetDir, and then copies any template images and the stylesheet
1653 if they haven't been copied
1654 * @see Converter::setTargetDir()
1655 */
1656 function setTargetDir($dir)
1657 {
1658     Converter::setTargetDir($dir);
1659     if ($this-> _wrote_tdir) return;
1660     $this-> _wrote_tdir = true;
1661     $template_images = array();
1662     $stylesheets = array();
1663     $tdir = $dir;
1664     $dir = $this-> templateDir;
1665     $this-> templateDir = $this-> templateDir.'templates/';
1666     $info = new Io;
1667     $this-> copyMediaRecursively($info-> getDirTree($this-> templateDir.'media'),$this-
1668 > templateDir),$tdir);
1669 }
1670 /**
1671 * Generate alphabetical index of all elements by package and subpackage
1672 *
1673 * @param string $package name of a package
1674 * @see $pkg_elements, walk(), generatePkgElementIndexes()
1675 */
1676 function generatePkgElementIndex($package)
1677 {
1678     $elementindex = array();
1679     $letters = array();
1680     $letterind = array();
1681     $used = array();
1682     $subp = '';
1683     foreach($this-> pkg_elements[$package] as $subpackage => $els)
1684     {
1685         if (empty($els)) continue;
1686         if (!empty($subpackage)) $subp = "      <b>subpackage:</b>
1687 $subpackage" ;
1688         else $subp = '';
1689         foreach($els as $letter => $yuh)
1690         {
1691             foreach($els[$letter] as $i => $yuh)
1692             {
1693                 if ($els[$letter][$i]-> type != 'include')
1694                 {
1695                     if (!isset($used[$letter]))
1696                     {
1697                         $letters[]['letter'] = $letter;
1698                         $letterind[$letter] = count($letters) - 1;
1699                         $used[$letter] = 1;
1700                     }
1701                     $selindex[$letter]['letter'] = $letter;
1702                     $selindex[$letter]['index'][] = $this-
1703 > getIndexInformation($els[$letter][$i]);
1704                 }
1705             }
1706             ksort($selindex);
1707             usort($letters,'HTMLSmarty_lettersort');
1708             if (isset($selindex))
1709             {
1710                 while(list($letter,$tempel) = each($selindex))
1711                 {

```

```

1712         if (!isset($tempel))
1713         {
1714             unset($letters[$letterind[$tempel['letter']]]);
1715         } else
1716             $elementindex[] = $tempel;
1717     }
1718 } else $letters = array();
1719 return array($elementindex,$letters);
1720 }
1721 /**
1722 *
1723 * @see generatePkgElementIndex()
1724 */
1725 function generatePkgElementIndexes()
1726 {
1727     $packages = array();
1728     $package_names = array();
1729     $pkg = array();
1730     $letters = array();
1731     foreach($this-> pkg_elements as $package => $trash)
1732     {
1733         $pkgs['package'] = $package;
1734         $pkg['package'] = $package;
1735         list($pkg['pindex'],$letters[$package]) = $this-
1736 > generatePkgElementIndex($package);
1737         if (count($pkg['pindex']))
1738         {
1739             $packages[] = $pkg;
1740             $package_names[] = $pkgs;
1741         }
1742         unset($pkgs);
1743         unset($pkg);
1744     }
1745     foreach($packages as $i => $package)
1746     {
1747         $pnames = array();
1748         for($j=0;$j< count($package_names);$j++)
1749         {
1750             if ($package_names[$j]['package'] != $package['package']) $pnames[] =
1751             $package_names[$j];
1752         }
1753         $packages[$i]['packageindexes'] = $pnames;
1754     }
1755     return array($packages,$package_names,$letters);
1756 }
1757 /**
1758 * @param string name of class
1759 * @param string package name
1760 * @param string full path to look in (used in index generation)
1761 * @param boolean deprecated
1762 * @param boolean return just the URL, or enclose it in an html a tag
1763 * @return mixed false if not found, or an html a link to the class's documentation
1764 * @see parent::getClassLink()
1765 */
1766 function getClassLink($expr,$package, $file = false,$text = false, $with_a = true)
1767 {
1768     $a = Converter::getClassLink($expr,$package,$file);
1769     if (!$a) return false;
1770     return $this-> returnSee($a, $text, $with_a);
1771 }
1772 /**
1773 * @param string name of function
1774 * @param string package name
1775 * @param string full path to look in (used in index generation)
1776 * @param boolean deprecated
1777 * @param boolean return just the URL, or enclose it in an html a tag
1778 * @return mixed false if not found, or an html a link to the function's documentation
1779 * @see parent::getFunctionLink()
1780 */
1781 function getFunctionLink($expr,$package, $file = false,$text = false)
1782 {
1783     $a = Converter::getFunctionLink($expr,$package,$file);
1784     if (!$a) return false;
1785     return $this-> returnSee($a, $text);
1786 }
1787
1788 /**

```

```

1790 * @param string name of define
1791 * @param string package name
1792 * @param string full path to look in (used in index generation)
1793 * @param boolean deprecated
1794 * @param boolean return just the URL, or enclose it in an html a tag
1795 * @return mixed false if not found, or an html a link to the define's documentation
1796 * @see parent::getDefineLink()
1797 */
1798 function getDefineLink($expr,$package, $file = false,$text = false)
1799 {
1800     $a = Converter::getDefineLink($expr,$package,$file);
1801     if (!$a) return false;
1802     return $this->  returnSee($a, $text);
1803 }
1804
1805 /**
1806 * @param string name of global variable
1807 * @param string package name
1808 * @param string full path to look in (used in index generation)
1809 * @param boolean deprecated
1810 * @param boolean return just the URL, or enclose it in an html a tag
1811 * @return mixed false if not found, or an html a link to the global variable's
1812 documentation
1813 * @see parent::getGlobalLink()
1814 */
1815 function getGlobalLink($expr,$package, $file = false,$text = false)
1816 {
1817     $a = Converter::getGlobalLink($expr,$package,$file);
1818     if (!$a) return false;
1819     return $this->  returnSee($a, $text);
1820 }
1821
1822 /**
1823 * @param string name of procedural page
1824 * @param string package name
1825 * @param string full path to look in (used in index generation)
1826 * @param boolean deprecated
1827 * @param boolean return just the URL, or enclose it in an html a tag
1828 * @return mixed false if not found, or an html a link to the procedural page's
1829 documentation
1830 * @see parent::getPageLink()
1831 */
1832 function getPageLink($expr,$package, $path = false,$text = false)
1833 {
1834     $a = Converter::getPageLink($expr,$package,$path);
1835     if (!$a) return false;
1836     return $this->  returnSee($a, $text);
1837 }
1838
1839 /**
1840 * @param string name of method
1841 * @param string class containing method
1842 * @param string package name
1843 * @param string full path to look in (used in index generation)
1844 * @param boolean deprecated
1845 * @param boolean return just the URL, or enclose it in an html a tag
1846 * @return mixed false if not found, or an html a link to the method's documentation
1847 * @see parent::getMethodLink()
1848 */
1849 function getMethodLink($expr,$class,$package, $file = false,$text = false)
1850 {
1851     $a = Converter::getMethodLink($expr,$class,$package,$file);
1852     if (!$a) return false;
1853     return $this->  returnSee($a, $text);
1854 }
1855
1856 /**
1857 * @param string name of var
1858 * @param string class containing var
1859 * @param string package name
1860 * @param string full path to look in (used in index generation)
1861 * @param boolean deprecated
1862 * @param boolean return just the URL, or enclose it in an html a tag
1863 * @return mixed false if not found, or an html a link to the var's documentation
1864 * @see parent::getVarLink()
1865 */
1866 function getVarLink($expr,$class,$package, $file = false,$text = false)
1867 {
1868     $a = Converter::getVarLink($expr,$class,$package,$file);
1869     if (!$a) return false;

```

```

1868     return $this->  returnSee($a, $text);
1869 }
1870
1871 /**
1872 * @param string name of class constant
1873 * @param string class containing class constant
1874 * @param string package name
1875 * @param string full path to look in (used in index generation)
1876 * @param boolean deprecated
1877 * @param boolean return just the URL, or enclose it in an html a tag
1878 * @return mixed false if not found, or an html a link to the var's documentation
1879 * @see parent::getVarLink()
1880 */
1881 function getConstLink($expr,$class,$package, $file = false,$text = false)
1882 {
1883     $a = Converter::getConstLink($expr,$class,$package,$file);
1884     if (!$a) return false;
1885     return $this->  returnSee($a, $text);
1886 }
1887
1888 /**
1889 * does a nat case sort on the specified second level value of the array
1890 *
1891 * @param mixed $a
1892 * @param mixed $b
1893 * @return int
1894 */
1895 function rcNatCmp ($a, $b)
1896 {
1897     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1898     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1899
1900     return strnatcasecmp($aa, $bb);
1901 }
1902
1903 /**
1904 * does a nat case sort on the specified second level value of the array.
1905 * this one puts constructors first
1906 *
1907 * @param mixed $a
1908 * @param mixed $b
1909 * @return int
1910 */
1911 function rcNatCmpl ($a, $b)
1912 {
1913     $aa = strtoupper($a[$this->  rcnatcmpkey]);
1914     $bb = strtoupper($b[$this->  rcnatcmpkey]);
1915
1916     if (strpos($aa,'CONSTRUCTOR') === 0)
1917     {
1918         return -1;
1919     }
1920     if (strpos($bb,'CONSTRUCTOR') === 0)
1921     {
1922         return 1;
1923     }
1924     if (strpos($aa,strtoupper($this->  class)) === 0)
1925     {
1926         return -1;
1927     }
1928     if (strpos($bb,strtoupper($this->  class)) === 0)
1929     {
1930         return -1;
1931     }
1932     return strnatcasecmp($aa, $bb);
1933 }
1934
1935 /**
1936 * This function is not used by HTMLdefaultConverter, but is required by Converter
1937 */
1938 function Output()
1939 {
1940 }
1941 }
1942
1943 /**
1944 * @access private
1945 * @global string name of the package to set as the first package
1946 */
1947 function HTMLSmarty_pindexcmp($a, $b)

```

```
1948 {
1949     global $phpDocumentor_DefaultPackageName;
1950     if ($a['title'] == $phpDocumentor_DefaultPackageName) return -1;
1951     if ($b['title'] == $phpDocumentor_DefaultPackageName) return 1;
1952     return strnatcasecmp($a['title'], $b['title']);
1953 }
1954
1955 /** @access private */
1956 function HTMLSmarty_lettersort($a, $b)
1957 {
1958     return strnatcasecmp($a['letter'], $b['letter']);
1959 }
1960
1961 /** @access private */
1962 function HTMLSmarty_outputfilter($src, & $smarty)
1963 {
1964     return str_replace('{subdir}', $smarty->_tpl_vars['subdir'], $src);
1965 }
1966 ?>
```

# File Source for class.phpdocpdf.php

Documentation for this file is available at [class.phpdocpdf.php](#)

```
1  <?php
2  /**
3  * Cezpdf callback class customized for phpDocumentor
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @package Converters
29 * @subpackage PDFdefault
30 * @author Greg Beaver <cellog@php.net>
31 * @copyright 2000-2006 Joshua Eichorn, Gregory Beaver
32 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33 * @version CVS: $Id: class.phpdocpdf.php 212211 2006-04-30 22:18:14Z cellog $
34 * @filesource
35 * @link http://www.phpdoc.org
36 * @link http://pear.php.net/PhpDocumentor
37 * @since 1.2
38 */
39
40 /**
41 * @ezPdf libraries */
42 include_once 'phpDocumentor/Converters/PDF/default/class.ezpdf.php';
43 include_once 'phpDocumentor/Converters/PDF/default/ParserPDF.inc';
44
45 // define a class extension to allow the use of a callback to get the table of
46 // contents, and to put the dots in the toc
47 /**
48 * @package Converters
49 * @subpackage PDFdefault
50 */
51 class phpdocpdf extends Cezpdf
52 {
53     var $reportContents = array();
54     var $indexContents = array();
55     var $indents = array();
56     var $font_dir = false;
57     var $set_pageNumbering = false;
58     var $converter;
59     var $save = '';
60     var $listType = 'ordered';
61     var $colorStack = array();
62
63     function phpdocpdf(& $pdfconverter, $fontdir, $paper='a4', $orientation='portrait')
64     {
65         Cezpdf::Cezpdf($paper, $orientation);
66         $this-> converter = $pdfconverter;
67         $this-> font_dir = $fontdir;
68     }
69 }
```

```

68
69     /**
70      * This really should be in the parent class
71      */
72     function getColor()
73     {
74         return $this-> currentColour;
75     }
76
77     function setColorArray($color)
78     {
79         $this-> setColor($color['r'], $color['g'], $color['b']);
80     }
81
82     /**
83      * Extract Pdfphp-format color from html-format color
84      * @return array
85      * @access private
86      */
87     function _extractColor($htmlcolor)
88     {
89         preg_match('/#[a-fA-F0-9][a-fA-F0-9][a-fA-F0-9][a-fA-F0-9][a-fA-F0-9][a-fA-F0-9]/', $htmlcolor, $color);
90         if (count($color) != 4)
91         {
92             return false;
93         }
94         $red = hexdec($color[1]) / hexdec('FF');
95         $green = hexdec($color[2]) / hexdec('FF');
96         $blue = hexdec($color[3]) / hexdec('FF');
97         return array('r' => $red, 'g' => $green, 'b' => $blue);
98     }
99
100    function validHTMLColor($color)
101    {
102        return $this-> _extractColor($htmlcolor);
103    }
104
105    function setHTMLColor($color)
106    {
107        fancy_debug('toplevel setting to', $color);
108        $this-> setColor($color['r'], $color['g'], $color['b']);
109    }
110
111    function textcolor($info)
112    {
113        if ($info['status'] == 'start')
114        {
115            array_push($this-> _colorStack, $this-> getColor());
116            $color = $this-> _extractColor($info['p']);
117            if ($color)
118            {
119                // fancy_debug('set color to ', $info['p'], $color, $this-> _colorStack);
120                $this-> setColorArray($color);
121            } else
122            {
123                array_pop($this-> _colorStack);
124            }
125        } elseif ($info['status'] == 'end')
126        {
127            // debug('unsetting');
128            $this-> setColorArray(array_pop($this-> _colorStack));
129        }
130    }
131
132    function rf($info)
133    {
134        $tmp = $info['p'];
135        $lvl = $tmp[0];
136        $lbl = rawurldecode(substr($tmp, 1));
137        $num = $this-> ezWhatPageNumber($this-> ezGetCurrentPageNumber());
138        $this-> reportContents[] = array($lbl, $num, $lvl);
139        $this-> addDestination('toc'.(count($this)-> reportContents)-1, 'FitH', $info['y']+ $info['height']);
140    }
141
142    function index($info)
143    {
144        $res = explode('|||', rawurldecode($info['p']));
145        $name = $res[0];

```

```

146     $descrip = $res[1];
147     $letter = $name[0];
148     if ($letter == '$') $letter = $name[1];
149     $this-> indexContents[strtoupper($letter)][] = array($name,$descrip,$this-
> ezWhatPageNumber($this-> ezGetCurrentPageNumber()),count($this-> reportContents) - 1);
150 }
151
152 function IndexLetter($info)
153 {
154     $letter = $info['p'];
155     $this-> transaction('start');
156     $ok=0;
157     while (! $ok){
158         $thisPageNum = $this-> ezPageCount;
159         $this-> saveState();
160         $this-> setColor(0.9,0.9,0.9);
161         $this-> filledRectangle($this-> ez['leftMargin'],$this-> y-$this-
> getFontHeight(18)+$this-> getFontDecender(18),$this-> ez['pageWidth']-$this-
> ez['leftMargin']-$this-> ez['rightMargin'],$this-> getFontHeight(18));
162         $this-> restoreState();
163         $this-> ezText($letter,18,array('justification'=> 'left'));
164         if ($this-> ezPageCount==$thisPageNum){
165             $this-> transaction('commit');
166             $ok=1;
167         } else {
168             // then we have moved onto a new page, bad bad, as the background colour will be on
the old one
169             $this-> transaction('rewind');
170             $this-> ezNewPage();
171         }
172     }
173 }
174
175 function dots($info)
176 {
177     // draw a dotted line over to the right and put on a page number
178     $tmp = $info['p'];
179     $lvl = $tmp[0];
180     $lbl = substr($tmp,1);
181     $xpos = 520;
182
183     switch($lvl)
184     {
185         case '1':
186             $size=16;
187             $thick=1;
188             break;
189         case '2':
190             $size=14;
191             $thick=1;
192             break;
193         case '3':
194             $size=12;
195             $thick=1;
196             break;
197         case '4':
198             $size=11;
199             $thick=1;
200             break;
201     }
202
203     $adjust = 0;
204     if ($size != 16) $adjust = 1;
205     $this-> saveState();
206     $this-> setLineStyle($thick,'round','','array(0,10));
207     $this-> line($xpos - (5*$adjust),$info['y'],$info['x']+5,$info['y']);
208     $this-> restoreState();
209     $this-> addText($xpos - (5*$adjust)+5,$info['y'],$size,$lbl);
210 }
211
212 /**
213 * @uses PDFParser extracts all meta-tags and processes text for output
214 */
215 function ezText($text,$size=0,$options=array(),$test=0)
216 {
217     $text = str_replace("\t" , " " , $text);
218     // paragraph breaks
219     $text = str_replace("<##P##>" , "\n" , $text);
220     $text = str_replace("<<c:i" , '< <c:i' , $text);
221     $text = str_replace("ilink>>" , "ilink> " , $text);

```

```

222     $this-> _save .= $text;
223 }
224
225 function setupTOC()
226 {
227     $parser = new PDFParser;
228     $parser-> parse($this-> _save,$this-> font_dir,$this);
229     $this-> _save = '';
230 }
231
232 function ezOutput($debug = false, $template)
233 {
234     if ($debug) return $this-> _save;
235     $this-> setupTOC();
236     if ($template)
237     {
238         uksort($this-> indexContents,'strnatcasecmp');
239         $xpos = 520;
240         $z = 0;
241         foreach($this-> indexContents as $letter => $contents)
242         {
243             if ($z++/50 == 0) {phpDocumentor_out('.');flush();}
244             uksort($this-> indexContents[$letter],array($this-
> converter,'mystrnatcasecmp'));
245         }
246         $template-> assign('indexcontents',$this-> indexContents);
247         $this-> ezText($template-> fetch('index.tpl'));
248         $this-> setupTOC();
249     }
250     return parent::ezOutput();
251 }
252
253 function _ezText($text,$size=0,$options=array(),$test=0)
254 {
255     return parent::ezText($text,$size,$options,$test);
256 }
257
258 function getYPlusOffset($offset)
259 {
260     return $this-> y + $offset;
261 }
262
263 function addMessage($message)
264 {
265     return parent::addMessage($message);
266     phpDocumentor_out($message . "\n");
267     flush();
268 }
269
270 function ezProcessText($text){
271     // this function will initially be used to implement underlining support, but could be
272     // used for a range of other
273     // purposes
274     $text = parent::ezProcessText($text);
275     $text =
276     str_replace(array('<UL>' , '</UL>' , '<LI>' , '</LI>' , '<OL>' , '</OL>' ,
277     'ti;/ol>' , '<blockquote>' , '</blockquote>' ), ,
278     array('<ul>' , '</ul>' , '<li>' , '</li>' , '<ol>' , '</ul>' , '</ul>' ,
279     quot;<C:indent:20>\n' , "<C:indent:-20>" ),$text);
280     // $text = str_replace("<ul>\n", "<ul>",$text);
281     $text = preg_replace("/\n+\s*(<ul>|<ol>)/"
282     $text;
283     // some problems fixed here - hack
284     $text = preg_replace('/<text [^>]+/>' , '' , $text);
285     $text = str_replace("<li>\n" , "<li>" , $text);
286     $text = preg_replace("/\n+\s*<li>/" , "<li>" , $text);
287     $text = str_replace("<mybr>" , "\n" , $text);
288     $text = str_replace('</li></ul>' , '</ul>' , $text);
289     $text = preg_replace("/^\n(\d+\s+.*)/" , '^\1' , $text);
290     $search = array('<ul>' , '</ul>' , '<ol>' , '<li>' , '</li>' );
291     $replace = array("<:indent:20>\n" , "\n<C:indent:-20>" , "<C:bullet>" , "\n" );
292     $text = str_replace($search,$replace,$text);
293     $text = preg_replace("/([\n])<C:bullet/" , "\\\1\n<C:bullet" ,
294     $text);
295     if (false) {
296         $fp = fopen("C:/Documents and Settings/Owner/Desktop/pdfsourceorig.txt" , 'a');
297         if ($fp)
298     }

```

```

293         fwrite($fp, $text);
294     }
295 }
296
297 return $text;
298 }
299
300 function indent($info)
301 {
302     $stuff = explode(':', $info['p']);
303     $margin = $stuff[0];
304     if (count($stuff) - 1)
305     {
306         $this-> listType = 'ordered';
307         $this-> listIndex = 1;
308     } else
309     {
310         if ($margin > 0)
311         {
312             $this-> listIndex = 1;
313         }
314         $this-> listType = 'unordered';
315     }
316     $this-> ez['leftMargin'] += $margin;
317 }
318
319 /**
320 * @author Murray Shields
321 */
322 function bullet($Data)
323 {
324     if ($this-> listType == 'ordered')
325     {
326         return $this-> orderedBullet($Data);
327     }
328     $D = abs($Data["decender"]);
329     $X = $Data["x"] - ($D * 2) - 10;
330     $Y = $Data["y"] + ($D * 1.5);
331     $this-> setLineStyle($D, "butt", "miter", array());
332     $this-> setColor(0,0,0);
333     $this-> ellipse($X, $Y, 1);
334 }
335
336 function orderedBullet($info)
337 {
338     $this-> addText($info['x']-20, $info['y']-1, 10, $this-> listIndex++ . '.');
339 }
340
341 function ezNewPage($debug=false)
342 {
343     parent::ezNewPage();
344     if (!$this-> set_pageNumbering)
345     {
346         $template = $this-> converter-> newSmarty();
347         $parser = new PDFParser;
348         $parser-> parse($template-> fetch('pagenumbering.tpl'), $this-
> font_dir, $this);
349     }
350     $this-> set_pageNumbering = true;
351 }
352 }
353 ?>

```

# File Source for ParserPDF.inc

Documentation for this file is available at [ParserPDF.inc](#)

```
1  <?php
2  /**
3   * This class handles the XML-based CezPDF markup language created to allow
4   * templates for the PDFdefaultConverter
5   *
6   * phpDocumentor :: automatic documentation generator
7   *
8   * PHP versions 4 and 5
9   *
10  * Copyright (c) 2002-2006 Gregory Beaver
11  *
12  * LICENSE:
13  *
14  * This library is free software; you can redistribute it
15  * and/or modify it under the terms of the GNU Lesser General
16  * Public License as published by the Free Software Foundation;
17  * either version 2.1 of the License, or (at your option) any
18  * later version.
19  *
20  * This library is distributed in the hope that it will be useful,
21  * but WITHOUT ANY WARRANTY; without even the implied warranty of
22  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23  * Lesser General Public License for more details.
24  *
25  * You should have received a copy of the GNU Lesser General Public
26  * License along with this library; if not, write to the Free Software
27  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28  *
29  * @package Converters
30  * @subpackage PDFdefault
31  * @author Greg Beaver <cellog@php.net>
32  * @copyright 2002-2006 Gregory Beaver
33  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version CVS: $Id: ParserPDF.inc 238276 2007-06-22 14:58:30Z ashnazg $
35  * @filesource
36  * @link http://www.phpdoc.org
37  * @link http://pear.php.net/PhpDocumentor
38  * @since 1.2
39  */
40 /**
41  * when <text> is found in an ezText input */
42  define('PHPDOCUMENTOR_PDF_EVENT_TEXT', 600);
43 /**
44  * when <text> is found in an ezText input */
45  define('PHPDOCUMENTOR_PDF_STATE_TEXT', 700);
46 /**
47  * used for parsing stuff between <text> */
48  define('PHPDOCUMENTOR_PDF_EVENT_CONTENT', 601);
49 /**
50  * used for parsing stuff between <text> */
51  define('PHPDOCUMENTOR_PDF_STATE_CONTENT', 701);
52 /**
53  * when <font> is found in an ezText input */
54  define('PHPDOCUMENTOR_PDF_EVENT_FONT', 602);
55 /**
56  * when <font> is found in an ezText input */
57  define('PHPDOCUMENTOR_PDF_STATE_FONT', 702);
58 /**
59  * when <newpage/> is found in an ezText input */
60  define('PHPDOCUMENTOR_PDF_EVENT_NEWPAGE', 603);
61 /**
62  * when <newpage/> is found in an ezText input */
63  define('PHPDOCUMENTOR_PDF_STATE_NEWPAGE', 703);
64 /**
65  * when <pdffunction> is found in an ezText input */
66  define('PHPDOCUMENTOR_PDF_EVENT_PDFFUNCTION', 604);
67 /**
68  * when <pdffunction> is found in an ezText input */
69  define('PHPDOCUMENTOR_PDF_STATE_PDFFUNCTION', 704);

70 /**
71  * @package Converters
72  * @subpackage PDFdefault
73  * @author Greg Beaver <cellog@php.net>
74  * @since 1.2
75  */
```

```

68 class PDFParser extends Parser
69 {
70     /**
71      * Mapping of event constants to events handler function names
72      * @var array
73      * @access private
74      */
75     var $eventHandlers
76     = array(
77         PHPDOCUMENTOR_PDF_EVENT_TEXT => 'handleText',
78         PHPDOCUMENTOR_PDF_EVENT_FONT => 'handleFont',
79         PHPDOCUMENTOR_PDF_EVENT_NEWPAGE => 'handleNewPage',
80         PARSER_EVENT_QUOTE => 'handleQuote',
81         PARSER_EVENT_NOEVENTS => 'defaultHandler',
82         PHPDOCUMENTOR_PDF_EVENT_CONTENT => 'handleContent',
83         PHPDOCUMENTOR_PDF_EVENT_PDFFUNCTION => 'handlePDFfunction',
84     );
85
86     /**
87      * Sets up the wordparser for this class
88      */
89     function PDFParser()
90     {
91         $this-> wp = new WordParser;
92         $this-> setupStates();
93     }
94     /**
95      * Parse text for PDFParser XML tags, and add the text to the PDF file
96      *
97      * @param string text to parse for PDFParser XML tags
98      * @param string full path to the font directory
99      * @param phpdcpdf
100     * @param boolean determines whether output is saved in a variable or
101         added directly to the output
102     * @staticvar integer used for recursion limiting if a handler for an event is not
103     found
104     * @return bool
105     */
106     function parse ($parse_data,$fontdir,& $pdf,$debug=false)
107     {
108         static $endrecur = 0;
109         $this-> _debug = $debug;
110
111         // initialize variables so E_ALL error_reporting doesn't complain
112         $pevent = 0;
113         $word = 0;
114         $this-> p_vars['event_stack'] = new EventStack;
115         $this-> p_flags['reset_quote_data'] = true;
116         $this-> p_vars['options'] = false;
117         $this-> p_vars['font_dir'] = $fontdir;
118         $this-> p_vars['text_size'] = false;
119         $this-> p_vars['pdf'] = & $pdf;
120
121         $this-> wp-> setup($parse_data);
122         $this-> wp-> setWhitespace(true);
123
124         do
125         {
126             $lpevent = $pevent;
127             $pevent = $this-> p_vars['event_stack']-> getEvent();
128             if ($lpevent != $pevent)
129             {
130                 $this-> p_vars['last_pevent'] = $lpevent;
131             }
132
133             if ($this-> p_vars['last_pevent'] != $pevent)
134             {
135                 // its a new event so the word parser needs to be reconfigured
136                 $this-> configWordParser($pevent);
137             }
138
139             $this-> p_vars['last_word'] = $word;
140             $word = $this-> wp-> getWord();
141
142             if (PHPDOCUMENTOR_DEBUG == true)
143             {
144                 echo "-----\n";
145                 echo "LAST: | " . $this-> p_vars['last_word'] . " |\n";
146                 echo "INDEX: ." . $this-> p_vars['curpar'] . "\n";
147             }
148         }
149     }

```

```

147             echo "PEVENT: " . $this-> getParserEventName($pevent) .
148             ;
149             echo "LASTPEVENT: " . $this-> getParserEventName($this-
150             > p_vars['last_pevent']) . "\n" ;
151             echo $this-> wp-> getPos() . " WORD: | " . $word."| \n\n" ;
152             }
153             if (isset($this-> eventHandlers[$pevent]))
154             {
155                 $handle = $this-> eventHandlers[$pevent];
156                 $this-> $handle($word, $pevent);
157             } else
158             {
159                 debug('WARNING: possible error, no ParserPDFParser handler for event number
160                 ' . $pevent);
161             }
162             if ($endrecur++ == 25)
163             {
164                 die("FATAL ERROR, recursion limit reached");
165             }
166             }
167             while (!$word === false));
168             if (false)
169             {
170                 $fp = fopen("C:/Documents and Settings/Owner/Desktop/pdfsource.txt"
171                 "a");
172                 fwrite($fp, $this-> wp-> data);
173                 fclose($fp);
174             }
175             }
176             */
177             /**
178             * Event Handlers
179             * @param string token
180             * @param integer event constant
181             * @access private
182             */
183             function defaultHandler($word, $pevent)
184             {
185                 if ($this-> checkEventPush($word, $pevent)) return;
186             }
187             /**
188             * Handles <newpage />
189             * @tutorial ParserPDF.cls#tags.newpage
190             */
191             function handleNewPage($word, $pevent)
192             {
193                 $this-> p_vars['event_stack']-> popEvent();
194                 $this-> p_vars['pdf']-> ezNewPage($this-> _debug);
195             }
196             /**
197             * Handles <text></text>
198             * @tutorial ParserPDF.cls#tags.text
199             */
200             function handleText($word, $pevent)
201             {
202                 $e = $this-> checkEventPush($word, $pevent);
203                 $el = $this-> checkEventPop($word, $pevent);
204                 if ($e == PARSER_EVENT_QUOTE) return;
205                 if ($el)
206                 {
207                     $this-> p_flags['textcolor'] = false;
208                     if (($a = $this-> p_vars['savecolor']) != $this-> p_vars['pdf']-> getColor())
209                     {
210                         $this-> p_vars['pdf']-> setColor($a['r'], $a['g'], $a['b']);
211                     }
212                     if ($this-> p_vars['last_word'] == '<text' )
213                     {
214                         // set up flags
215                         $this-> p_flags['paramval'] = false;
216                         $this-> p_flags['textcolor'] = false;
217                         $this-> p_vars['curparam'] = false;
218                         $this-> p_vars['savecolor'] = $this-> p_vars['pdf']-> getColor();
219                         $this-> p_vars['options'] = array();
220                         unset($this-> p_vars['quote_data']);
221                     }
222                     if (!$this-> p_flags['paramval'])
223                     {
224                         if ($e || $el) return;
225                         if ($word == '=') 
```

```

223     // {
224         debug('set paramval '.$this->p_vars['curparam']);
225         $this-> p_flags['paramval'] = true;
226         return;
227     }
228     $this-> p_vars['curparam'] = trim($word);
229 }
230 {
231     if ($this-> p_vars['last_pevent'] == PARSER_EVENT_QUOTE)
232     {
233         if ($this-> p_vars['curparam'] == 'size')
234         {
235             $this-> p_vars['text_size'] = (int)$this-> p_vars['quote_data'];
236             $this-> p_flags['paramval'] = false;
237             $this-> p_vars['curparam'] = false;
238             if (!$e && ! $el)
239             {
240                 $this-> p_vars['curparam'] = trim($word);
241             }
242             unset($this-> p_vars['quote_data']);
243         } elseif ($this-> p_vars['curparam'] == 'color')
244         {
245             if ($a = $this-> p_vars['pdf']-> validHTMLColor($this-
> p_vars['quote_data']))
246             {
247                 $this-> p_flags['textcolor'] = true;
248                 $this-> p_vars['pdf']-> setHTMLColor($a);
249             }
250         } else
251         {
252             if ($this-> p_vars['quote_data'] === (string)(int)$this-
> p_vars['quote_data']) $this-> p_vars['quote_data'] = (int)$this-> p_vars['quote_data'];
253             // debug('added '.$this->p_vars['curparam']. ' = '.$this-
>p_vars['quote_data']);
254             $this-> p_vars['options'][$this-> p_vars['curparam']] = $this-
> p_vars['quote_data'];
255             $this-> p_flags['paramval'] = false;
256             $this-> p_vars['curparam'] = false;
257             if (!$e && ! $el)
258             {
259                 $this-> p_vars['curparam'] = trim($word);
260             }
261             unset($this-> p_vars['quote_data']);
262         }
263     }
264 }
265
266 /**
267 * handles <font></font>
268 * @tutorial ParserPDF.cls#tags.font
269 */
270 function handleFont($word, $pevent)
271 {
272     $e = $this-> checkEventPush($word, $pevent);
273     $el = $this-> checkEventPop($word, $pevent);
274     if ($e == PARSER_EVENT_QUOTE) return;
275     if ($this-> p_vars['last_word'] == '<font' )
276     {
277         // set up flags
278         $this-> p_flags['paramval'] = false;
279         $this-> p_vars['curparam'] = false;
280         unset($this-> p_vars['quote_data']);
281     }
282     if (!$this-> p_flags['paramval'])
283     {
284         if ($e || $el) return;
285         if ($word == '=')
286         {
287             //debug('set paramval '.$this->p_vars['curparam']);
288             $this-> p_flags['paramval'] = true;
289             return;
290         }
291         $this-> p_vars['curparam'] = trim($word);
292     } else
293     {
294         if ($this-> p_vars['last_pevent'] == PARSER_EVENT_QUOTE)
295         {
296             if ($this-> p_vars['curparam'] == 'face')
297             {

```

```

299           //debug('set face to '.$this->p_vars['font_dir'] . $this-
>p_vars['quote_data'] . '.afm');
300           $this-> p_vars['pdf']-> selectFont($this-> p_vars['font_dir'] . $this-
> p_vars['quote_data'] . '.afm');
301           $this-> p_flags['paramval'] = false;
302           $this-> p_vars['curparam'] = false;
303           unset($this-> p_vars['quote_data']);
304       }
305   }
306 }
307 */
308 /**
309 * handles <pdffunction>
310 * @tutorial ParserPDF.cls#tags.pdffunction
311 */
312 function handlePDFFunction($word, $pevent)
313 {
    $e = $this-> checkEventPush($word, $pevent);
    $el = $this-> checkEventPop($word, $pevent);
    if ($e == PARSER_EVENT_QUOTE) return;
    if ($this-> p_vars['last_word'] == '<pdffunction:>')
    {
        // set up flags
        $this-> p_flags['paramval'] = $this-> p_flags['curparam'] = false;
        $this-> p_flags['returnval'] = false;
        $this-> p_vars['funcname'] = trim($word);
        debug("funcname is $word");
        $this-> p_vars['options'] = array();
        unset($this-> p_vars['quote_data']);
        if ($el) addErrorDie(PDERROR_PDFFUNCTION_NO_FUNC);
    }
    if (!$this-> p_flags['paramval'])
    {
        if ($el)
        { // call function, no parameters
            $func = $this-> p_vars['funcname'];
            if (!$func) addErrorDie(PDERROR_PDFFUNCTION_NO_FUNC);
            if (method_exists($this-> p_vars['pdf'], $func))
            {
                if (count($this-> p_vars['options']))
                {
                    fancy_debug("calling function $func", $this-
>p_vars['options']);
                }
                $a = call_user_func_array(array(& $this-> p_vars['pdf']), $func),
$this-> p_vars['options']);
            } else
            {
                debug("calling function $func");
                $a = $this-> p_vars['pdf']-> $func();
            }
            if ($this-> p_flags['returnval'])
            {
                debug("setting returnvar ".$this-
>p_vars['return_varname']);
                $this-> tempvars[$this-> p_vars['return_varname']] = $a;
            }
        } else
        {
            addWarning(PDERROR_PDF_METHOD_DOESNT_EXIST, $func);
        }
        return;
    }
    if ($e) return;
    if ($word == '=')
    {
        debug('set paramval '.$this->p_vars['curparam']);
        $this-> p_flags['paramval'] = true;
        return;
    }
    $this-> p_vars['curparam'] = trim($word);
} else
{
    if ($this-> p_vars['last_word'] == '=')
    { // check to see if we should use a tempvar from a previous return
        if (substr(trim($word), 0, 1) == '$')
        {
            if (substr(trim($word), 0, 7) == '$this->')
            {
                if (substr(trim($word), 0, 7) == '$this->')
                {
                    $a = substr(trim($word), 7);

```

```

374         $a = $this-> p_vars['pdf']-> $a;
375         // debug("set option to $word");
376         $this-> p_vars['options'][] = $a;
377         $this-> p_flags['paramval'] = false;
378         unset($this-> p_vars['quote_data']);
379     } else
380     { // this is a tempvar
381         if (!isset($this-> tempvars[substr(trim($word),1)]))
382         {
383             addErrorDie(PDERROR_PDF_TEMPVAR_DOESNT_EXIST,$this-
384 > p_vars['funcname'],trim($word),trim($word));
385         }
386         // $a = $this-> tempvars[substr(trim($word),1)];
387         // debug("set option to $word");
388         $this-> p_vars['options'][] = $a;
389         $this-> p_flags['paramval'] = false;
390         unset($this-> p_vars['quote_data']);
391     }
392 }
393 {
394     if ($this-> p_vars['last_pevent'] == PARSER_EVENT_QUOTE)
395     {
396         if ($this-> p_vars['quote_data'] === (string)(int)$this-
397 > p_vars['quote_data'])
398         {
399             $this-> p_vars['quote_data'] = (int)$this-> p_vars['quote_data'];
400         }
401         if ($this-> p_vars['curparam'] == 'return')
402         {
403             debug("param is return");
404             $this-> p_vars['return_varname'] = $this-> p_vars['quote_data'];
405             $this-> p_flags['returnval'] = true;
406         }
407     }
408     // fancy_debug("set option to arg",$this-
409 >p_vars['quote_data']);
410 }
411 {
412     $this-> p_flags['paramval'] = false;
413     unset($this-> p_vars['quote_data']);
414 }
415 if ($e1)
416 {
417     // call function, with parameters
418     $func = $this-> p_vars['funcname'];
419     if (method_exists($this-> p_vars['pdf'],$func))
420     {
421         // fancy_debug("calling function $func",$this-
422 >p_vars['options']);
423
424         if (count($this-> p_vars['options']))
425         {
426             fancy_debug("calling function $func");
427
428             if ($func == 'ezImage')
429             {
430                 // set padding to 5, width to 0, resize to none
431                 $this-> p_vars['options'][0] = 5;
432                 $this-> p_vars['options'][1] = 0;
433                 $this-> p_vars['options'][2] = 'none';
434             }
435             $a = call_user_func_array(array(& $this-> p_vars['pdf']),$func),
436             $this-> p_vars['options']);
437         } else
438         {
439             debug("calling function $func");
440             $a = $this-> p_vars['pdf']-> $func();
441
442             if ($this-> p_flags['returnval'])
443             {
444                 debug("setting returnvar ".$this-
445 >p_vars['return_varname']);
446                 $this-> tempvars[$this-> p_vars['return_varname']] = $a;
447             }
448         }
449     }
450 }
451 /**

```

```

448 * Adds content to the <text> tag
449 */
450 function handleContent($word, $pevent)
451 {
452     if ($e = $this-> checkEventPush($word, $pevent))
453     {
454         if ($e == PHPDOCUMENTOR_PDF_EVENT_FONT)
455         {
456             if (!isset($this-> p_vars['content'])) return;
457             $this-> p_vars['pdf']-> _ezText($this-> p_vars['content'], $this-
458 > p_vars['text_size'], $this-> p_vars['options']);
459             unset($this-> p_vars['content']);
460         }
461         return;
462     }
463     if ($this-> checkEventPop($word, $pevent))
464     {
465         $this-> wp-> backupPos($word);
466         if (!isset($this-> p_vars['content'])) return;
467         $this-> p_vars['pdf']-> _ezText($this-> p_vars['content'], $this-
468 > p_vars['text_size'], $this-> p_vars['options']);
469         unset($this-> p_vars['content']);
470     }
471     else
472     {
473         if (!isset($this-> p_vars['content'])) $this-> p_vars['content'] = '';
474         if (isset($this-> p_vars['quote_data']))
475         {
476             $this-> p_vars['content'] .= $this-> p_vars['quote_data'];
477             unset($this-> p_vars['quote_data']);
478         }
479     }
480 /**
481 * setup the parser tokens, and the pushEvent/popEvent arrays
482 * @see $tokens, $pushEvent, $popEvent
483 */
484
485 function setupStates()
486 {
487     $this-> tokens[STATE_NOEVENTS] =
array("<text"
488 >"      , "<font"      , "<newpage"
489 "      , "<newpage/>" , '<pdffunction:' , "''') ;
        $this-> tokens[PHPDOCUMENTOR_PDF_STATE_TEXT] =
array(">"      , "''", "''", "</text>" ) ;
        $this-> tokens[PHPDOCUMENTOR_PDF_STATE_FONT] =
array("/>"    , "''", "''") ;
        $this-> tokens[PHPDOCUMENTOR_PDF_STATE_CONTENT] =
array("<font"
491 >"      , "<pdffunction:' , "</text>" ) ;
        $this-> tokens[PHPDOCUMENTOR_PDF_STATE_PDFFUNCTION] =
array("''"
492 >"      , "''", "''") ;
        $this-> tokens[STATE_QUOTE] =
array("\\\\\""
493 >"      , "\\\\" , "\\\\" , "\\\\" ) ;
        $this-> tokens[STATE_ESCAPE] = false; // this tells the word parser to just
cycle
494
495 // For each event word to event mapings
496 $this-> pushEvent[PARSER_EVENT_QUOTE] =
497 array(
498     "\\\\"      => PARSER_EVENT_ESCAPE
499 );
500 $this-> popEvent[PARSER_EVENT_QUOTE] = array("''"
501 ##########
502 $this-> pushEvent[PARSER_EVENT_NOEVENTS] =
503 array(
504     "<text"      => PHPDOCUMENTOR_PDF_EVENT_TEXT,
505     "<font"       => PHPDOCUMENTOR_PDF_EVENT_FONT,
506     "<newpage />" => PHPDOCUMENTOR_PDF_EVENT_NEWPAGE,
507     "<newpage/>" => PHPDOCUMENTOR_PDF_EVENT_NEWPAGE,
508     "<pdffunction:" => PHPDOCUMENTOR_PDF_EVENT_PDFFUNCTION,
509     "''"        => PARSER_EVENT_QUOTE,
510 );
##########
512 $this-> pushEvent[PHPDOCUMENTOR_PDF_EVENT_TEXT] =
513 array(
514     "''"        => PARSER_EVENT_QUOTE,
515     '>'        => PHPDOCUMENTOR_PDF_EVENT_CONTENT,
516 );
517

```

```

518     $this-> popEvent[PHPDOCUMENATOR_PDF_EVENT_TEXT] = array("</text>" );
519 #####
520     $this-> pushEvent[PHPDOCUMENATOR_PDF_EVENT_FONT] =
521         array(
522             " " => PARSER_EVENT_QUOTE,
523         );
524
525     $this-> popEvent[PHPDOCUMENATOR_PDF_EVENT_FONT] = array(" />" );
526 #####
527     $this-> pushEvent[PHPDOCUMENATOR_PDF_EVENT_PDFFUNCTION] =
528         array(
529             " " => PARSER_EVENT_QUOTE,
530         );
531
532     $this-> popEvent[PHPDOCUMENATOR_PDF_EVENT_PDFFUNCTION] = array(" />" );
533 #####
534     $this-> pushEvent[PHPDOCUMENATOR_PDF_EVENT_CONTENT] =
535         array(
536             "<font" => PHPDOCUMENATOR_PDF_EVENT_FONT,
537             "<newpage />" => PHPDOCUMENATOR_PDF_EVENT_NEWPAGE,
538             "<newpage/>" => PHPDOCUMENATOR_PDF_EVENT_NEWPAGE,
539             "<pdffunction:" => PHPDOCUMENATOR_PDF_EVENT_PDFFUNCTION,
540         );
541
542     $this-> popEvent[PHPDOCUMENATOR_PDF_EVENT_CONTENT] = array(" </text>" );
543 }
544
545 /**
546 * Return the name of the parser event
547 * @param integer
548 */
549 function getParserEventName ($value)
550 {
551     $lookup = array(
552         PARSER_EVENT_NOEVENTS => "PARSER_EVENT_NOEVENTS",
553         PARSER_EVENT_QUOTE => "PARSER_EVENT_QUOTE",
554         PHPDOCUMENATOR_PDF_EVENT_TEXT => "PHPDOCUMENATOR_PDF_EVENT_TEXT",
555         PHPDOCUMENATOR_PDF_EVENT_CONTENT =>
556             "PHPDOCUMENATOR_PDF_EVENT_CONTENT",
557             PHPDOCUMENATOR_PDF_EVENT_FONT => "PHPDOCUMENATOR_PDF_EVENT_FONT",
558             PHPDOCUMENATOR_PDF_EVENT_PDFFUNCTION =>
559             "PHPDOCUMENATOR_PDF_EVENT_PDFFUNCTION"
560         );
561     if (isset($lookup[$value]))
562         return $lookup[$value];
563     else return $value;
564 }
565 ?>
```

# File Source for PDFdefaultConverter.inc

Documentation for this file is available at [PDFdefaultConverter.inc](#)

```
1  <?php
2  /**
3  * Outputs documentation in PDF format
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2002-2006 Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @package Converters
29 * @subpackage PDFdefault
30 * @author Greg Beaver <cellog@php.net>
31 * @copyright 2002-2006 Gregory Beaver
32 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33 * @version CVS: $Id: PDFdefaultConverter.inc 236747 2007-05-31 02:02:42Z ashnazg $
34 * @filesource
35 * @link http://www.phpdoc.org
36 * @link http://pear.php.net/PhpDocumentor
37 * @since 1.2
38 */
39 /**
40 * The Cezpdf class library
41 */
42 include_once('phpDocumentor/Converters/PDF/default/class.phpdcpdf.php');
43 /**
44 * PDF output converter.
45 * This Converter takes output from the {@link Parser} and converts it to PDF-ready output for
use with {@link Cezpdf}.
46 * This is now beta code
47 *
48 * @package Converters
49 * @subpackage PDFdefault
50 * @author Greg Beaver <cellog@php.net>
51 * @since 1.1
52 * @version $Id: PDFdefaultConverter.inc 236747 2007-05-31 02:02:42Z ashnazg $
53 * @todo Implement links to conflicts/inheritance
54 */
55 class PDFdefaultConverter extends Converter
56 {
57 /**
58 * default PDF Converter wants elements sorted by type as well as alphabetically
59 * @see Converter::sort_page_contents_by_type
60 * @var boolean
61 */
62 var $sort_absolutely_everything = true;
63 var $leftindex = array('classes' => false, 'pages' => false, 'functions' => false,
'defines' => false, 'globals' => false);
64 var $pagepackage_pagenums = array();
65 var $classpackage_pagenums = array();
```

```

66      /** @var string always PDF */
67      var $outputformat = 'PDF';
68      /** @var string always default */
69      var $name = 'default';
70      var $urpagepackage = false;
71      var $urclasspackage = false;
72      var $smarty_dir;
73      /**
74       * @var Cezpdf
75       */
76      var $pdf = false;
77      var $ric_set = array();
78      /**
79       * Source files for appendix C are stored here
80       *
81       * Format: array(array(package => packagename, code => array(highlightedsource code
82       * , ...)))
83       * @var array
84       */
85      var $sourcecode;
86      /**
87       * @see Converter::Converter()
88       */
89      function PDFdefaultConverter(& $allp, & $packp, & $classes, & $procpages, $po,
90      $pp, $qm, $targetDir, $templateDir, $title)
91      {
92          Converter::Converter($allp, $packp, $classes, $procpages, $po, $pp, $qm, $targetDir,
93          $templateDir, $title);
94          $this-> pdf =& new phpdocpdf($this, $this-> getConverterDir() . PATH_DELIMITER
95          .'templates/fonts/','letter');
96          $this-> pdf-> selectFont($this-> getConverterDir() . PATH_DELIMITER
97          .'templates/fonts/Helvetica.afm');
98          // put a line top and bottom on all the pages
99          $this-> pdf-> ezSetMargins(50,70,50,50);
100         $template = & $this-> newSmarty();
101         $this-> pdf-> ezText($template-> fetch('footer.tpl'));
102         $template-> assign('title',$title);
103         if (file_exists($this-> templateDir . 'templates' . PATH_DELIMITER . 'media'.
104         PATH_DELIMITER . 'logo.jpg'))
105         {
106             $template-> assign('logo',$this-> templateDir . 'templates' . PATH_DELIMITER .
107             'media'. PATH_DELIMITER . 'logo.jpg');
108         }
109         $this-> pdf-> ezText($template-> fetch('title_page.tpl'));
110         unset($template);
111     }
112     function writeSource($path, $value)
113     {
114         $templ = & $this-> newSmarty();
115         $pathinfo = $this-> proceduralpages-> getPathInfo($path, $this);
116         $templ-> assign('source',$value);
117         $templ-> assign('package',$pathinfo['package']);
118         $templ-> assign('subpackage',$pathinfo['subpackage']);
119         $templ-> assign('name',$pathinfo['name']);
120         $templ-> assign('source_loc',$pathinfo['source_loc']);
121         $templ-> assign('docs',$pathinfo['docs']);
122         $templ-> assign('dest', $this-> getFileSourceName($path));
123         $this-> setSourcePaths($path);
124         $sourcecode[$pathinfo['package']][] = $templ-> fetch('filesource.tpl');
125     }
126     function postProcess($text)
127     {
128         return htmlspecialchars($text);
129     }
130     function writeExample($title, $path, $source)
131     {
132         $templ = & $this-> newSmarty();
133         $templ-> assign('source',$source);
134         if (empty($title))
135         {
136             $title = 'example';
137             addWarning(PDERROR_EMPTY_EXAMPLE_TITLE, $path, $title);
138         }
139         $templ-> assign('title',$title);
140         $templ-> assign('file',$path);
141         $this-> pdf-> ezText($templ-> fetch('examplesource.tpl'));
142     }

```

```

139
140     function getExampleLink($path, $title)
141     {
142         return '';
143         return $this->  returnLink('{$subdir}__examplesource' . PATH_DELIMITER .
144 'exsource_'. $path.'.html', $title);
145     }
146
147     function getSourceLink($path)
148     {
149         //      var_dump(htmlentities('<c:ilink:'.$this->getFileSourceName($path).'>Source
150         //Code for this file</c:ilink>'));
151         return '<c:ilink:' . $this->  getFileSourceName($path).'>Source Code for this
152         file</c:ilink>' ;
153     }
154
155     function getFileSourceName($path, $anchor = '')
156     {
157         return urlencode($anchor . parent::getFileSourceName($path));
158     }
159
160     /**
161      * Retrieve a Converter-specific anchor to a segment of a source code file
162      * parsed via a {@tutorial tags.filesource.pkg} tag.
163      * @param string full path to source file
164      * @param string name of anchor
165      * @param string link text, if this is a link
166      * @param boolean returns either a link or a destination based on this
167      *          parameter
168      * @return string link to an anchor, or the anchor
169      */
170     function getSourceAnchor($sourcefile,$anchor,$text = '',$link = false)
171     {
172         if ($link)
173         {
174             return '<c:ilink:' . $this->  getFileSourceName($sourcefile, $anchor). '>' .
175 $text . '</c:ilink>' ;
176         } else
177         {
178             return '</text><pdffunction:addDestination arg="" . $this-
179 >  getFileSourceName($sourcefile, $anchor). '" arg="FitH" arg='.$this->y
180 /><text size="8">' ;
181         }
182     }
183
184     /**
185      * Returns a bookmark using Cezpdf 009
186      * @param abstractLink a descendant of abstractlink should be passed, and never text
187      * @param string text to display in the link
188      */
189     function returnSee(&      $element, $seltext = false)
190     {
191         if (!$element) return false;
192         if (!$seltext)
193         {
194             $seltext = '';
195             switch($element->  type)
196             {
197                 case 'tutorial' :
198                     $seltext = $element->  title;
199                     break;
200                 case 'method' :
201                     $seltext = $element->  name;
202                     break;
203                 case 'var' :
204                     $seltext = $element->  name;
205                     break;
206                 case 'const' :
207                     $seltext = $element->  name;
208                     break;
209                 case 'class' :
210                     $seltext = $element->  name;
211                     break;
212                 case 'function' :
213                     $seltext = $element->  name;
214                     break;
215                 case 'global' :
216                     $seltext = $element->  name;
217                     break;
218                 default :
219                     $seltext = $element->  name;
220                     if ($element->  type == 'function' || $element->  type == 'method') $seltext
221                     .= '()' ;
222                     break;
223             }
224             switch ($element->  type)
225             {

```

```

212         case 'tutorial' :
213             return '<c:ilink:' . urlencode($element-> type.$element-> package.$element-
> subpackage.$element-> name.$element-> section).'>' . $eltext.'</c:ilink>' ;
214         case 'page' :
215             return '<c:ilink:' . urlencode($element-> type.$element-> package.$element-
> path).'>' . $eltext.'</c:ilink>' ;
216         case 'define' :
217         case 'global' :
218         case 'class' :
219         case 'function' :
220             return '<c:ilink:' . urlencode($element-> type.$element-> package.$element-
> name).'>' . $eltext.'</c:ilink>' ;
221         case 'method' :
222         case 'var' :
223         case 'const' :
224             return '<c:ilink:' . urlencode($element-> type.$element-> package.$element-
> class.'::'.$element-> name).'>' . $eltext.'</c:ilink>' ;
225     }
226     return $element;
227 }
228 /**
229 * @param string
230 * @param string
231 * @return string <c:alink:$link>$text</c:alink>
232 */
233 function returnLink($link,$text)
234 {
235     return "      <c:alink: $link> $text</c:alink>      " ;
236 }
237 }
238 /**
239 * Convert README/INSTALL/CHANGELOG file contents to output format
240 * @param README/INSTALL/CHANGELOG
241 * @param string contents of the file
242 */
243 function Convert_RIC($name, $contents)
244 {
245     $this-> ric_set[$name] = $contents;
246 }
247
248 function convertDocBlock(& $element)
249 {
250     if (!$element-> docblock) return;
251     $template = & $this-> newSmarty();
252
253     $nopackage = true;
254     if ($element-> type == 'page' || $element-> type == 'class') $nopackage = false;
255     $stages = $element-> docblock-> listTags();
256     $tags = array();
257     $names = array('staticvar' => 'Static Variable', 'deprec' =>
'Deprecated', 'abstract' => 'Abstract Element', 'todo' => 'TODO');
258     if (!$nopackage)
259     {
260         $tags[] = array('keyword' => 'Package', 'data' => $element-> docblock-
> package);
261         if (!empty($element-> docblock-> subpackage)) $tags[] = array('keyword' =>
'Sub-Package', 'data' => $element-> docblock-> subpackage);
262     }
263     if ($element-> docblock-> var)
264     {
265         $a = $element-> docblock-> var-> Convert($this);
266         if (!empty($a))
267             $tags[] = array('keyword' => 'Var', 'data' => $a);
268     }
269     if ($element-> docblock-> funcglobals)
270     foreach($element-> docblock-> funcglobals as $global => $val)
271     {
272         if ($a = $this-> getGlobalLink($global,$element-> docblock-> package))
273         {
274             $global = $a;
275         }
276         $b = Converter::getLink($val[0]);
277         if (is_object($b) && phpDocumentor_get_class($b) == 'classlink')
278         {
279             $val[0] = $this-> returnSee($b);
280         }
281         $tags[] = array('keyword' => 'Global Variable Used', 'data' => $val[0].'
'. $global.' : ' . $val[1]-> Convert($this));
282     }

```

```

284     if ($element-> docblock-> statics)
285     foreach($element-> docblock-> statics as $static => $val)
286     {
287         $a = $val-> Convert($this);
288         $tags[] = array('keyword' => 'Static Variable Used', 'data' => $val-
289 > converted_returnType.'.'.$static.':'.$a);
290     }
291     if ($element-> docblock-> properties)
292     foreach($element-> docblock-> properties as $property => $val)
293     {
294         $a = $val-> Convert($this);
295         $tags[] = array('keyword' => ucfirst($val-> keyword), 'data' => $val-
296 > converted_returnType.'.'.$property.':'.$a);
297     }
298     foreach($tags as $tag)
299     {
300         if (isset($names[$tag-> keyword])) $tag-> keyword = $names[$tag-> keyword];
301         $tags[] = array("keyword" => ucfirst($tag-> keyword), "data" => $tag->
302 > keyword);
303     }
304     $utags = array();
305     foreach($element-> docblock-> unknown_tags as $keyword => $t)
306     {
307         foreach($t as $tag)
308             $utags[] = array('keyword' => $keyword, 'data' => $tag-> Convert($this));
309     }
310     if ($element-> type == 'packagepage') return;
311     $sdesc = $element-> docblock-> getSDesc($this);
312     $desc = $element-> docblock-> getDesc($this);
313     $template-> assign('utags', $utags);
314     $template-> assign('tags', $tags);
315     $template-> assign('sdesc', $sdesc);
316     $template-> assign('desc', $desc);
317     if (false) // $element->type != 'page'
318     {
319         if ($element-> type != 'var' && $element-> type != 'method')
320         {
321             $this-> pdf-> addDestination(urlencode($element-> type.$element-
322 > docblock-> package.$element-> name), 'FitH', $this-> pdf-> y);
323         } else
324         {
325             $this-> pdf-> addDestination(urlencode($element-> type.$element-
326 > docblock-> package.$element-> class.'::'.$element-> name), 'FitH', $this-> pdf-> y);
327         } elseif (false)
328         {
329             $this-> pdf-> addDestination(urlencode('page').$element-> parent-
330 > package.$element-> parent-> getPath()), 'FitH', $this-> pdf-> y);
331             $this-> convertParams($element);
332             $this-> pdf-> ezText($template-> fetch('docblock.tpl'));
333         }
334     function convertParams(& $element)
335     {
336         if ($element-> type != 'function' && $element-> type != 'method') return;
337         if (count($element-> docblock-> params))
338         {
339             $template = & $this-> newSmarty();
340             $params = array();
341             if (count($element-> docblock-> params))
342             foreach($element-> docblock-> params as $param => $val)
343             {
344                 $a = $val-> Convert($this);
345                 $params[] = array("name" => $param, "type" => $val-
346 > converted_returnType, "description" => $a);
347             }
348             $template-> assign('params', $params);
349             $this-> pdf-> ezText($template-> fetch('params.tpl'));
350         }
351     function convertGlobal(& $element)
352     {
353         $sdesc = '';
354         if ($element-> docblock-> sdesc)
355         {
356             $sdesc = $element-> docblock-> sdesc-> Convert($this);
357         }
358         $template = & $this-> newSmarty();

```

```

357     $template-> assign('linenumber',$element-> getLineNumber());
358     if ($this-> hasSourceCode($element-> getPath()))
359     $template-> assign('slink',$this-> getSourceAnchor($element-> getPath(),$element-
> getLineNumber(),$element-> getLineNumber(),true));
360     else
361     $template-> assign('slink', false);
362     $template-> assign('dest', urlencode($element-> type.$element-> docblock-
> package.$element-> name));
363     $template-> assign('type',$element-> getType($this));
364     $template-> assign('name',$element-> name);
365     $template-> assign('value',$this-> getGlobalValue($element-> getValue()));
366     $template-> assign('sdesc',$sdesc);
367     $this-> pdf-> ezText($template-> fetch('global.tpl'));
368     $this-> convertDocBlock($element);
369 }
370
371 function getGlobalValue($value)
372 {
373     return parent::getGlobalValue(htmlspecialchars($value));
374 }
375
376 function convertMethod(& $element)
377 {
378     $sdesc = '';
379     if ($element-> docblock-> sdesc)
380     {
381         $sdesc = $element-> docblock-> sdesc-> Convert($this);
382     }
383     $params = array();
384     if (count($element-> docblock-> params))
385     foreach($element-> docblock-> params as $param => $val)
386     {
387         $a = $val-> Convert($this);
388         $params[$param] = array("var" => $param, "datatype" =>
389         $val-> convertedReturnType, "data" => $a);
390     }
391     if ($element-> docblock-> return)
392     {
393         if (!$element-> docblock-> return-> returnType) $element-> docblock-
> returnType = 'void';
394         $template = & $this-> newSmarty();
395         $template-> assign('class',$this-> class);
396         $template-> assign('constructor',$element-> isConstructor);
397         $template-> assign('dest', urlencode($element-> type.$element-> docblock-
> package.$element-> class.'::'.$element-> name));
398         $template-> assign('linenumber',$element-> getLineNumber());
399         if ($this-> hasSourceCode($element-> getPath()))
400         $template-> assign('slink',$this-> getSourceAnchor($element-> getPath(),$element-
> getLineNumber(),$element-> getLineNumber(),true));
401     else
402         $template-> assign('slink',false);
403     $ret = 'void';
404     if ($element-> docblock-> return)
405     {
406         $ret = $element-> docblock-> return-> returnType;
407     }
408     $template-> assign('return',$ret);
409     $template-> assign('functioncall',$element-> getFunctionCall());
410     $template-> assign('intricatefunctioncall',$element-
> getIntricateFunctionCall($this,$params));
411     $template-> assign('sdesc',$sdesc);
412     $this-> pdf-> ezText($template-> fetch('method.tpl'));
413     $this-> convertDocBlock($element);
414 }
415
416 function convertVar(& $element)
417 {
418     $sdesc = '';
419     if ($element-> docblock-> sdesc)
420     {
421         $sdesc = $element-> docblock-> sdesc-> Convert($this);
422     }
423     $template = & $this-> newSmarty();
424     $template-> assign('class',$this-> class);
425     $template-> assign('linenumber',$element-> getLineNumber());
426     if ($this-> hasSourceCode($element-> getPath()))
427     $template-> assign('slink',$this-> getSourceAnchor($element-> getPath(),$element-
> getLineNumber(),$element-> getLineNumber(),true));
428     else

```

```

429     $template-> assign('slink',false);
430     $template-> assign('type',$element-> docblock-> var-> returnType);
431     $template-> assign('name',$element-> name);
432     $template-> assign('dest', urlencode($element-> type.$element-> docblock-
> package.$element-> class.'::'.$element-> name));
433     $template-> assign('value',$element-> value);
434     $template-> assign('sdesc',$sdesc);
435     $this-> pdf-> ezText($template-> fetch('var.tpl'));
436     $this-> convertDocBlock($element);
437 }
438
439 function convertConst(& $element)
440 {
441     $sdesc = '';
442     if ($element-> docblock-> sdesc)
443     {
444         $sdesc = $element-> docblock-> sdesc-> Convert($this);
445     }
446     $template = & $this-> newSmarty();
447     $template-> assign('class',$this-> class);
448     $template-> assign('linenumber',$element-> getLineNumber());
449     if ($this-> hasSourceCode($element-> getPath()))
450     $template-> assign('slink',$this-> getSourceAnchor($element-> getPath(),$element-
> getLineNumber(),$element-> getLineNumber(),true));
451     else
452         $template-> assign('slink',false);
453     $template-> assign('name',$element-> name);
454     $template-> assign('dest', urlencode($element-> type.$element-> docblock-
> package.$element-> class.'::'.$element-> name));
455     $template-> assign('value',$element-> value);
456     $template-> assign('sdesc',$sdesc);
457     $this-> pdf-> ezText($template-> fetch('const.tpl'));
458     $this-> convertDocBlock($element);
459 }
460
461 function convertClass(& $element)
462 {
463     $template = & $this-> newSmarty();
464     if ($this-> curclasspackage != $element-> docblock-> package)
465     {
466         $template-> assign('includeheader',true);
467         if (isset($this-> package_pages[$element-> docblock-> package]))
468         {
469             $template-> assign('ppage',$this-> package_pages[$element-> docblock-
> package]);
470             $template-> assign('isclass',true);
471             unset($this-> package_pages[$element-> docblock-> package]);
472         }
473         $template-> assign('classeslink', rawurlencode("Package " . $element-
> docblock-> package." Classes" ));
474     }
475     $sdesc = '';
476     if ($element-> docblock-> sdesc)
477     {
478         $sdesc = $element-> docblock-> sdesc-> Convert($this);
479     }
480     $this-> curclasspackage = $element-> docblock-> package;
481     $template-> assign('dest', urlencode($element-> type.$element-> docblock-
> package.$element-> name));
482     $template-> assign('package',$element-> docblock-> package);
483     $template-> assign('linenumber',$element-> getLineNumber());
484     if ($this-> hasSourceCode($element-> getPath()))
485     $template-> assign('slink',$this-> getSourceAnchor($element-> getPath(),$element-
> getLineNumber(),$element-> getLineNumber(),true));
486     else
487         $template-> assign('slink',false);
488     $template-> assign('name',$element-> name);
489     $template-> assign('sdesc',$sdesc);
490     $this-> pdf-> ezText($template-> fetch('class.tpl'));
491     $this-> convertDocBlock($element);
492 }
493
494 function convertInclude(& $element)
495 {
496     $template = & $this-> newSmarty();
497     $template-> assign('linenumber',$element-> getLineNumber());
498     if ($this-> hasSourceCode($element-> getPath()))
499     $template-> assign('slink',$this-> getSourceAnchor($element-> getPath(),$element-
> getLineNumber(),$element-> getLineNumber(),true));
500     else

```

```

501     $template-> assign('slink',false);
502     $template-> assign('name',$element-> name);
503     $template-> assign('value',$this-> getIncludeValue($element-> getValue(),
504     getPath()));
505     $this-> pdf-> ezText($template-> fetch('include.tpl'));
506     $this-> convertDocBlock($element);
507 }
508
509     function convertFunction(& $element)
510     {
511         $sdesc = '';
512         if ($element-> docblock-> sdesc)
513         {
514             $sdesc = $element-> docblock-> sdesc-> Convert($this);
515         }
516         $params = array();
517         if (count($element-> docblock-> params))
518         foreach($element-> docblock-> params as $param => $val)
519         {
520             $a = $val-> Convert($this);
521             $params[$param] = array("var" => $param, "datatype" =>
522             $val-> converted_returnType, "data" => $a);
523         }
524         if (!$element-> docblock-> return)
525         {
526             $element-> docblock-> return-> returnType = 'void';
527         }
528         $template = & $this-> newSmarty();
529         $template-> assign('dest', urlencode($element-> type.$element-> docblock-
530         > package.$element-> name));
531         $template-> assign('linenumber', $element-> getLineNumber());
532         if ($this-> hasSourceCode($element-> getPath()))
533         $template-> assign('slink', $this-> getSourceAnchor($element-> getPath()), $element-
534         > getLineNumber(), $element-> getLineNumber(true));
535         else
536             $template-> assign('slink',false);
537             $template-> assign('return', $element-> docblock-> return-> returnType);
538             $template-> assign('functioncall', $element-> getFunctionCall());
539             $template-> assign('intricatefunctioncall', $element-
540         > getIntricateFunctionCall($this, $params));
541             $template-> assign('sdesc', $sdesc);
542             $this-> pdf-> ezText($template-> fetch('function.tpl'));
543             $this-> convertDocBlock($element);
544     }
545
546     function convertDefine(& $element)
547     {
548         $sdesc = '';
549         if ($element-> docblock-> sdesc)
550         {
551             $sdesc = $element-> docblock-> sdesc-> Convert($this);
552         }
553         $template = & $this-> newSmarty();
554         $template-> assign('linenumber', $element-> getLineNumber());
555         if ($this-> hasSourceCode($element-> getPath()))
556         $template-> assign('slink', $this-> getSourceAnchor($element-> getPath()), $element-
557         > getLineNumber(), $element-> getLineNumber(true));
558         else
559             $template-> assign('slink',false);
560             $template-> assign('name', $element-> name);
561             $template-> assign('dest', urlencode($element-> type.$element-> docblock-
562         > package.$element-> name));
563             $template-> assign('value', $element-> value);
564             $template-> assign('sdesc', $sdesc);
565             $this-> pdf-> ezText($template-> fetch('define.tpl'));
566             $this-> convertDocBlock($element);
567     }
568
569     function convertPage(& $element)
570     {
571         $template = & $this-> newSmarty();
572         $template-> assign('includeheader', false);
573         $sdesc = '';
574         if ($element-> docblock-> sdesc)
575         {
576             $sdesc = $element-> docblock-> sdesc-> Convert($this);
577         }
578         if (count($element-> elements) || ($sdesc) || count($element-> docblock-> tags))
579         {
580             if ($this-> curpagepackage != $element-> parent-> package)

```

```

574         {
575             $template-> assign('includeheader',true);
576             if (isset($this-> package_pages[$element-> parent-> package]))
577             {
578                 $template-> assign('ppage',$this-> package_pages[$element-> parent-
579 > package]);
580             }
581         }
582         $this-> curpagepackage = $element-> parent-> package;
583         $template-> assign('dest', urlencode('page' . $element-> parent-
584 > package . $element-> parent-> getPath()));
585         $template-> assign('sdesc',$sdesc);
586         $template-> assign('package',$element-> parent-> package);
587         $template-> assign('name',$element-> parent-> file);
588         $this-> pdf-> ezText($template-> fetch('page.tpl'));
589         $this-> convertDocBlock($element);
590     }
591 }
592 /**
593 * Used to translate an XML DocBook tag from a tutorial by reading the
594 * options.ini file for the template.
595 * @param string tag name
596 * @param string any attributes Format: array(name => value)
597 * @param string the tag contents, if any
598 * @param string the tag contents, if any, unpost-processed
599 * @return string
600 */
601 function TranslateTag($name,$attr,$cdata,$unconvertedcdata)
602 {
603     if ($name == 'example' && @
604     {
605         $cdata = htmlspecialchars($cdata);
606         $unconvertedcdata = htmlspecialchars($unconvertedcdata);
607     }
608     if ($name == 'programlisting' && @
609     {
610         $unconvertedcdata = strtr($unconvertedcdata,
611         array_flip(get_html_translation_table(HTML_SPECIALCHARS)));
612         $a = parent::TranslateTag($name, $attr, $cdata, $unconvertedcdata);
613         // var_dump(htmlspecialchars($cdata), htmlspecialchars($unconvertedcdata),
614         htmlspecialchars($a));
615         return $a;
616     }
617     return parent::TranslateTag($name, $attr, $cdata, $unconvertedcdata);
618 }
619 function getPageName(& $element)
620 {
621     if (phpDocumentor_get_class($element) == 'parserpage') return $element-> getName();
622     return $element-> parent-> getName();
623 }
624 function getTutorialId($package,$subpackage,$tutorial,$id)
625 {
626     return 'tutorial' . $package . $subpackage . $tutorial . $id;
627 }
628 function getCData($value)
629 {
630     return
631     str_replace(array('<C:' , '<C:' ),array("amp;lt;c:" , "&lt;c:" ),$value);
632 }
633 /**
634 * @deprecated html package pages just don't work with PDF, use {@tutorial tutorials.pkg}
635 */
636 function convertPackagepage(& $element)
637 {
638     $x = $element-> Convert($this);
639     $x = substr($x,strpos($x,'<body' ));
640     $this-> package_pages[$element-> package] = trim(substr($x,strpos($x,'>' ) +
641 1,strpos($x,'</body>' ) - 6));
642 }
643 function convertTutorial(& $element)
644 {
645     $x = $element-> Convert($this, true);
646 }

```

```

648     $template = & $this-> newSmarty();
649     $template-> assign('package', $element-> package);
650     $template-> assign('subpackage', $element-> subpackage);
651     $template-> assign('contents', $x);
652     $template-> assign('title', $element-> getTitle($this));
653     $template-> assign('child', $element-> parent);
654     if (isset($element-> parent-> parent)) $template-> assign('hasparent', $element-
> parent);
655     $template-> assign('element', $element);
656     $this-> pdf-> ezText($template-> fetch('tutorial.tpl'));
657 }
658 /**
659 * returns a template-enabled array of class trees
660 *
661 * @param string $package package to generate a class tree for
662 * @see $roots, getRootTree()
663 */
664 function generateFormattedClassTrees($package)
665 {
666     if (!isset($this-> roots[$package])) return array();
667     $roots = $trees = array();
668     $roots = $this-> roots[$package];
669     for ($i=0; $i< count($roots); $i++)
670     {
671         $trees[] = array('class' => $roots[$i], 'class_tree' =>
672 " <ul>\n" . $this-> getRootTree($this-
673 > getSortedClassTreeFromClass($roots[$i], $package, ''), $package). "</ul>\n");
674     }
675     return $trees;
676 }
677 /**
678 * return formatted class tree for the Class Trees page
679 *
680 * @param array $tree output from {@link getSortedClassTreeFromClass()}
681 * @see Classes::$definitechild, generateFormattedClassTrees()
682 * @return string
683 */
684 function getRootTree($tree, $package)
685 {
686     if (!$tree) return '';
687     $my_tree = '';
688     $cur = '#root';
689     $lastcur = array(false);
690     $kids = array();
691     $dopar = false;
692     if ($tree[$cur]['parent'])
693     {
694         $dopar = true;
695         if (!is_object($tree[$cur]['parent']))
696         {
697             // debug("parent ".$tree[$cur]['parent']."' not found");
698             $my_tree .= '<li>' . $tree[$cur]['parent'] . '<ul>';
699         }
700         else
701         {
702             // debug("parent ".$this->returnSee($tree[$cur]['parent'],
703             false, false)." in other package");
704             $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['parent'], false,
705             false);
706             if ($tree[$cur]['parent']-> package != $package) $my_tree .= '
707 <b>(Different package)</b><ul>';
708         }
709     }
710     fancy_debug($cur, $lastcur, $kids);
711     if (count($tree[$cur]['children']))
712     {
713         debug("$cur has children");
714         if (!isset($kids[$cur]))
715         {
716             debug("set $cur kids");
717             $kids[$cur] = 1;
718             $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link'], false,
719             false);
720             $my_tree .= '<ul>' . "\n";
721             array_push($lastcur, $cur);

```

```

721     list($cur) = each($tree[$cur]['children']);
722     //      var_dump('listed',$cur);
723     if ($cur)
724     {
725         $cur = $cur['package'] . '#' . $cur['class'];
726         //      debug("set cur to child $cur");
727         //      $my_tree .= '<li>' . $this->returnSee($tree[$cur]['link'], false,
728         //      continue;
729     } else
730     {
731         //      debug("end of children for $cur");
732         $cur = array_pop($lastcur);
733         $cur = array_pop($lastcur);
734         $my_tree .= '</ul></li>' . "\n";
735         if ($dopar && ($cur == '#root' || !$cur)) $my_tree .=
736         '</ul></li>';
737     }
738     {
739         //      debug("$cur has no children");
740         $my_tree .= '<li>' . $this-> returnSee($tree[$cur]['link'], false,
741         false). "</li>" ;
742         if ($dopar && $cur == '#root') $my_tree .= '</ul></li>';
743         $cur = array_pop($lastcur);
744     }
745     } while ($cur);
746     return $my_tree;
747 }
748 /**
749 * calls {@link Cezpdf::ezOutput()} and writes documentation.pdf to targetDir
750 */
751 function Output($title = 'Generated Documentation')
752 {
753     phpDocumentor_out("Generating PDF file..");
754     flush();
755     $template = & $this-> newSmarty();
756     $this-> pdf-> ezText($template-> fetch('appendix_title_page.tpl'));
757     $trees = array();
758     foreach($this-> all_packages as $package => $rest)
759     {
760         if (!isset($this-> pkg_elements[$package])) continue;
761         $a = array();
762         phpDocumentor_out('.');
763         flush();
764         $a['package'] = $package;
765         $a['trees'] = $this-> generateFormattedClassTrees($package);
766         $trees[] = $a;
767     }
768     $template-> assign('trees', $trees);
769     $this-> pdf-> ezText($template-> fetch('classtree.tpl'));
770     phpDocumentor_out('.');
771     if (count($this-> ric_set))
772     $this-> pdf-> ezText($template-> fetch('ric_title_page.tpl'));
773     foreach($this-> ric_set as $name => $contents)
774     {
775         $template-> assign('contents', $contents);
776         $template-> assign('name', $name);
777         $this-> pdf-> ezText($template-> fetch('ric.tpl'));
778     }
779     if (count($this-> sourcecode))
780     {
781         $this-> pdf-> ezText($template-> fetch('source_title_page.tpl'));
782         $template-> assign('source', $this-> sourcecode);
783         $this-> pdf-> ezText($template-> fetch('source_loop.tpl'));
784     }
785     flush();
786     if (count($this-> todoList))
787     {
788         $todolist = array();
789         foreach($this-> todoList as $package => $alltodos)
790         {
791             foreach($alltodos as $todos)
792             {
793                 $converted = array();
794                 $converted['link'] = $this-> returnSee($todos[0]);
795                 if (!is_array($todos[1]))
796                 {
797                     $converted['todos'][] = $todos[1]-> Convert($this);

```

```

798     } else
799     {
800         foreach($todos[1] as $todo)
801         {
802             $converted['todos'][] = $todo-> Convert($this);
803         }
804     }
805     $todolist[$package][] = $converted;
806 }
807
808     $template-> assign('todos', $todolist);
809
810     $this-> pdf-> ezText($template-> fetch('todolist.tpl'));
811 }
812 if (false) {
813 $fp = fopen("C:/Documents and Settings/Owner/Desktop/pdfsourceorig.txt" , 'w');
814 if ($fp)
815 {
816     $a = $this-> pdf-> ezOutput(true); // debug
817     fwrite($fp, $a, strlen($a));
818     fclose($fp);
819 }
820
821 $this-> pdf-> setupTOC();
822 $template-> assign('contents', $this-> pdf-> reportContents);
823 $this-> pdf-> ezText($template-> fetch('toc.tpl'));
824 $x = $this-> pdf-> ezOutput(false, $template);
825 phpDocumentor_out("done\n");
826 flush();
827 $this-> writeFile("documentation.pdf" , $x, true);
828 }
829
830 function mystrnatcasecmp($a,$b)
831 {
832     return strnatcasecmp($a[0],$b[0]);
833 }
834
835
836 /**
837 * @param string name of class
838 * @param string package name
839 * @param string full path to look in (used in index generation)
840 * @return mixed false if not found, or an html a link to the class's documentation
841 * @see parent::getClassLink()
842 */
843 function getClassLink($expr,$package, $file = false,$text = false)
844 {
845     $a = Converter::getClassLink($expr,$package,$file);
846     if (!$a) return false;
847     return $this-> returnSee($a, $text);
848 }
849
850 /**
851 * @param string name of function
852 * @param string package name
853 * @param string full path to look in (used in index generation)
854 * @param boolean deprecated
855 * @return mixed false if not found, or an html a link to the function's documentation
856 * @see parent::getFunctionLink()
857 */
858 function getFunctionLink($expr,$package, $file = false,$text = false)
859 {
860     $a = Converter::getFunctionLink($expr,$package,$file);
861     if (!$a) return false;
862     return $this-> returnSee($a, $text);
863 }
864
865 /**
866 * @param string name of define
867 * @param string package name
868 * @param string full path to look in (used in index generation)
869 * @param boolean deprecated
870 * @return mixed false if not found, or an html a link to the define's documentation
871 * @see parent::getDefineLink()
872 */
873 function getDefineLink($expr,$package, $file = false,$text = false)
874 {
875     $a = Converter::getDefineLink($expr,$package,$file);
876     if (!$a) return false;
877     return $this-> returnSee($a, $text);

```

```

878 }
879 /**
880 * @param string name of global variable
881 * @param string package name
882 * @param string full path to look in (used in index generation)
883 * @param boolean deprecated
884 * @return mixed false if not found, or an html a link to the global variable's
885 documentation
886 * @see parent::getGlobalLink()
887 */
888 function getGlobalLink($expr,$package, $file = false,$text = false)
889 {
890     $a = Converter::getGlobalLink($expr,$package,$file);
891     if (!$a) return false;
892     return $this-> returnSee($a, $text);
893 }
894 /**
895 * @param string name of procedural page
896 * @param string package name
897 * @param string full path to look in (used in index generation)
898 * @param boolean deprecated
899 * @return mixed false if not found, or an html a link to the procedural page's
900 documentation
901 * @see parent::getPageLink()
902 */
903 function getPageLink($expr,$package, $path = false,$text = false)
904 {
905     $a = Converter::getPageLink($expr,$package,$path);
906     if (!$a) return false;
907     return $this-> returnSee($a, $text);
908 }
909 /**
910 * @param string name of method
911 * @param string class containing method
912 * @param string package name
913 * @param string full path to look in (used in index generation)
914 * @param boolean deprecated
915 * @return mixed false if not found, or an html a link to the method's documentation
916 * @see parent::getMethodLink()
917 */
918 function getMethodLink($expr,$class,$package, $file = false,$text = false)
919 {
920     $a = Converter::getMethodLink($expr,$class,$package,$file);
921     if (!$a) return false;
922     return $this-> returnSee($a, $text);
923 }
924 /**
925 * @param string name of var
926 * @param string class containing var
927 * @param string package name
928 * @param string full path to look in (used in index generation)
929 * @param boolean deprecated
930 * @param boolean return just the URL, or enclose it in an html a tag
931 * @return mixed false if not found, or an html a link to the var's documentation
932 * @see parent::getVarLink()
933 */
934 function getVarLink($expr,$class,$package, $file = false,$text = false)
935 {
936     $a = Converter::getVarLink($expr,$class,$package,$file);
937     if (!$a) return false;
938     return $this-> returnSee($a, $text);
939 }
940 /**
941 * @param string name of class constant
942 * @param string class containing class constant
943 * @param string package name
944 * @param string full path to look in (used in index generation)
945 * @param boolean deprecated
946 * @param boolean return just the URL, or enclose it in an html a tag
947 * @return mixed false if not found, or an html a link to the var's documentation
948 * @see parent::getConstLink()
949 */
950 function getConstLink($expr,$class,$package, $file = false,$text = false)
951 {
952     $a = Converter::getConstLink($expr,$class,$package,$file);

```

```
956     if (!$a) return false;
957     return $this->  returnSee($a, $text);
958 }
959
960 function setTemplateDir($dir)
961 {
962     Converter::setTemplateDir($dir);
963     $this-> smarty_dir = $this-> templateDir;
964 }
965
966 /** @return 1 always the same */
967 function getState()
968 {
969     return 1;
970 }
971
972 /**
973 * @see parent::unmangle()
974 */
975 function unmangle($notused,$source)
976 {
977 //     $source = str_replace("\n", "<br>", $source);
978 //     return $source;
979 }
980 }
981 ?>
```

# File Source for XMLDocBookpeardecor2Converter.inc

Documentation for this file is available at [XMLDocBookpeardecor2Converter.inc](#)

```
1  <?php
2  /**
3   * Outputs documentation in XML DocBook format, in the version expected by
4   * pear.php.net's documentation team
5   *
6   * phpDocumentor :: automatic documentation generator
7   *
8   * PHP versions 4 and 5
9   *
10  * Copyright (c) 2002-2006 Gregory Beaver
11  *
12  * LICENSE:
13  *
14  * This library is free software; you can redistribute it
15  * and/or modify it under the terms of the GNU Lesser General
16  * Public License as published by the Free Software Foundation;
17  * either version 2.1 of the License, or (at your option) any
18  * later version.
19  *
20  * This library is distributed in the hope that it will be useful,
21  * but WITHOUT ANY WARRANTY; without even the implied warranty of
22  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23  * Lesser General Public License for more details.
24  *
25  * You should have received a copy of the GNU Lesser General Public
26  * License along with this library; if not, write to the Free Software
27  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28  *
29  * @package Converters
30  * @subpackage XMLDocBook
31  * @author Greg Beaver <cellog@php.net>
32  * @copyright 2002-2006 Gregory Beaver
33  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version CVS: $Id: XMLDocBookpeardecor2Converter.inc 234423 2007-04-24 21:32:15Z ashnazg $
35  * @filesource
36  * @link http://www.phpdoc.org
37  * @link http://pear.php.net/PhpDocumentor
38  * @since 1.2
39  */
40 /**
41  * XML DocBook converter.
42  * This Converter takes output from the {@link Parser} and converts it to DocBook
43  * output for PEAR documentation.
44  *
45  * This Converter differs from the parent DocBook Converter in that it does not
46  * recognize the possibility of procedural pages or of functions! All functions
47  * must be defined as static methods for namespace purposes. In addition, all
48  * constants and global variables for a package are grouped together as per
49  * peardecor requirements. Include statements are not documented. If you want
50  * to document a normal project, don't use the peardecor converter, use the
51  * DocBook converter.
52  * @package Converters
53  * @subpackage XMLDocBook
54  * @author Greg Beaver <cellog@php.net>
55  * @since 1.2
56  * @version $Id: XMLDocBookpeardecor2Converter.inc 234423 2007-04-24 21:32:15Z ashnazg $
57  */
58 class XMLDocBookpeardecor2Converter extends Converter
59 {
60 /**
61  * This converter knows about the new root tree processing
62  * In order to fix PEAR Bug #6389
63  * @var boolean
64  */
```

```

65  var $processSpecialRoots = true;
66  /**
67   * XMLDocBookConverter wants elements sorted by type as well as alphabetically
68   * @see Converter::$sort_page_contents_by_type
69   * @var boolean
70   */
71  var $sort_page_contents_by_type = true;
72  /** @var string */
73  var $outputformat = 'XML';
74  /** @var string */
75  var $name = 'DocBook/peardoc2';
76  /**
77   * indexes of elements by package that need to be generated
78   * @var array
79   */
80  var $leftindex = array('classes' => true, 'pages' => false, 'functions' => false,
81  'defines' => true, 'globals' => true);
82  /**
83   * whether a @see is going to be in the {@link $base_dir}, or in a package/subpackage
84   * subdirectory of $base_dir
85   * @var boolean
86   */
87  var $local = true;
88  /**
89   * name of current page being converted
90   * @var string
91   */
92  var $page;
93  /**
94   * path of current page being converted
95   * @var string
96   */
97  var $path;
98  /**
99   * name of current class being converted
100  * @var string
101  */
102  var $class;
103  /**
104   * template for the procedural page currently being processed
105  * @var Template
106  */
107  var $page_data;
108  /**
109   * output directory for the current procedural page being processed
110  * @var string
111  */
112  var $page_dir;
113  /**
114   * Constants, used for constants.tpl
115   * @var array
116   */
117  var $peardoc2_constants = false;
118  /**
119   * Global Variables, used for globals.tpl
120   * @var array
121   */
122  var $peardoc2_globals = false;
123  /**
124   * target directory passed on the command-line.
125   * {@link $targetDir} is malleable, always adding package/ and package/subpackage/
126   * subdirectories onto it.
127   * @var string
128   */
129  var $base_dir;
130  /**
131   * output directory for the current class being processed
132   * @var string
133   */
134  var $class_dir;
135  /**
136   * output directory for the current class being processed
137   * @var string
138   */
139  var $class_dir;
140
141

```

```

142 /**
143 * template for the class currently being processed
144 * @var Template
145 */
146 var $class_data;
147 /**
148 * array of converted package page names.
149 * Used to link to the package page in the left index
150 * @var array Format: array(package => 1)
151 */
152 var $package_pages = array();
153 /**
154 * Contents of the packagename.xml file are stored in this template variable
155 * @var Smarty
156 */
157 var $packagexml;
158 /**
159 * controls formatting of parser informative output
160 *
161 * Converter prints:
162 * "Converting /path/to/file.php... Procedural Page Elements... Classes..."
163 * Since HTMLdefaultConverter outputs files while converting, it needs to send a \n to
164 * start a new line. However, if there
165 * is more than one class, output is messy, with multiple \n's just between class file
166 * output. This variable prevents that
167 * and is purely cosmetic
168 * @var boolean
169 */
170 var $juststarted = false;
171 /**
172 * contains all of the template procedural page element loop data needed for the current
173 * template
174 * @var array
175 */
176 var $current;
177 /**
178 * contains all of the template class element loop data needed for the current template
179 * @var array
180 */
181 var $currentclass;
182 /**
183 * Pass elements by package, simplifies generation of package.xml/category.xml
184 */
185 var $sort_absolutely_everything = true;
186 /**
187 * template options. Currently only 1 recognized option usepear
188 *
189 * usepear tells the getLink() function to return a package link to PEAR and PEAR_ERROR if
190 * possible, and to link directly
191 * to the fully-delimited link package#class.method or package#file.method in PEAR style,
192 * if possible, even if the
193 * package is not parsed. This will allow parsing of separate PEAR packages without
194 * parsing the entire thing at once!
195 */
196 var $template_options = array('usepear' => false);
197 /**
198 * $function_data = array();
199 * $method_data = array();
200 * $write_constants_xml = array();
201 * $write_globals_xml = array();
202 * $sourceloc = '';
203 */
204 * peardoc2 Category
205 * @var string
206 */
207 var $category;
208 /**
209 * Used to re-format output so that it's easy for translators to handle
210 *
211 * @var XML_Beautifier/false
212 * @access private
213 */
214 var $_beautifier = false;
215

```

```

216     /**
217      * sets {@link $base_dir} to $targetDir
218      * @see Converter()
219      */
220     function XMLDocBookpearDoc2Converter(& $allp, & $packp, & $classes,
221     & $procpages, $po, $pp, $qm, $targetDir, $templateDir, $title)
222     {
223         if (!class_exists('XML_Beautifier')) {
224             @include_once 'XML/Beautifier.php';
225         }
226         Converter::Converter($allp, $packp, $classes, $procpages, $po, $pp, $qm, $targetDir,
227         $templateDir, $title);
228         if (class_exists('XML_Beautifier')) {
229             require_once 'phpDocumentor/Converters/XML/DocBook/pearDoc2/Beautifier.php';
230             $this-> _beautifier = new phpDocumentor_pearDoc2_XML_Beautifier;
231             $this-> _beautifier-> setOption('indent', ' ');
232         }
233         $this-> base_dir = $targetDir;
234     }
235
236     /**
237      * do that stuff in $template_options
238      */
239     function & getLink($expr, $package = false, $packages = false)
240     {
241         return Converter::getLink($expr, $package, $packages);
242     }
243
244     function unmangle($s,$sourcecode)
245     {
246         return '<programlisting role="php"><![CDATA['
247         . $sourcecode . ']]></programlisting>' ;
248     }
249
250     /**
251      * Writes a file to target dir, beautify any .xml files first
252      * @param string filename
253      * @param string file contents
254      * @param boolean true if the data is binary and not text
255      */
256     function writeFile($file,$data,$binary = false)
257     {
258         if ($this-> _beautifier && substr($file, -4) == '.xml') {
259             $ret = $this-> _beautifier-> formatString($data);
260             if (PEAR::isError($ret)) {
261                 addWarning(PDERROR_BEAUTIFYING_FAILED, $ret-> getMessage());
262                 $ret = $data;
263             }
264             $data = $ret;
265         }
266         return parent::writeFile($file, $data, $binary);
267     }
268
269     /**
270      * Used to convert the {@example} inline tag in a docblock.
271      *
272      * By default, this just wraps ProgramExample
273      * @see XMLDocBookpearDoc2Converter::exampleProgramExample
274      * @param string
275      * @param boolean true if this is to highlight a tutorial <programlisting>
276      * @return string
277      */
278     function exampleProgramExample($example, $tutorial = false, $inlinesourceparse =
279     null/*false*/,
280                               $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
281     null/*false*/)
282     {
283         return '<example><title>Example</title><programlisting
284         role="php"><![CDATA['
285         . $example . ']]></programlisting></example>' ;
286         $this-> ProgramExample($example, $tutorial, $inlinesourceparse, $class, $linenum,
287         $filesourcepath)
288         . '</example>' ;
289     }
290
291     function writeExample($title, $path, $source)
292     {
293         $this-> _save_example = array($title, $source);
294     }

```

```

290     function getExampleLink($unused, $title)
291     {
292         $source = $this-> _save_example[1];
293         return '<para><example><title>' . $title . '</title>' . $source
294     }
295
296     function type_adjust($typename)
297     {
298         if (isset($this-> template_options['typechanging'][trim($typename)]))
299             return $this-> template_options['typechanging'][trim($typename)];
300         $a = $this-> getLink($typename);
301         if (is_object($a))
302         {
303             if (phpDocumentor_get_class($a) == 'classlink')
304                 return '<classname>' . $typename . '</classname>';
305             if (phpDocumentor_get_class($a) == 'functionlink' || phpDocumentor_get_class($a) ==
306                 'methodlink')
307                 return '<function>' . $typename . '</function>';
308             if (phpDocumentor_get_class($a) == 'definelink')
309                 return '<constant>' . $typename . '</constant>';
310             if (phpDocumentor_get_class($a) == 'varlink')
311                 return '<varname>' . $typename . '</varname>';
312         }
313         return $typename;
314     }
315
316     /**
317      * Writes out the template file of {@link $class_data} and unsets the template to save
318      * memory
319      * @see registerCurrentClass()
320      * @see parent::endClass()
321      * @todo move class summary into an array to be written out at the end
322      *       of parsing each package
323      */
324     function endClass()
325     {
326         $a = '.../';
327         if (!empty($this-> subpackage)) $a .= '/';
328         if ($this-> juststarted)
329         {
330             $this-> juststarted = false;
331             phpDocumentor_out("\n");
332             flush();
333         }
334         foreach($this-> method_data as $func)
335         {
336             $func[0]-> assign("phpdocversion" , PHPDOCUMENTOR_VER);
337             $func[0]-> assign("phpdocwebsite" , PHPDOCUMENTOR_WEBSITE);
338             $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . strtolower($this-
339             > category) . PATH_DELIMITER . strtolower($this-> class_dir) . PATH_DELIMITER .
340             str_replace(array('_','.'),array('--','--'),$this-> class));
341             $this-> writefile(strtolower($func[1]) . '.xml','!-- $' . "Revision$ --"
342             . $func[0]-> fetch('method.tpl'));
343         }
344         // code below is in packagename.xml handling, see Output()
345         /* $this->setTargetDir($this->base_dir . PATH_DELIMITER . strtolower($this-
346         >category) . PATH_DELIMITER . strtolower($this->class_dir));
347         $this->writefile(str_replace(array('_','.'),array('--','--'),strtolower($this-
348         >class)) . '.xml',$this->class_data->fetch('class.tpl'));*/
349         unset($this-> class_data);
350     }
351
352     function addSummaryToPackageXml($template_output)
353     {
354         $this-> packagexml-> append('ids',$template_output);
355     }
356
357     /**
358      * @param parserClass/false $element is false if this is the end of all conversion
359      */
360     function flushPackageXml($element)
361     {
362         if (isset($this-> packagexml))
363         {
364             if (!$element || $element-> docblock-> package != $this-> package) ///
365             finished with package
366             {
367                 if (isset($this-> _write_constants_xml[$this-> category][$this-
368                 > package])) &&

```

```

360             $this-> _write_constants_xml[$this-> category][$this-> package]);
361         {
362             $this-> packagexml-> append('ids',
363                 '&package.' . strtolower($this-> category) . ' .
364                 str_replace(array('_','.'),array('-','--'),$this-
365 > package).'.constants');");
366             $this-> _write_constants_xml[$this-> category][$this-> package] =
367             false;
368         }
369         if (isset($this-> _write_globals_xml[$this-> category][$this-> package]))
370         {
371             $this-> _write_globals_xml[$this-> category][$this-> package]);
372             $this-> packagexml-> append('ids',
373                 '&package.' . strtolower($this-> category) . ' .
374                 str_replace(array('_','.'),array('-','--'),$this-
375 > package).'.globals');");
376             $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . strtolower($this-
377 > category));
378             $this-> writefile(str_replace('_', '-', strtolower($this-> package)).'.xml',
379                 '<!-- $' . "Revision$ -->\n" . $this-> packagexml-
380 > fetch('package.tpl'));
381             $this-> packagexml-> clear_all_assign();
382             if ($element)
383             {
384                 $this-> packagexml-> assign('package',$element-> docblock-
385 > package);
386                 $this-> packagexml-> assign('ids',array());
387                 $this-> packagexml-> assign('id',$this-> getId($element, true));
388             }
389         }
390     }
391     /**
392      * @param string
393      * @param string
394      * @return string <ulink url='".$link."'>.$text.</ulink>
395      */
396     function returnLink($link,$text)
397     {
398         return '<ulink url="' . $link . '">' . $text . '</ulink>';
399     }
400     function makeLeft()
401     {
402     }
403     /**
404      * Does nothing
405      */
406     function formatPkgIndex()
407     {
408     }
409     /**
410      * Does nothing
411      */
412     function formatIndex()
413     {
414     }
415     /**
416      * Does nothing
417      */
418     function writeNewPPage($key)
419     {
420     }
421     /**
422      * Does nothing
423      */
424     function writeNewPPage($key)
425     {
426     }
427     /**
428      * Does nothing
429      */
430     /**
431      * Does nothing
432      */

```

```

433     function writeSource()
434     {
435     }
436
437     /**
438      * Creates package/lang/categoryname/packagename.xml for each package
439      */
440     function formatLeftIndex()
441     {
442         $this-> makeLeft();
443     }
444
445     /**
446      * This function takes an {@link abstractLink} descendant and returns an html link
447      *
448      * @param abstractLink a descendant of abstractlink should be passed, and never text
449      * @param string text to display in the link
450      * @param boolean this parameter is not used, and is deprecated
451      * @param boolean determines whether the returned text is enclosed in an <link> tag
452      */
453     function returnSee(& $element, $eltext = false, $local = true, $with_a = true)
454     {
455         if (!$element) return false;
456         if (!$eltext)
457         {
458             $eltext = '';
459             switch($element-> type)
460             {
461                 case 'tutorial' :
462                     $eltext = $element-> title;
463                     break;
464                 case 'class' :
465                     $eltext = '<classname>' . $element-> name . '</classname>';
466                     break;
467                 case 'method' :
468                     $eltext .= '<function>';
469                     case 'var' :
470                         if ($element-> type == 'var') $eltext .= '<varname>';
471                         $eltext .= $element-> class . '::';
472                     case 'page' :
473                     case 'define' :
474                         if ($element-> type == 'define')
475                             $eltext .= '<constant>';
476                     case 'function' :
477                         if ($element-> type == 'function')
478                             $eltext .= '<function>';
479                     case 'global' :
480                     default :
481                         $eltext .= $element-> name;
482                         if ($element-> type == 'function' || $element-> type == 'method') $eltext
483                         .= '</function>';
484                         if ($element-> type == 'var') $eltext .= '</varname>';
485                         if ($element-> type == 'define') $eltext .= '</constant>';
486                         break;
487                     } elseif (!is_object($element)) {
488                         return false;
489                     } elseif ($element-> type == 'method')
490                     {
491                         $eltext = str_replace($element-> name . '()', $element-> name, $eltext);
492                     }
493
494                     if ($element-> type == 'page' || $element-> type == 'function' || $element-> type
495 == 'var')
496                     { // we ignore all procedural pages, instead, constant, function and
497                     // global variable pages are output
498                         return $eltext;
499                     } if ($element-> type == 'class')
500                     {
501                         return '<link linkend=""' . $this-> getId($element) . '-'
502                         . $eltext . '</link>';
503                     }
504                     return '<link linkend=""' . $this-
505 > getId($element) . '">' . $eltext . '</link>';
506
507     /**
508      * Get the id value needed to allow linking
509      * @param mixed descendant of parserElement or parserData/parserPage

```

```

509     * @param boolean true to return the id for the package page
510     * @see parserElement, parserData, parserPage
511     * @return string the id value for this element type
512     */
513     function getId(& $sel, $returnpackage = false)
514     {
515         if (phpDocumentor_get_class($sel) == 'parserdata')
516         {
517             $element = $this-> addLink($sel-> parent);
518             $elp = $sel-> parent;
519         } elseif (!is_a($sel, 'abstractlink'))
520         {
521             $elp = $sel;
522             $element = $this-> addLink($sel);
523         } else $element = $sel;
524         $a = '';
525         if (!empty($element-> subpackage))
526         {
527             $a = str_replace(array('_','.'),array('-','--'),$element-> subpackage).'.';
528         }
529         if ($returnpackage) return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package));
530         switch ($element-> type)
531         {
532             case 'page' :
533                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.$a.$element-
> fileAlias);
534                 break;
535             case 'define' :
536                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.constants.details.'.$element-> fileAlias);
537                 break;
538             case 'global' :
539                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.globals.details.'.$element-> fileAlias);
540                 break;
541             case 'class' :
542                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.$a.str_replace(array('_','.'),array('-','--'),$element-> name));
543                 break;
544             case 'function' :
545                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.$a.$element-
> fileAlias.'.str_replace('_','-',$element-> name));
546                 break;
547             case 'method' :
548                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.$a.str_replace(array('_','.'),array('-','--'),$element-> class).'.'.str_replace('_','-',$element-> name));
549                 break;
550             case 'var' :
551                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.$a.str_replace(array('_','.'),array('-','--'),$element-> class).'.'.summary.vars.'str_replace(array('$','_'),array('var--','_'),$element-> name));
552                 break;
553             case 'tutorial' :
554                 return 'package.'.strtolower($element-
> category.'.str_replace(array('_','.'),array('-','--'),$element-> package).'.'.$a.str_replace(array('_','.'),array('-','--'),$element-> name)).'-tutorial';
555                 break;
556         }
557     }
558     /**
559      * Create errors.html template file output
560      *
561      * This method takes all parsing errors and warnings and spits them out ordered by file and
562      * line number.
563      * @global ErrorTracker We'll be using it's output facility
564      */
565     function ConvertErrorLog()
566     {
567         global $phpDocumentor_errors;
568         $allfiles = array();

```

```

569     $files = array();
570     $warnings = $phpDocumentor_errors->    returnWarnings();
571     $errors = $phpDocumentor_errors->    returnErrors();
572     $template = & $this-> newSmarty();
573     foreach($warnings as $warning)
574     {
575         $file = '##none';
576         $linenum = 'Warning';
577         if ($warning-> file)
578         {
579             $file = $warning-> file;
580             $allfiles[$file] = 1;
581             $linenum .= ' on line '.$warning-> linenum;
582         }
583         $files[$file]['warnings'][] = array('name' => $linenum, 'listing' => $warning-
> data);
584     }
585     foreach($errors as $error)
586     {
587         $file = '##none';
588         $linenum = 'Error';
589         if ($error-> file)
590         {
591             $file = $error-> file;
592             $allfiles[$file] = 1;
593             $linenum .= ' on line '.$error-> linenum;
594         }
595         $files[$file]['errors'][] = array('name' => $linenum, 'listing' => $error-
> data);
596     }
597     $i=1;
598     $af = array();
599     foreach($allfiles as $file => $num)
600     {
601         $af[$i++] = $file;
602     }
603     $allfiles = $af;
604     usort($allfiles,'strnatcasecmp');
605     $allfiles[0] = "Post-parsing";
606     foreach($allfiles as $i => $a)
607     {
608         $allfiles[$i] = array('file' => $a);
609     }
610     $out = array();
611     foreach($files as $file => $data)
612     {
613         if ($file == '##none') $file = 'Post-parsing';
614         $out[$file] = $data;
615     }
616     $template-> assign("files" , $allfiles);
617     $template-> assign("all" , $out);
618     $template-> assign("title" , "phpDocumentor Parser Errors and
Warnings");
619     $this-> setTargetDir($this-> base_dir);
620     $this-> writefile("errors.html" , $template-> fetch('errors.tpl'));
621     unset($template);
622     phpDocumentor_out("\n\nTo view errors and warnings, look at "
.PATH_DELIMITER . "errors.html\n");
623     flush();
624 }
625
626 function postProcess($text)
627 {
628     return str_replace("'" , '&apos;' , htmlentities($text));
629 }
630
631 function prepareDocBlock(& $element, $nopackage = true)
632 {
633     $a = new parserStringWithInlineTags;
634     $a-> add('no exceptions thrown');
635     if (!$element-> docblock-> getKeyword('throws')) $element-> docblock-
> addKeyword('throws',$a);
636     $tags = parent::prepareDocBlock($element,
637         array('staticvar' => 'note','deprec' => 'deprecated',
638               'abstract' => 'abstract','TODO' => 'note','link' => 'see',
639               'uses' => 'see','usedby' => 'see','tutorial' => 'see',
640               'return' => 'returns','access' => false), $nopackage);
641     $ret = array();
642     foreach($tags['tags'] as $tag)
643     {

```

```

644         if ($tag['keyword'] == 'return')
645         {
646             // hack because stupid Converter isn't doing its job
647             $tag['keyword'] = 'returns';
648         }
649         $ret[$tag['keyword']][] = $tag;
650     }
651     $tags['tags'] = $ret;
652     $tags['sdesc'] = $this-> wordwrap($tags['sdesc']);
653     return $tags;
654 }
655
656 function getTutorialId($package,$subpackage,$tutorial,$id,$category)
657 {
658     $subpackage = (empty($subpackage) ? '' : '.'.$subpackage);
659     $id = (empty($id) ? '' : '.'.$id);
660     return
'package.'.strtolower($category.'.'.$package.$subpackage.str_replace(array('_','.'),array('-','--'),$tutorial).$id);
661 }
662
663 /**
664 * Retrieve a Converter-specific anchor to a segment of a source code file
665 * parsed via a {@tutorial tags.filesource.pkg} tag.
666 *
667 * NOTE: unused
668 * @param string full path to source file
669 * @param string name of anchor
670 * @param string link text, if this is a link
671 * @param boolean returns either a link or a destination based on this
672 *           parameter
673 * @return string link to an anchor, or the anchor
674 */
675 function getSourceAnchor($sourcefile,$anchor,$text = '',$link = false)
676 {
677     return '';
678 }
679
680 function Br($input)
681 {
682     return "    $input\n";
683 }
684
685 function getCData($value)
686 {
687     return '<![CDATA[' . $value .']]>    ';
688 }
689
690
691 function ProgramExample($listing, $tutorial = false, $inlinesourceparse = null/*false*/,
692                         $class = null/*false*/, $linenum = null/*false*/, $filesourcepath =
null/*false*/, $origsource = null)
693 {
694     if ($origsource !== null) {
695         $listing = $origsource;
696     }
697     if (!tokenizer_ext)
698     {
699         $listing = $this-> getCData($listing);
700     }
701     return '<programlisting role="php">' . $this-> getCData($listing) .
'</programlisting>    ';
702 }
703
704 /**
705 * Does nothing - use tutorials for DocBook
706 * @param parserPackagePage
707 */
708 function convertPackagePage(& $element)
709 {
710 }
711
712 /**
713 * Convert tutorials for output
714 * @param parserTutorial
715 */
716 function convertTutorial(& $element)
717 {
    $template = & parent::convertTutorial($element);
    phpDocumentor_out("\n");
}

```

```

720
721     flush();
722     $x = $element-> Convert($this, false);
723     if ($element-> ini)
724     { // add child tutorial list to the tutorial through a slight hack : )
725         $subtutorials = '';
726         $b = '';
727         if (!empty($element-> subpackage)) $b = '.' . $element-> subpackage;
728         foreach($element-> ini['Linked Tutorials'] as $child)
729         {
730             $subtutorials .= '        &' . $element-> category . '.' . $element-
> package.$b . '.' . str_replace(array('_', '.'), array('-', '--'), $child) . '-' . $element-
> tutorial_type . "-tutorial;\n";
731         }
732         $x = str_replace('</refsect1></refentry>', '</refsect1>',
733                         <refsect1>
734                         <title>Related Docs</title>
735                         <para>
736                         .' . $subtutorials .
737                         </para>
738                         </refsect1></refentry>' , $x);
739         $template-> assign('contents', $x);
740         $contents = $template-> fetch('tutorial.tpl');
741         $a = '';
742         if ($element-> subpackage) $a = PATH_DELIMITER . $element-> subpackage;
743         phpDocumentor_out("\n");
744         flush();
745         $this-> setTargetDir($this-> base_dir . PATH_DELIMITER .
str_replace(array('_', '.'), array('-', '--'), strtolower($element-> category)))
746             . PATH_DELIMITER . strtolower(str_replace(array('_', '.'), array('-', '--'), $element-
> package)) . $a));
747         $this-> writeFile(str_replace(array('_', '.'), array('-', '--'), strtolower($element-
> name)) . "-tutorial.xml",
748             '<!-- $' . "Revision$ -->\n" . $contents);
749     }
750
751     /**
752      * Does nothing in this converter
753      * @param parserVar
754      */
755     function convertVar(& $element)
756     {
757         return;
758         $docblock = $this-> prepareDocBlock($element);
759         $b = 'mixed';
760         if ($element-> docblock-> var)
761         {
762             $b = $element-> docblock-> var-> converted_returnType;
763         }
764         // var_dump($this->getFormattedOverrides($element));
765         if (isset($this-> template_options['separatepage']) && $this-
> template_options['separatepage'])
766             $this-> class_summary-> append('vars', array('sdesc' => $docblock['sdesc'],
767                                                 'desc' => $docblock['desc'],
768                                                 'tags' => $docblock['tags'],
769                                                 'var_name' => $this-> type_adjust($element-
> getName()),
770                                                 'var_default' => htmlspecialchars($element-
> getValue()),
771                                                 'var_type' => $b,
772                                                 'var_overrides' => $this-
773                                                 'line_number' => $element-> getLineNumber(),
774                                                 'id' => $this-> getId($element)));
775         else
776             $this-> class_data-> append('vars', array('sdesc' => $docblock['sdesc'],
777                                                 'desc' => $docblock['desc'],
778                                                 'tags' => $docblock['tags'],
779                                                 'var_name' => $this-> type_adjust($element-
> getName()),
780                                                 'var_default' => htmlspecialchars($element-
> getValue()),
781                                                 'var_type' => $b,
782                                                 'var_overrides' => $this-
783                                                 'line_number' => $element-> getLineNumber(),
784                                                 'id' => $this-> getId($element));
785     }
786
787     /**

```

```

788     * Converts class for template output
789     * @param parserClass
790     * @uses flushPackageXml() creates packagename.xml file when all classes in
791     *       a package have been converted
792     */
793     function convertClass(& $element)
794     {
795         $this-> flushPackageXml($element);
796         parent::convertClass($element);
797         $docblock = $this-> prepareDocBlock($element);
798         $this-> method_data = array();
799         $this-> class_dir = str_replace(array('_', '.'), array('-', '--'), $element-> docblock-
> package);
800         $this-> package = $element-> docblock-> package;
801         $this-> category = strtolower($element-> docblock-> category);
802         if (!empty($element-> docblock-> subpackage)) $this-> class_dir .= PATH_DELIMITER
. $element-> docblock-> subpackage;
803         $docblock = $this-> prepareDocBlock($element, false);
804         $this-> class_data-> assign("sdesc" , $docblock['sdesc']);
805         $this-> class_data-> assign("desc" , $docblock['desc']);
806         $this-> class_data-> assign("tags" , $docblock['tags']);
807
808         $this-> class_data-> assign("source_location" , $element-
> getSourceLocation($this, $this-> template_options['usepear']));
809         $this-> class_data-> assign("id" , $this-> getId($element));
810         $this-> class_data-> assign("method_ids" , array());
811         $this-> left[$this-> package][] = array('link' => $this-> getId($element). '-'
summary');
812         if ($t = $element-> getTutorial())
813         {
814             $this-> class_data-> append("method_ids" , $this-> getId($t));
815         }
816
817         if (isset($this-> template_options['separatepage']) && $this-
> template_options['separatepage'])
818         {
819             $this-> class_summary = & $this-> newSmarty(true);
820             if ($t = $element-> getTutorial())
821             {
822                 $this-> class_summary-> assign("tutorial" , $this-
> returnSee($t));
823             }
824
825             $this-> class_summary-> assign("class_name" , $this-
> type_adjust($element-> getName()));
826             $this-> class_summary-> assign("sdesc" , $docblock['sdesc']);
827             $this-> class_summary-> assign("desc" , $docblock['desc']);
828             $this-> class_summary-> assign("tags" , $docblock['tags']);
829             $this-> class_summary-> assign("vars" , array());
830             $this-> class_summary-> assign("methods" , array());
831             $this-> class_summary-> assign("package" , $element-> docblock-
> package);
832
833             $this-> class_summary-> assign("children" , $this-
> generateChildClassList($element));
834             $this-> class_summary-> assign("class_tree" , $this-
> generateFormattedClassTree($element));
835             $this-> class_summary-> assign("conflicts" , $this-
> getFormattedConflicts($element, "classes"));
836
837             $this-> class_summary-> assign("source_location" , $element-
> getSourceLocation($this, $this-> template_options['usepear']));
838             $this-> class_summary-> assign("id" , $this-> getId($element). '-'
summary');
839             $this-> class_data-> append("method_ids" , $this-
> getId($element). '.' . strtolower(str_replace('_', '-' , $element-> getName())) . '-summary');
840             $inherited_methods = $this-> getFormattedInheritedMethods($element);
841             if (!empty($inherited_methods))
842             {
843                 $this-> class_summary-> assign("imethods" , $inherited_methods);
844             }
845             $inherited_vars = $this-> getFormattedInheritedVars($element);
846             // variables are irrelevant in peardoc
847             if (false) // !empty($inherited_vars)
848             {
849                 $this-> class_summary-> assign("ivars" , $inherited_vars);
850             }
851             $this-> addSummaryToPackageXml($this-> class_summary-
> fetch('class_summary.tpl'));
852         }

```

```

853     $this-> sourceloc = $element-> getSourceLocation($this,$this-
> template_options['usepear']);
854 }
855
856 /**
857 * Converts method for template output
858 * @see prepareDocBlock(), parserMethod::getFunctionCall(), getFormattedDescMethods(),
859 *      getFormattedOverrides()
860 */
861 function convertMethod(& $element)
862 {
863     $docblock = $this-> prepareDocBlock($element);
864     $returntype = 'void';
865     if ($element-> docblock-> return)
866     {
867         $a = $element-> docblock-> return-> Convert($this);
868         $returntype = $element-> docblock-> return-> converted_returnType;
869         if ($returntype != $element-> docblock-> return-> returnType)
870         {
871             $returntype = "    <replaceable>    $returntype</replaceable>    ";
872         }
873     }
874     $params = array();
875     if (count($element-> docblock-> params))
876         foreach($element-> docblock-> params as $param => $val)
877     {
878         $a = $val-> Convert($this);
879         $b = explode(' ', $a);
880         $c = '';
881         foreach($b as $blah) {
882             if (!empty($c)) {
883                 $c .= ' ';
884             }
885             $c .= str_replace(array('true', 'false', 'null'), array('&true;', '&false;', '&null;'), $blah);
886         }
887         $params[$param] = array("var" => $param, "datatype" => str_replace(array('true', 'false', 'null'), array('&true;', '&false;', '&null;'), $val-> returnType), "cdatatype" => $val-> dataType);
888     }
889     $converted_returnType = "data" => $this-> wordwrap($c));
890 }
891
892     $call = $element-> getIntricateFunctionCall($this, $params);
893     if (isset($call['params']))
894     {
895         foreach($call['params'] as $i => $param)
896         {
897             if (!is_string($call['params'][$i]['default']))
898             {
899                 continue;
900             }
901             $call['params'][$i]['default'] = str_replace(array('true', 'false', 'null'), array('&true;', '&false;', '&null;'), $param['default']);
902         }
903     }
904     $this-> packagexml-> append('ids','&' . $this-> getId($element). '/');
905     $this-> class_data-> append('method_ids',$this-> getId($element));
906     $this-> class_summary-> append('methods',array('id' => $this-> getId($element),
907                                                 'sdesc' => $docblock['sdesc'],
908                                                 'desc' => $docblock['desc'],
909                                                 'tags' => $docblock['tags'],
910                                                 'is_constructor' => $element-
911                                                 'function_name' => $element-> getName(),
912                                                 'function_return' => $returntype,
913                                                 'function_call' => $call,
914                                                 'descmethod' => $this-
915                                                 'method_overrides' => $this-
916                                                 'line_number' => $element-
917                                                 'params' => $params));
918     $this-> method_data[$i = count($this-> newSmarty(true));
919     $this-> method_data[$i][1] = str_replace(array('_', '.'),array('-', '--'),$element-
920     getName()));
921     $this-> method_data[$i][0]-> assign('class',$this-> class);
922     $this-> method_data[$i][0]-> assign('source_location',$this-> returnSee($this-
```

```

> getLink(basename($this-> curpage-> getFile()),$this-> sourceloc));
921     $this-> method_data[$i][0]-> assign('sdesc',$docblock['sdesc']);
922     $this-> method_data[$i][0]-> assign('desc',$docblock['desc']);
923     $this-> method_data[$i][0]-> assign('tags',$docblock['tags']);
924     $this-> method_data[$i][0]-> assign('function_name',$element-> getName());
925     $this-> method_data[$i][0]-> assign('function_return',$returntype);
926     $this-> method_data[$i][0]-> assign('function_call',$call);
927     $this-> method_data[$i][0]-> assign('descmethod',$this-
> getFormattedDescMethods($element));
928     $this-> method_data[$i][0]-> assign('method_overrides',$this-
> getFormattedOverrides($element));
929     $this-> method_data[$i][0]-> assign('params',$params);
930     $this-> method_data[$i][0]-> assign('id',$this-> getId($element));
931 }
932 /**
933 * Converts function for template output - does nothing in peardoc2!
934 * @param parserFunction
935 */
936 function convertFunction(& $element)
937 {
938     /* parent::convertFunction($element);
939     $docblock = $this->prepareDocBlock($element);
940     $fname = $element->getName();
941     $params = array();
942     if (count($element->docblock->params))
943         foreach($element->docblock->params as $param => $val)
944     {
945         $a = $val->Convert($this);
946         $params[$param] = array("var" => $param,"datatype" =>
947         $val->converted_returnType,"data" => $a);
948     }
949     $returntype = 'void';
950     if ($element->docblock->return)
951     {
952         $a = $element->docblock->return->Convert($this);
953         $returntype = $element->docblock->return->converted_returnType;
954     }
955
956     $this->page_data->append("function_ids",$this->getId($element));
957     $this->page_summary->append("function_ids",$this->getId($element));
958     $this->page_summary->append('functions',array('id' => $this-
>getId($element),
959                                         'sdesc' => $docblock['sdesc'],
960                                         'desc' => $docblock['desc'],
961                                         'tags' => $docblock['tags'],
962                                         'function_name' => $element->getName(),
963                                         'line_number' => $element-
>getLineNumber(),
964                                         'function_return' => $returntype,
965                                         'function_call' => $element-
>getIntricateFunctionCall($this,$params),
966                                         'function_conflicts' => $this-
>getFormattedConflicts($element,'functions'),
967                                         'params' => $params));
968     $this->function_data[$i = count($this->function_data) - 1][0] = $this-
>newSmarty(true);
969     $this->function_data[$i][1] = $element->getName();
970     $this->function_data[$i][0]->assign('sdesc',$docblock['sdesc']);
971     $this->function_data[$i][0]->assign('desc',$docblock['desc']);
972     $this->function_data[$i][0]->assign('tags',$docblock['tags']);
973     $this->function_data[$i][0]->assign('function_name',$fname);
974     $this->function_data[$i][0]->assign('line_number',$element->getLineNumber());
975     $this->function_data[$i][0]->assign('function_return',$returntype);
976     $this->function_data[$i][0]->assign('function_call',$element-
>getIntricateFunctionCall($this,$params));
977     $this->function_data[$i][0]->assign('function_conflicts',$this-
>getFormattedConflicts($element,"functions"));
978     $this->function_data[$i][0]->assign('params',$params);
979     $this->function_data[$i][0]->assign('source_location',$this->returnSee($this-
>getLink(basename($this->curpage->getFile()),$this->sourceloc)));
980     $this->function_data[$i][0]->assign('id',$this->getId($element));*/
981 }
982 /**
983 * Converts include elements for template output
984 *
985 * Completely ignored by this converter
986 * @param parserInclude
987 */

```

```

989     function convertInclude(& $element)
990     {
991         /*      parent::convertInclude($element, array('include_file' => '-' . strtr($element-
992 >getValue(),array('' => '', '"' => '' , '.' => '-'))));
993         $docblock = $this->prepareDocBlock($element);
994         $per = $this->getIncludeValue($element->getValue(), $element->getPath());
995         $this->page_summary->append('includes',array('sdesc' => $docblock['sdesc'],
996                                         'desc' => $docblock['desc'],
997                                         'tags' => $docblock['tags'],
998                                         'utags' => $docblock['utags'],
999                                         'include_name' => $element-
1000                                         'getName(),
1001                                         'include_value' => $per,
1002                                         'line_number' => $element-
1003                                         'getLineNumber(),
1004                                         'include_file' => '-' . strtr($element-
1005 >getValue(),array('' => '', '"' => '' , '.' => '-')));*/
1006     }
1007
1008 /**
1009  * Converts defines for template output
1010  * @see prepareDocBlock(), getFormattedConflicts()
1011  * @param parserDefine
1012  */
1013 function convertDefine(& $element)
1014 {
1015     $docblock = $this-> prepareDocBlock($element);
1016     $this-> _appendDefines(array('sdesc' => $docblock['sdesc'],
1017                               'desc' => $docblock['desc'],
1018                               'tags' => $docblock['tags'],
1019                               'name' => $this-> postProcess($element-
1020 > getName()),
1021                               'value' => $this-> postProcess($element-
1022 > getValue()),
1023                               'conflicts' => $this-
1024 > getFormattedConflicts($element,"defines"),
1025                               'line_number' => $element-> getLineNumber(),
1026                               'id' => $this-> getId($element));
1027 }
1028
1029 /**
1030  * Append the constant information to the Smarty information
1031  *
1032  * Uses category, package, and current file to organize constants defined
1033  * in a package for the constants.xml output file
1034  * @param array
1035  * @uses $_peardoc2_constants appends $define to them
1036  * @access private
1037  */
1038 function _appendDefines($define)
1039 {
1040     if (!isset($this-> _peardoc2_constants[$this-> category][$this-> package][$this-
1041 > sourceloc]))
1042     {
1043         $this-> _peardoc2_constants[$this-> category][$this-> package][$this-
1044 > sourceloc]['name'] =
1045             $this-> sourceloc;
1046         $this-> _peardoc2_constants[$this-> category][$this-> package][$this-
1047 > sourceloc]['page'] =
1048             $this-> page;
1049         }
1050         $this-> write_constants_xml[$this-> category][$this-> package] = true;
1051         $this-> _peardoc2_constants[$this-> category][$this-> package][$this-
1052 > sourceloc]['defines'][] = $define;
1053     }
1054
1055 /**
1056  * Converts global variables for template output
1057  * @param parserGlobal
1058  * @see prepareDocBlock(), getFormattedConflicts()
1059  */
1060 function convertGlobal(& $element)
1061 {
1062     $docblock = $this-> prepareDocBlock($element);
1063     $value = $this-> getGlobalValue($element-> getValue());
1064     if ($value == $element-> getValue())
1065     {
1066         $value = $this-> programExample($value);
1067     } else
1068     {

```

```

1058     $value = $this-> _getGlobalValue('<! [CDATA[ ' . $element-> getValue() . '
1059   );
1060   }
1061   $this-> _appendGlobals(array('sdesc' => $docblock['sdesc'],
1062     'desc' => $docblock['desc'],
1063     'tags' => $docblock['tags'],
1064     'name' => $this-> postProcess($element-
1065       > getName())),
1066     'link' => $element-> getName(),
1067     'value' => $value,
1068     'type' => $element-> getDataType($this),
1069     'line_number' => $element-> getLineNumber(),
1070     'conflicts' => $this-
1071     > getFormattedConflicts($element,"global variables"),
1072     'id' => $this-> getId($element));
1073   }
1074
1075 /**
1076  * Append the global variable information to the Smarty information
1077  *
1078  * Uses category, package, and current file to organize globals defined
1079  * in a package for the globals.xml output file
1080  * @param array
1081  * @uses $_peardoc2_globals appends $global to them
1082  * @access private
1083  */
1084 function _appendGlobals($global)
1085 {
1086   if (!isset($this-> _peardoc2_globals[$this-> category][$this-> package][$this-
1087     > sourceloc]))
1088   {
1089     $this-> _peardoc2_globals[$this-> category][$this-> package][$this-
1090     > sourceloc]['name'] =
1091     $this-> sourceloc;
1092     $this-> _peardoc2_globals[$this-> category][$this-> package][$this-
1093     > sourceloc]['page'] =
1094     $this-> page;
1095     $this-> _write_globals_xml[$this-> category][$this-> package] = true;
1096     $this-> _peardoc2_globals[$this-> category][$this-> package][$this-
1097     > sourceloc]['globals'][] = $global;
1098   }
1099
1100 /**
1101  * converts procedural pages for template output
1102  * @see prepareDocBlock(), getClassessOnPage()
1103  * @param parserData
1104  */
1105 function convertPage(& $element)
1106 {
1107   parent::convertPage($element);
1108   $this-> juststarted = true;
1109   $this-> page_dir = $element-> parent-> package;
1110   $this-> page = $this-> getPageName($element-> parent);
1111   $this-> category = strtolower($element-> parent-> category);
1112   $this-> sourceloc = $element-> parent-> getSourceLocation($this,true);
1113   if (!empty($element-> parent-> subpackage)) $this-> page_dir .= PATH_DELIMITER .
1114   $element-> parent-> subpackage;
1115   // registering stuff on the template
1116
1117 /**
1118  * returns an array containing the class inheritance tree from the root object to the class
1119  * @param parserClass    class variable
1120  * @return array Format: array(root,child,child,child,...,$class)
1121  * @uses parserClass::getParentClassTree()
1122  */
1123
1124 function generateFormattedClassTree($class)
1125 {
1126   $tree = $class-> getParentClassTree($this);
1127   $out = '';
1128   if ($count($tree) - 1)

```

```

1129     {
1130         $result = array($class-> getName());
1131         $parent = $tree[$class-> getName()];
1132         while ($parent)
1133         {
1134             if (is_string($parent)) {
1135                 $result[] = $parent;
1136                 break;
1137             }
1138             $subpackage = $parent-> docblock-> subpackage;
1139             $package = $parent-> docblock-> package;
1140             $x = $parent;
1141             if (is_object($parent))
1142                 $x = $parent-> getLink($this);
1143             if (!$x) $x = $parent-> getName();
1144             $result[] =
1145                 $x;
1146             if (is_object($parent))
1147                 $parent = $tree[$parent-> getName()];
1148             elseif (isset($tree[$parent]))
1149                 $parent = $tree[$parent];
1150             }
1151             return array_reverse($result);
1152         } else
1153         {
1154             return array($class-> getName());
1155         }
1156     }
1157
1158 /**
1159 * returns a list of child classes
1160 *
1161 * @param parserClass class variable
1162 * @uses parserClass::getChildClassList()
1163 */
1164
1165 function generateChildClassList($class)
1166 {
1167     $kids = $class-> getChildClassList($this);
1168     $list = array();
1169     if (count($kids))
1170     {
1171         for($i=0; $i< count($kids); $i++)
1172         {
1173             $lt['link'] = '<link linkend=" ' . $this-> getId($kids[$i]) . ' -' .
summary">' . $kids[$i]-> getName() . '</link> ' ;
1174             $lt['sdesc'] = $kids[$i]-> docblock-> getSDesc($this);
1175             $list[] = $lt;
1176         }
1177     } else return false;
1178     return $list;
1179 }
1180
1181 /** @access private */
1182 function sortVar($a, $b)
1183 {
1184     return strnatcasecmp($a-> getName(), $b-> getName());
1185 }
1186
1187 /** @access private */
1188 function sortMethod($a, $b)
1189 {
1190     if ($a-> isConstructor) return -1;
1191     if ($b-> isConstructor) return 1;
1192     return strnatcasecmp($a-> getName(), $b-> getName());
1193 }
1194
1195 /**
1196 * returns a template-enabled array of class trees
1197 *
1198 * @param string $package package to generate a class tree for
1199 * @see $roots, HTMLConverter::getRootTree()
1200 */
1201 function generateFormattedClassTrees($package)
1202 {
1203     if (!isset($this-> roots['normal'][$package]) &&
1204         !isset($this-> roots['special'][$package])) {
1205         return array();
1206     }
1207     $trees = array();

```

```

1208     if (isset($this-> roots['normal'][$package])) {
1209         $roots = $this-> roots['normal'][$package];
1210         for($i=0;$i< count($roots);$i++)
1211         {
1212             $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1213             if ($root && $root-> isInterface()) {
1214                 continue;
1215             }
1216             $trees[] = array('class' => $roots[$i], 'class_tree' =>
1217             . $this-> getRootTree($this-
1218             > getSortedClassTreeFromClass($roots[$i],$package,''),$package). "</ul>\n"
1219         }
1220     if (isset($this-> roots['special'][$package])) {
1221         $roots = $this-> roots['special'][$package];
1222         foreach ($roots as $parent => $classes) {
1223             $thistree = '';
1224             foreach ($classes as $classinfo) {
1225                 $root = $this-> classes-> getClassByPackage($classinfo, $package);
1226                 if ($root && $root-> isInterface()) {
1227                     continue;
1228                 }
1229                 $thistree .=
1230                 $this-> getRootTree(
1231                 $this-> getSortedClassTreeFromClass(
1232                     $classinfo,
1233                     $package,
1234                     ''),
1235                     $package,
1236                     true);
1237             if (!$thistree) {
1238                 continue;
1239             }
1240             $trees[] = array(
1241                 'class' => $parent,
1242                 'class_tree' => "<ul>\n"
1243             . $thistree .
1244             );
1245         }
1246     return $trees;
1247 }
1248 /**
1249 * returns a template-enabled array of interface inheritance trees
1250 *
1251 * @param string $package package to generate a class tree for
1252 * @see $roots, HTMLConverter::getRootTree()
1253 */
1254 function generateFormattedInterfaceTrees($package)
1255 {
    if (!isset($this-> roots['normal'][$package]) &&
        !isset($this-> roots['special'][$package])) {
1259     return array();
1260 }
1261 $trees = array();
1262 if (isset($this-> roots['normal'][$package])) {
1263     $roots = $this-> roots['normal'][$package];
1264     for($i=0;$i< count($roots);$i++)
1265     {
1266         $root = $this-> classes-> getClassByPackage($roots[$i], $package);
1267         if ($root && !$root-> isInterface()) {
1268             continue;
1269         }
1270         $trees[] = array('class' => $roots[$i], 'class_tree' =>
1271             . $this-> getRootTree($this-
1272             > getSortedClassTreeFromClass($roots[$i],$package,''),$package). "</ul>\n"
1273         }
1274     if (isset($this-> roots['special'][$package])) {
1275         $roots = $this-> roots['special'][$package];
1276         foreach ($roots as $parent => $classes) {
1277             $thistree = '';
1278             foreach ($classes as $classinfo) {
1279                 $root = $this-> classes-> getClassByPackage($classinfo, $package);
1280                 if ($root && !$root-> isInterface()) {
1281                     continue;
1282                 }
1283             $thistree =

```

```

1283             $this-> _getRootTree(
1284                 $this-> _getSortedClassTreeFromClass(
1285                     $classinfo,
1286                     $package,
1287                     ''),
1288                     $package,
1289                     true);
1290         }
1291         if (!$histree) {
1292             continue;
1293         }
1294         $trees[] = array(
1295             'class' => $parent,
1296             'class_tree' => "<ul>\n" . $histree .
1297             "</ul>\n"
1298         );
1299     }
1300     return $trees;
1301 }
1302 /**
1303 * return formatted class tree for the Class Trees page
1304 *
1305 * @param array $tree output from {@link getSortedClassTreeFromClass()}
1306 * @param string $package package
1307 * @param boolean $nounknownparent if true, an object's parent will not be checked
1308 * @see Classes::$definitechild, generateFormattedClassTrees()
1309 * @return string
1310 */
1311 function getRootTree($tree, $package, $nparent = false)
1312 {
1313     if (!$tree) return '';
1314     $my_tree = '';
1315     $cur = '#root';
1316     $lastcur = array(false);
1317     $kids = array();
1318     $dopar = false;
1319     if (!$nparent && $tree[$cur]['parent'])
1320     {
1321         $dopar = true;
1322         if (!is_object($tree[$cur]['parent']))
1323         {
1324             // debug("parent ".$tree[$cur]['parent']." not found");
1325             $my_tree .= '<listitem>' . $tree[$cur]['parent'] . '<itemizedlist>';
1326         }
1327         else
1328         {
1329             //
1330             // debug("parent ".$this->returnSee($tree[$cur]['parent']),
1331             // false, false)." in other package");
1332             $my_tree .= '<listitem>' . $this-> returnSee($tree[$cur]['parent'],
1333             false, false);
1334             if ($tree[$cur]['parent']-> package != $package) $my_tree .= '
1335             <emphasis>(Different package)</emphasis><itemizedlist>';
1336         }
1337     }
1338     do
1339     {
1340         //
1341         fancy_debug($cur,$lastcur,$kids);
1342         if (count($tree[$cur]['children']))
1343         {
1344             //
1345             debug("$cur has children");
1346             if (!isset($kids[$cur]))
1347             {
1348                 debug("set $cur kids");
1349                 $kids[$cur] = 1;
1350                 $my_tree .= '<listitem>' . $this-> returnSee($tree[$cur]['link'],
1351                 false, false);
1352                 $my_tree .= '<itemizedlist>' . "\n";
1353                 array_push($lastcur,$cur);
1354                 list($cur) = each($tree[$cur]['children']);
1355                 var_dump('listed',$cur);
1356                 if ($cur)
1357                 {
1358                     $cur = $cur['package'] . '#' . $cur['class'];
1359                     debug("set cur to child $cur");
1360                     $my_tree .= '<li>' . $this->returnSee($tree[$cur]['link']), false,
1361                     false);
1362                     continue;
1363                 }
1364             }
1365         }
1366     }
1367 }

```

```

1357     } else
1358     {
1359         //           debug("end of children for $cur");
1360         $cur = array_pop($lastcur);
1361         $cur = array_pop($lastcur);
1362         $my_tree .= '</itemizedlist></listitem>' . "\n";
1363         if ($dopar && (           $cur == '#root' || !$cur)) $my_tree .=
1364             '</itemizedlist></listitem>' ;
1365     }
1366     {
1367         //           debug("$cur has no children");
1368         $my_tree .= '<listitem>' . $this-> returnSee($tree[$cur]['link'], false,
1369             "</listitem>" );
1370         if ($dopar &&           $cur == '#root') $my_tree .=
1371             '</itemizedlist></listitem>' ;
1372         $cur = array_pop($lastcur);
1373     }
1374 }
1375 /**
1376 * does nothing
1377 */
1378 function generateElementIndex()
1379 {
1380 }
1381
1382 function setTemplateDir($dir)
1383 {
1384     Converter::setTemplateDir($dir);
1385     $this-> smarty_dir = $this-> templateDir;
1386 }
1387
1388 /**
1389 * Generate alphabetical index of all elements by package and subpackage
1390 *
1391 * @param string $package name of a package
1392 * @see $pkg_elements, walk(), generatePkgElementIndexes()
1393 */
1394 function generatePkgElementIndex($package)
1395 {
1396 }
1397
1398 /**
1399 *
1400 * @see generatePkgElementIndex()
1401 */
1402 function generatePkgElementIndexes()
1403 {
1404 }
1405
1406 /**
1407 * @param string name of class
1408 * @param string package name
1409 * @param string full path to look in (used in index generation)
1410 * @param boolean deprecated
1411 * @param boolean return just the URL, or enclose it in an html a tag
1412 * @return mixed false if not found, or an html a link to the class's documentation
1413 * @see parent::getClassLink()
1414 */
1415 function getClassLink($expr,$package, $file = false,$text = false, $local = true, $with_a =
true)
1416 {
1417     $a = Converter::getClassLink($expr,$package,$file);
1418     if (!$a) return false;
1419     return $this-> returnSee($a, $text, $local, $with_a);
1420 }
1421
1422 /**
1423 * @param string name of function
1424 * @param string package name
1425 * @param string full path to look in (used in index generation)
1426 * @param boolean deprecated
1427 * @param boolean return just the URL, or enclose it in an html a tag
1428 * @return mixed false if not found, or an html a link to the function's documentation
1429 * @see parent::getFunctionLink()
1430 */
1431 function getFunctionLink($expr,$package, $file = false,$text = false, $local = true)
1432 {

```

```

1433     $a = Converter::getFunctionLink($expr,$package,$file);
1434     if (!$a) return false;
1435     return $this->  returnSee($a, $text, $local);
1436 }
1437
1438 /**
1439 * @param string name of define
1440 * @param string package name
1441 * @param string full path to look in (used in index generation)
1442 * @param boolean deprecated
1443 * @param boolean return just the URL, or enclose it in an html a tag
1444 * @return mixed false if not found, or an html a link to the define's documentation
1445 * @see parent::getDefineLink()
1446 */
1447 function getDefineLink($expr,$package, $file = false,$text = false, $local = true)
1448 {
1449     $a = Converter::getDefineLink($expr,$package,$file);
1450     if (!$a) return false;
1451     return $this->  returnSee($a, $text, $local);
1452 }
1453
1454 /**
1455 * @param string name of global variable
1456 * @param string package name
1457 * @param string full path to look in (used in index generation)
1458 * @param boolean deprecated
1459 * @param boolean return just the URL, or enclose it in an html a tag
1460 * @return mixed false if not found, or an html a link to the global variable's
documentation
1461 * @see parent::getGlobalLink()
1462 */
1463 function getGlobalLink($expr,$package, $file = false,$text = false, $local = true)
1464 {
1465     $a = Converter::getGlobalLink($expr,$package,$file);
1466     if (!$a) return false;
1467     return $this->  returnSee($a, $text, $local);
1468 }
1469
1470 /**
1471 * @param string name of procedural page
1472 * @param string package name
1473 * @param string full path to look in (used in index generation)
1474 * @param boolean deprecated
1475 * @param boolean return just the URL, or enclose it in an html a tag
1476 * @return mixed false if not found, or an html a link to the procedural page's
documentation
1477 * @see parent::getPageLink()
1478 */
1479 function getPageLink($expr,$package, $path = false,$text = false, $local = true)
1480 {
1481     $a = Converter::getPageLink($expr,$package,$path);
1482     if (!$a) return false;
1483     return $this->  returnSee($a, $text, $local);
1484 }
1485
1486 /**
1487 * @param string name of method
1488 * @param string class containing method
1489 * @param string package name
1490 * @param string full path to look in (used in index generation)
1491 * @param boolean deprecated
1492 * @param boolean return just the URL, or enclose it in an html a tag
1493 * @return mixed false if not found, or an html a link to the method's documentation
1494 * @see parent::getMethodLink()
1495 */
1496 function getMethodLink($expr,$class,$package, $file = false,$text = false, $local = true)
1497 {
1498     $a = Converter::getMethodLink($expr,$class,$package,$file);
1499     if (!$a) return false;
1500     return $this->  returnSee($a, $text, $local);
1501 }
1502
1503 /**
1504 * @param string name of var
1505 * @param string class containing var
1506 * @param string package name
1507 * @param string full path to look in (used in index generation)
1508 * @param boolean deprecated
1509 * @param boolean return just the URL, or enclose it in an html a tag
1510 * @return mixed false if not found, or an html a link to the var's documentation

```

```

1511     * @see parent::getVarLink()
1512     */
1513     function getVarLink($expr,$class,$package, $file = false,$text = false, $local = true)
1514     {
1515         $a = Converter::getVarLink($expr,$class,$package,$file);
1516         if (!$a) return false;
1517         return $this-> returnSee($a, $text, $local);
1518     }
1519
1520 /**
1521 * does a nat case sort on the specified second level value of the array
1522 *
1523 * @param mixed $a
1524 * @param mixed $b
1525 * @return int
1526 */
1527 function rcNatCmp ($a, $b)
1528 {
1529     $aa = strtoupper($a[$this-> rcnatcmpkey]);
1530     $bb = strtoupper($b[$this-> rcnatcmpkey]);
1531
1532     return strnatcasecmp($aa, $bb);
1533 }
1534
1535 /**
1536 * does a nat case sort on the specified second level value of the array.
1537 * this one puts constructors first
1538 *
1539 * @param mixed $a
1540 * @param mixed $b
1541 * @return int
1542 */
1543 function rcNatCmp1 ($a, $b)
1544 {
1545     $aa = strtoupper($a[$this-> rcnatcmpkey]);
1546     $bb = strtoupper($b[$this-> rcnatcmpkey]);
1547
1548     if (strpos($aa,'CONSTRUCTOR') === 0)
1549     {
1550         return -1;
1551     }
1552     if (strpos($bb,'CONSTRUCTOR') === 0)
1553     {
1554         return 1;
1555     }
1556     if (strpos($aa,strtoupper($this-> class)) === 0)
1557     {
1558         return -1;
1559     }
1560     if (strpos($bb,strtoupper($this-> class)) === 0)
1561     {
1562         return -1;
1563     }
1564     return strnatcasecmp($aa, $bb);
1565 }
1566
1567 function wordwrap($string)
1568 {
1569     return wordwrap($string);
1570 }
1571
1572 /**
1573 * Generate the constants.xml, packagename.xml, and globals.xml files
1574 */
1575 function Output()
1576 {
1577     $this-> flushPackageXml(false);
1578     $templ = & $this-> newSmarty();
1579     $categories = array();
1580     $packages = array_flip($this-> all_packages);
1581     foreach($this-> packageCategories as $package => $category)
1582     {
1583         $categories[$category]['package.' . $category . '.str_replace('_', '-' ,strtolower($package))] = 1;
1584         if (isset($packages[$package])) unset($packages[$package]);
1585     }
1586     $category = $GLOBALS['phpDocumentor_DefaultCategoryName'];
1587     foreach($packages as $package)
1588     {
1589         $categories[$category]['package.' . $category . '.str_replace('_', '-' ,

```

```

',strtolower($package))] = 1;
1590     }
1591     foreach($categories as $category =>      $ids)
1592     {
1593         $templ-> assign('id','package.' . $category);
1594         $templ-> assign('ids',array());
1595         $templ-> assign('category',$category);
1596         $this-> setTargetDir($this-> base_dir);
1597         if (file_exists($this-> base_dir . PATH_DELIMITER . strtolower($category) .
1598 '.xml'))
1599         {
1600             $contents = @file($this-> base_dir . PATH_DELIMITER . strtolower($category) .
1601 '.xml');
1602             if (is_array($contents))
1603             {
1604                 $found = false;
1605                 foreach($contents as $i =>      $line)
1606                 {
1607                     $line = trim($line);
1608                     if (strlen($line) &&          $line[0] == '&'      )
1609                     {
1610                         $found = $i;
1611                         if (in_array(str_replace(array ('&', ';'), array ('', ''),
1612 trim($line)), array_keys($ids)))
1613                         {
1614                             unset($ids[str_replace(array('&', ';'), array('', ''), trim($line))]);
1615                         }
1616                     }
1617                 }
1618             }
1619             $newids = array();
1620             foreach($ids as $id =>      $unll)
1621             {
1622                 $newids[] = ' & ' . $id . ";\n";
1623             }
1624             $newcontents = array_merge(array_slice($contents, 0, $i), $newids);
1625             $newcontents = array_merge($newcontents, array_slice($contents, $i));
1626             $categorycontents = implode($newcontents, '');
1627         } else
1628         {
1629             foreach($ids as $id =>      $unll)
1630             {
1631                 if (!in_array($id, $templ-> _tpl_vars['ids']))
1632                 {
1633                     $templ-> append('ids',$id);
1634                 }
1635             }
1636             $categorycontents = '<!-- $' . "Revision$ -->\n" . $templ-
1637 > fetch('category.tpl');
1638         }
1639         $this-> writefile(strtolower($category) . '.xml',
1640             $categorycontents);
1641         phpDocumentor_out("\n");
1642         flush();
1643     }
1644     $my = & $this-> newSmarty();
1645     if ($this-> _peardoc2_constants)
1646     {
1647         foreach($this-> _peardoc2_constants as $category =>      $r)
1648         {
1649             foreach($r as $package =>      $s)
1650             {
1651                 $my-> assign('id','package.' . strtolower($category) . '.str_replace('_', '-' ,
1652 strtolower($package))).'.constants');
1653                 $my-> assign('package',$package);
1654                 $defines = array();
1655                 foreach($s as $file =>      $t)
1656                 {
1657                     $arr = array();
1658                     $arr['name'] = $file;
1659                     $arr['page'] = strtolower($t['page']);
1660                     $arr['defines'] = $t['defines'];
1661                     $defines[] = $arr;
1662                 }
1663                 $my-> assign('defines',$defines);

```

```

1663     $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $category
1664         . PATH_DELIMITER . strtolower(str_replace('_', '-',strtolower($package
1665 ))));
1665     $this-> writefile('constants.xml',
1666         '<!-- $' . "Revision$ -->\n" . $my-
1667     );
1668     $my-> clear_all_assign();
1669 }
1670 $this-> _peardoc2_constants = false;
1671 if ($this-> _peardoc2_globals)
1672 {
1673     foreach($this-> _peardoc2_globals as $category => $r)
1674     {
1675         foreach($r as $package => $s)
1676         {
1677             $my-> assign('id','package.'.strtolower($category).'.'.str_replace('_', '-',
1678 ',$tolower($package ))).'.'globals');
1679             $my-> assign('package',$package);
1680             $defines = array();
1681             foreach($s as $file => $t)
1682             {
1683                 $arr = array();
1684                 $arr['name'] = $file;
1685                 $arr['page'] = strtolower($t['page']);
1686                 $arr['globals'] = $t['globals'];
1687                 $defines[] = $arr;
1688             }
1689             $my-> assign('globals',$defines);
1690             $this-> setTargetDir($this-> base_dir . PATH_DELIMITER . $category
1691                 . PATH_DELIMITER . strtolower(str_replace('_', '-',strtolower($package
1692 ))));
1693             $this-> writefile('globals.xml',
1694                 '<!-- $' . "Revision$ -->\n" . $my-
1695             );
1696         }
1697     $this-> _peardoc2_globals = false;
1698 }
1699 }
1700 }
1701 ?>
```

## Package phpDocumentor

# File Source for file\_dialog.php

Documentation for this file is available at [file\\_dialog.php](#)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <?php
3  /**
4   * phpDocumentor :: docBuilder Web Interface
5   *
6   * Advanced Web Interface to phpDocumentor
7   *
8   * PHP versions 4 and 5
9   *
10  * Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver
11  *
12  * LICENSE:
13  *
14  * This library is free software; you can redistribute it
15  * and/or modify it under the terms of the GNU Lesser General
16  * Public License as published by the Free Software Foundation;
17  * either version 2.1 of the License, or (at your option) any
18  * later version.
19  *
20  * This library is distributed in the hope that it will be useful,
21  * but WITHOUT ANY WARRANTY; without even the implied warranty of
22  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23  * Lesser General Public License for more details.
24  *
25  * You should have received a copy of the GNU Lesser General Public
26  * License along with this library; if not, write to the Free Software
27  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28  *
29  * @package    phpDocumentor
30  * @author     Andrew Eddie
31  * @author     Greg Beaver <cellog@php.net>
32  * @copyright  2003-2006 Andrew Eddie, Greg Beaver
33  * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version    CVS: $Id: file_dialog.php 231860 2007-03-14 12:48:37Z ashnazg $
35  * @filesource
36  * @see       phpdoc.php
37  */
38  if (!function_exists( 'version_compare' )) {
39      print "phpDocumentor requires PHP version 4.1.0 or greater to function";
40      exit;
41  }
42
43  if ('@WEB-DIR@' != '@'.'WEB-DIR@')
44  {
45      /**
46      * common file information
47      */
48      require_once 'PhpDocumentor/phpDocumentor/common.inc.php';
49      require_once 'PhpDocumentor/HTML_TreeMenu-1.1.2/TreeMenu.php';
50      require_once( '@WEB-DIR@' . PATH_DELIMITER .
'PhpDocumentor/docbuilder/includes/utilities.php' );
51
52      define(PHPDOC_WEBROOT_DIR, 'PhpDocumentor');
53
54      // set up include path so we can find all files, no matter what
55      $GLOBALS['_phpDocumentor_install_dir'] = PHPDOC_WEBROOT_DIR;
56
57      // find the .ini directory by parsing phpDocumentor.ini and extracting
58      // _phpDocumentor_options[userdir]
59      $ini = phpDocumentor_parse_ini_file('@DATA-DIR@' . PATH_DELIMITER .
'PhpDocumentor/phpDocumentor.ini', true);
60      if (isset($ini['_phpDocumentor_options'][['userdir']])) {
61          $configdir = $ini['_phpDocumentor_options'][['userdir']];
62      } else {
63          $configdir = '@DATA-DIR@' . PATH_DELIMITER . 'PhpDocumentor/user';
64      }
64 } else {
```

```

65
66     define(PHPDOC_WEBROOT_DIR, dirname(dirname(__FILE__)));
67     /**
68      * common file information
69     */
70     include_once(PHPDOC_WEBROOT_DIR . "/phpDocumentor/common.inc.php" );
71     include_once(PHPDOC_WEBROOT_DIR . "/HTML_TreeMenu-1.1.2/TreeMenu.php" );
72     include_once(PHPDOC_WEBROOT_DIR . "/docbuilder/includes/utilities.php" );
73
74     // set up include path so we can find all files, no matter what
75     $GLOBALS['_phpDocumentor_install_dir'] = dirname(dirname( realpath( __FILE__ ) ));
76     // add my directory to the include path, and make it first, should fix any errors
77     if (substr(PHP_OS, 0, 3) == 'WIN') {
78
78     ini_set('include_path',$GLOBALS['_phpDocumentor_install_dir'].':'.ini_get('include_path'));
79     } else {
80
80     ini_set('include_path',$GLOBALS['_phpDocumentor_install_dir'].':'.ini_get('include_path'));
81     }
82
83     // find the .ini directory by parsing phpDocumentor.ini and extracting
83     _phpDocumentor_options[userdir]
84     $ini = _phpDocumentor_parse_ini_file($phpDocumentor_install_dir . PATH_DELIMITER .
84     'phpDocumentor.ini', true);
85     if (isset($ini['_phpDocumentor_options'][['userdir']])) {
86         $configdir = $ini['_phpDocumentor_options'][['userdir']];
87     } else {
88         $configdir = $phpDocumentor_install_dir . '/user';
89     }
90
91
92     // allow the user to change this at runtime
93     if (!empty( $_REQUEST['altuserdir'] )) {
94         $configdir = $_REQUEST['altuserdir'];
95     }
96     ?>
97     <html>
98     <head>
99         <title>
100             File browser
101         </title>
102         <style type="text/css">
103             body, td, th, select, input {
104                 font-family: verdana,sans-serif;
105                 font-size: 9pt;
106             }
107             .text {
108                 font-family: verdana,sans-serif;
109                 font-size: 9pt;
110                 border: solid 1px #000000;
111             }
112             .button {
113                 border: solid 1px #000000;
114             }
115             .small {
116                 font-size: 7pt;
117             }
118         </style>
119
120         <script src="../HTML_TreeMenu-1.1.2/TreeMenu.js"
language="JavaScript" type="text/javascript"></script>
121
122     <?php
123         set_time_limit(0); // six minute timeout
124         ini_set("memory_limit", "256M");
125
126         /**
127          * Directory Node
128          * @package HTML_TreeMenu
129          */
130         class DirNode extends HTML_TreeNode
131     {
132
133             /**
134              * full path to this node
135              * @var string
136             */
136             var $path;
137
138             function DirNode($text = false, $link = false, $icon = false, $path, $events = array())
139             {

```

```

140         $this-> path = $path;
141         $options = array();
142         if ($text) $options['text'] = $text;
143         if ($link) $options['link'] = $link;
144         if ($icon) $options['icon'] = $icon;
145         HTML_TreeNode::HTML_TreeNode($options,$events);
146     }
147 }
148
149
150 $menu = new HTML_TreeMenu();
151 $filename = '';
152 if (isset($_GET) && isset($_GET['fileName'])) {
153     $filename = $_GET['fileName'];
154 }
155 $filename = realpath($filename);
156 $pd = (substr(PHP_OS, 0, 3) == 'WIN') ? '\\\\' : '/';
157 $test = ($pd == '/') ? '/' : 'C:\\\\';
158 if (empty($filename) || ($filename == $test)) {
159     $filename = ($pd == '/') ? '/' : 'C:\\\\';
160     $node = false;
161     getDir($filename,$node);
162 } else {
163     flush();
164 //     if ($pd != '/') $pd = $pd.$pd;
165     $anode = false;
166     switchDirTree($filename,$anode);
167 //     recurseDir($filename,$anode);
168     $node = new HTML_TreeNode(array('text' => "Click to view
" .addslashes($filename), 'link' => "", 'icon' => 'branchtop.gif'));
169     $node-> addItem($anode);
170 };
171 $menu-> addItem($node);
172 if ('@WEB-DIR@' != '@'. 'WEB-DIR@')
173 {
174     $DHTMLmenu = & new HTML_TreeMenu_DHTML($menu,
175                                     array('images' => '../HTML_TreeMenu-1.1.2/images'));
176 } else {
177     $DHTMLmenu = & new HTML_TreeMenu_DHTML($menu,
178                                     array('images' =>
str_replace('/docbuilder/file_dialog.php', '', $SERVER['PHP_SELF']) .
179                                         '/HTML_TreeMenu-1.1.2/images'));
180 }
181 ?>
182 <script type="text/javascript" language="Javascript">
183 /**
184     Creates some global variables
185 */
186 function initialize() {
187 //
188 //The "platform independent" newLine
189 //
190 //Taken from http://developer.netscape.com/docs/manuals/communicator/jeref/brow1.htm#1010426
191     if (navigator.appVersion.lastIndexOf( 'Win' ) != -1) {
192         $pathdelim="\\\\";
193         $newLine="\r\n";
194     } else {
195         $newLine="\n";
196         $pathdelim="/";
197     }
198     /* for($a=0;$a<document.dataForm.elements.length;$a++) {
199         alert("The name is "+document.dataForm.elements[$a].name+" "+$a);
200     }
201     */
202 }
203 /**
204 Sets the contents of the help box, and submits the form
205 */
206 function setHelp( $str ) {
207     document.helpForm.fileName.value = $str;
208     document.helpForm.submit();
209 }
210 /**
211 Sets the contents of the help box only
212 */
213 function setHelpVal( $str ) {
214     document.helpForm.fileName.value = $str;
215 }
216 /**
217 **Takes a given string and leaves it ready to add a new string
218     That is, puts the comma and the new line if needed
219 */

```

```

218     function prepareString($myString) {
219         //First verify that a comma is not at the end
220         if ($myString.lastIndexOf(",") >= $myString.length-2) {
221             //We have a comma at the end
222             return $myString;
223         }
224         if ($myString.length > 0) {
225             $myString+=", "+$newLine;
226         }
227         return $myString;
228     }
229
230
231     function myReplace($string,$text,$by) {
232         // Replaces text with by in string
233         var $strLength = $string.length, $txtLength = $text.length;
234         if (($strLength == 0) || ($txtLength == 0)) return $string;
235
236         var $i = $string.indexOf($text);
237         if (((!$i) && ($text != $string.substring(0,$txtLength))) return $string;
238         if ($i == -1) return $string;
239
240         var $newstr = $string.substring(0,$i) + $by;
241
242         if ($i+$txtLength < $strLength)
243             $newstr += myReplace($string.substring($i+$txtLength,$strLength),$text,$by);
244
245         return $newstr;
246     }
247 </script>
248 </head>
249
250 <body bgcolor="#ffffff" onload="javascript:initialize()">
251 <strong>Directory Browser</strong>
252
253 <table cellpadding="1" cellspacing="1" border="0"
width="100%">
254
255     <form name="helpForm" action=<?php print
256 ?>" method="get" enctype="multipart/form-data">
257         <tr>
258             <td colspan="2" width="100%">
259                 Use this to find directories and files which can be used below:
260             </td>
261         </tr>
262         <tr>
263             <td align="right">
264                 <a href="javascript:document.helpForm.submit();" title="browse
tree">
265                     <?php
266                         echo showImage( 'images/rc-gui-install-24.png', '24', '24' );
267                     ?>
268                 </a>
269             </td>
270             <td>
271                 <input size="60" type="text" name="fileName"
value=<?php print      $filename;?>" class="text" />
272             </td>
273         </tr>
274         <tr>
275             <td>
276                 <input type="submit" name="helpdata" value="close"
class="button" onclick="window.close();" />
277             </td>
278             <td align="right">
279                 <input type="submit" name="helpdata" value="accept"
class="button"
onclick="opener.setFile(document.helpForm.fileName.value);window.close();" />
280             </td>
281         </tr>
282         <tr>
283             <td colspan="2">
284                 <div id='menuLayer'></div>
285                 <?php      $DHTMLmenu->    printMenu(); ?>
286             </td>
287         </tr>
288     </form>
289
290 </table>

```

```
291
292  </body>
293  </html>
```

## Package phpDocumentor

# File Source for builder.php

Documentation for this file is available at [builder.php](#)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <?php
3  /**
4   * phpDocumentor :: docBuilder Web Interface
5   *
6   * Advanced Web Interface to phpDocumentor
7   *
8   * PHP versions 4 and 5
9   *
10  * Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver
11  *
12  * LICENSE:
13  *
14  * This library is free software; you can redistribute it
15  * and/or modify it under the terms of the GNU Lesser General
16  * Public License as published by the Free Software Foundation;
17  * either version 2.1 of the License, or (at your option) any
18  * later version.
19  *
20  * This library is distributed in the hope that it will be useful,
21  * but WITHOUT ANY WARRANTY; without even the implied warranty of
22  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23  * Lesser General Public License for more details.
24  *
25  * You should have received a copy of the GNU Lesser General Public
26  * License along with this library; if not, write to the Free Software
27  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28  *
29  * @package    phpDocumentor
30  * @author     Andrew Eddie
31  * @author     Greg Beaver <cellog@php.net>
32  * @copyright  2003-2006 Andrew Eddie, Greg Beaver
33  * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version    CVS: $Id: builder.php 212211 2006-04-30 22:18:14Z cellog $
35  * @filesource
36  * @see       phpdoc.php
37  */
38
39  if (!function_exists('version_compare'))
40  {
41      print "phpDocumentor requires PHP version 4.1.0 or greater to function";
42      exit;
43  }
44
45
46  if ('@DATA-DIR@' != '@'. 'DATA-DIR@') {
47      // set up include path so we can find all files, no matter what
48      $root_dir = 'PhpDocumentor';
49      /**
50      * common file information
51      */
52      include_once("      $root_dir/phpDocumentor/common.inc.php" );
53      $GLOBALS['_phpDocumentor_install_dir'] = 'PhpDocumentor';
54      // find the .ini directory by parsing phpDocumentor.ini and extracting
55      $ini = phpdDocumentor_parse_ini_file('@DATA-DIR@/PhpDocumentor/phpDocumentor.ini', true);
56      if (isset($ini['_phpDocumentor_options']['userdir'])) {
57          $configdir = $ini['_phpDocumentor_options']['userdir'];
58      } else {
59          $configdir = '@DATA-DIR@/user';
60      }
61  } else {
62      // set up include path so we can find all files, no matter what
63      $GLOBALS['_phpDocumentor_install_dir'] = dirname(dirname(realpath(__FILE__)));
64      $root_dir = dirname(dirname(__FILE__));
65      /**
66
```

```

67     * common file information
68     */
69     include_once("$root_dir/phpDocumentor/common.inc.php");
70     // add my directory to the include path, and make it first, should fix any errors
71     if (substr(PHP_OS, 0, 3) == 'WIN')
72     {
73
74     ini_set('include_path',$_GLOBALS['_phpDocumentor_install_dir'].":"._ini_get('include_path'));
75     } else {
76
77     ini_set('include_path',$_GLOBALS['_phpDocumentor_install_dir'].":"._ini_get('include_path'));
78     }
79     // find the .ini directory by parsing phpDocumentor.ini and extracting
80     // _phpDocumentor_options[userdir]
81     $ini = phpDocumentor_parse_ini_file($_phpDocumentor_install_dir . PATH_DELIMITER .
82     'phpDocumentor.ini', true);
83     if (isset($ini['_phpDocumentor_options']['userdir']))
84     {
85         $configdir = $ini['_phpDocumentor_options']['userdir'];
86     } else {
87         $configdir = $_phpDocumentor_install_dir . '/user';
88     }
89
90     // allow the user to change this at runtime
91     if (!empty($_REQUEST['altuserdir'])) $configdir = $_REQUEST['altuserdir'];
92     ?>
93     <html>
94     <head>
95         <title>
96             output: docbuilder - phpDocumentor v<?php print      PHPDOCUMENTOR_VER; ?> doc
97             generation information
98         </title>
99         <style type="text/css">
100             body, td, th {
101                 font-family: verdana,sans-serif;
102                 font-size: 8pt;
103             }
104         </style>
105     </head>
106     <body bgcolor="#e0e0e0" text="#000000" topmargin="0"
107     leftmargin="0" marginheight="0" marginwidth="0">
108     <?php
109     // Find out if we are submitting and if we are, send it
110     // This code originally by Joshua Eichorn on phpdoc.php
111     //
112     if (isset($_GET['dataform']) && empty(          $_REQUEST['altuserdir'])) {
113         foreach ($_GET as $k=> $v) {
114             if (strpos( $k, 'setting_' ) === 0) {
115                 $_GET['setting'][substr( $k, 8 )] = $v;
116             }
117
118             echo "<strong>Parsing Files ...</strong>" ;
119             flush();
120             echo "<pre>\n" ;
121             /** phpdoc.inc */
122             include("$root_dir/phpDocumentor/phpdoc.inc");
123             echo "</pre>\n" ;
124             echo "<h1>Operation Completed!!</h1>" ;
125         } else {
126             echo "whoops!" ;
127         }
128     ?>

```

# File Source for config.php

Documentation for this file is available at [config.php](#)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <?php
3  /**
4   * phpDocumentor :: docBuilder Web Interface
5   *
6   * Advanced Web Interface to phpDocumentor
7   *
8   * PHP versions 4 and 5
9   *
10  * Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver
11  *
12  * LICENSE:
13  *
14  * This library is free software; you can redistribute it
15  * and/or modify it under the terms of the GNU Lesser General
16  * Public License as published by the Free Software Foundation;
17  * either version 2.1 of the License, or (at your option) any
18  * later version.
19  *
20  * This library is distributed in the hope that it will be useful,
21  * but WITHOUT ANY WARRANTY; without even the implied warranty of
22  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23  * Lesser General Public License for more details.
24  *
25  * You should have received a copy of the GNU Lesser General Public
26  * License along with this library; if not, write to the Free Software
27  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28  *
29  * @package    phpDocumentor
30  * @author     Andrew Eddie
31  * @author     Greg Beaver <cellog@php.net>
32  * @copyright  2003-2006 Andrew Eddie, Greg Beaver
33  * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version    CVS: $Id: config.php 234145 2007-04-19 20:20:57Z ashnazg $
35  * @filesource
36  * @see       phpdoc.php
37  */
38
39 if (!function_exists( 'version_compare' )) {
40     print "phpDocumentor requires PHP version 4.1.0 or greater to function";
41     exit;
42 }
43
44 if ('@DATA-DIR@' != '@'. 'DATA-DIR@')
45 {
46     $root_dir = 'PhpDocumentor';
47     $path = '@WEB-DIR@/PhpDocumentor/docbuilder/images/';
48
49     /**
50      * common file information
51     */
52     include_once("PhpDocumentor/phpDocumentor/common.inc.php");
53     include_once("@WEB-DIR@/PhpDocumentor/docbuilder/includes/utilities.php");
54
55     // find the .ini directory by parsing phpDocumentor.ini and extracting
56     // phpDocumentor_options[userdir]
57     $ini = phpDocumentor_parse_ini_file('@DATA-DIR@/PhpDocumentor/phpDocumentor.ini', true);
58     if (isset($ini['_phpDocumentor_options']['userdir'])) {
59         $configdir = $ini['_phpDocumentor_options']['userdir'];
60     } else {
61         $configdir = '@DATA-DIR@/PhpDocumentor/user';
62     } else {
63         $root_dir = dirname(dirname(__FILE__));
64         $path = 'images/';
65
66     // set up include path so we can find all files, no matter what
```

```

67     $GLOBALS['_phpDocumentor_install_dir'] = dirname(dirname( realpath( __FILE__ ) ));
68     // add my directory to the include path, and make it first, should fix any errors
69     if (substr(PHP_OS, 0, 3) == 'WIN') {
70
71         ini_set('include_path', $GLOBALS['_phpDocumentor_install_dir'].';'.ini_get('include_path'));
72     } else {
73
74
75         /**
76          * common file information
77         */
78         include_once("      $root_dir/phpDocumentor/common.inc.php"      );
79         include_once("      $root_dir/docbuilder/includes/utilities.php"      );
80
81         // find the .ini directory by parsing phpDocumentor.ini and extracting
82         // phpDocumentor_options[userdir]
82         $ini = phpDocumentor_parse_ini_file($phpDocumentor_install_dir . PATH_DELIMITER .
82         'phpDocumentor.ini', true);
83         if (isset($ini['_phpDocumentor_options']['userdir'])) {
84             $configdir = $ini['_phpDocumentor_options']['userdir'];
85         } else {
86             $configdir = $phpDocumentor_install_dir . '/user';
87         }
88     }
89
90     // allow the user to change this at runtime
91     if (!empty( $_REQUEST['altuserdir'] )) {
92         $configdir = $_REQUEST['altuserdir'];
93     }
94
95     // assign the available converters
96     $converters = array(
97         "HTML:frames:default"                      => 'HTML:frames:default',
98         "HTML:frames:earthli"                      => 'HTML:frames:earthli',
99         "HTML:frames:10133t"                       => 'HTML:frames:10133t',
100        "HTML:frames:phpdoc.de"                    => 'HTML:frames:phpdoc.de',
101        "HTML:frames:phptmllib"                   => 'HTML:frames:phptmllib',
102        "HTML:frames:phpedit"                     => 'HTML:frames:phpedit',
103        "HTML:frames:DOM/default"                 => 'HTML:frames:DOM/default',
104        "HTML:frames:DOM/earthli"                 => 'HTML:frames:DOM/earthli',
105        "HTML:frames:DOM/10133t"                  => 'HTML:frames:DOM/10133t',
106        "HTML:frames:DOM/phpdoc.de"               => 'HTML:frames:DOM/phpdoc.de',
107        "HTML:frames:DOM/phptmllib"              => 'HTML:frames:DOM/phptmllib',
108        "HTML:Smarty:default"                    => 'HTML:Smarty:default',
109        "HTML:Smarty:HandS"                     => 'HTML:Smarty:HandS',
110        "HTML:Smarty:PHP"                       => 'HTML:Smarty:PHP',
111        "PDF:default:default"                   => 'PDF:default:default',
112        "CHM:default:default"                   => 'CHM:default:default',
113        "XML:DocBook/peardoc2:default"          => 'XML:DocBook/peardoc2:default'
114    );
115
116    // compile a list of available screen shots
117    $convScreenShots = array();
118
119    if ($dir = opendir($path)) {
120        while (($file = readdir($dir)) != false) {
121            if ($file != '.' && $file != '..') {
122                if (!is_dir($path . $file)) {
123                    if (strpos($file, 'ss_') === 0) {
124                        $key = substr($file, 3);
125                        $key = str_replace('_', ':', $key);
126                        $key = str_replace('/', ':', $key);
127                        $key = str_replace('.png', '', $key);
128                        $convScreenShots[$key] = $file;
129                    }
130                }
131            }
132        }
133    }
134
135    ?>
136    <html>
137    <head>
138        <title>
139            Form to submit to phpDocumentor v<?php print PHPDOCUMENTOR_VER; ?>
140        </title>
141        <style type="text/css">
142            body, td, th {

```

```

143         font-family: verdana,sans-serif;
144         font-size: 9pt;
145     }
146     .text {
147         font-family: verdana,sans-serif;
148         font-size: 9pt;
149         border: solid 1px #000000;
150     }
151     .small {
152         font-size: 7pt;
153     }
154 
```

`</style>`

```

155
156     <script src="includes/tabpane.js" language="JavaScript"
157     type="text/javascript"></script>
158     <link id="webfx-tab-style-sheet" type="text/css"
159     rel="stylesheet" href="includes/tab.webfx.css" />
160
161     <script type="text/javascript" language="Javascript">
162     /**
163      Creates some global variables
164     */
165     function initialize() {
166     //
167     //The "platform independent" newLine
168     //
169     //Taken from http://developer.netscape.com/docs/manuals/communicator/jsref/brow1.htm#1010426
170     if (navigator.appVersion.lastIndexOf('Win') != -1) {
171         $pathdelim="\\" ;
172         $newLine="\r\n";
173     } else {
174         $newLine="\n";
175         $pathdelim="/";
176     }
177     /**
178      Adds the contents of the help box as a directory
179     */
180     function addDirectory($object) {
181         //$.a = document.helpForm.fileName.value;
182         $.a = parent.ActionFrame.document.actionFrm.fileName.value;
183         $.a = myReplace( $.a, '\\\\', '\\');
184         if ($.a[$.a.length - 1] == $pathdelim) {
185             $.a = $.a.substring(0, $.a.length - 1);
186         }
187         if ($.a.lastIndexOf('.') > 0) {
188             $.a = $.a.substring(0,$.a.lastIndexOf($pathdelim));
189         }
190         $object.value = prepareString($object.value)+$.a;
191     /**
192      Adds the contents of the converter box to the converters list
193     */
194     function addConverter($object) {
195         $object.value = prepareString($object.value)+document.dataForm.ConverterSetting.value;
196     /**
197      Replaces the converters list with the contents of the converter box
198     */
199     function replaceConverter($object) {
200         $object.value = document.dataForm.ConverterSetting.value;
201     /**
202      Adds the contents of the help box as a file to the given control
203     */
204     function addFile($object) {
205         //$.a = document.helpForm.fileName.value;
206         $.a = parent.ActionFrame.document.actionFrm.fileName.value;
207         $.a = myReplace($.a,'\\\\','\\');
208         $object.value = prepareString($object.value)+$.a;
209     /**
210      Takes a given string and leaves it ready to add a new string
211      That is, puts the comma and the new line if needed
212     */
213     function prepareString($myString) {
214         //First verify that a comma is not at the end
215         if ($myString.lastIndexOf(",") >= $myString.length-2) {
216             //We have a comma at the end
217             return $myString;
218         }
219         if ($myString.length > 0) {
220             $myString+=", "+$newLine;
221         }
222     }

```

```

221     return $myString;
222 }
223 /**Do the validation needed before sending the form and return a truth value indicating if the
form can be sent
224 */
225 function validate() {
226     //Take out all newLines and change them by nothing
227     //This could be done by using javascript function's replacebut that was implemented only
until Navigator 4.0 and so it is better to use more backward compatible functions like substr
228     document.dataForm.elements[0].value= stripNewLines(document.dataForm.elements[0].value);
229     document.dataForm.elements[1].value= stripNewLines(document.dataForm.elements[1].value);
230     document.dataForm.elements[2].value= stripNewLines(document.dataForm.elements[2].value);
231     document.dataForm.elements[3].value= stripNewLines(document.dataForm.elements[3].value);
232     document.dataForm.elements[4].value= stripNewLines(document.dataForm.elements[4].value);
233     document.dataForm.elements[5].value= stripNewLines(document.dataForm.elements[5].value);
234     document.dataForm.elements[6].value= stripNewLines(document.dataForm.elements[6].value);
235     document.dataForm.elements[7].value= stripNewLines(document.dataForm.elements[7].value);
236     //By returning true we are allowing the form to be submitted
237     return true;
238 }
239 /**Takes a string and removes all the occurrences of new lines
240 Could have been implemented a lot easier with replace but it's not very backwards compatible
241 */
242 function stripNewLines( $myString ) {
243     return myReplace($myString,$newLine,'');
244 }
245
246 function myReplace($string,$text,$by) {
247     // Replaces text with by in string
248     var $strLength = $string.length, $txtLength = $text.length;
249     if (($strLength == 0) || ($txtLength == 0)) {
250         return $string;
251     }
252     var $i = $string.indexOf($text);
253     if (!($i) && ($text != $string.substring(0,$txtLength))) {
254         return $string;
255     }
256     if ($i == -1) {
257         return $string;
258     }
259     var $newstr = $string.substring(0,$i) + $by;
260     if ($i+$txtLength < $strLength) {
261         $newstr += myReplace($string.substring($i+$txtLength,$strLength),$text,$by);
262     }
263     return $newstr;
264 }
265
266 var screenShots = new Array();
267 <?php
268     $temp = array();
269     foreach ($converters as $k=> $v) {
270         if (array_key_exists($k, $convScreenShots )) {
271             echo "    \n$convScreenShots[$k] = '$v'      ";
272         } else {
273             echo "    \n$convScreenShots[$k] = ''      ";
274         }
275     }
276 ?>
277
278
279     /** Swaps the converted screen shot image
280 */
281     function swapImage( key ) {
282         document.screenshot.src = 'images/' + screenShots[key];
283     }
284
285 </script>
286
287 </head>
288
289 <body bgcolor="#ffffff" onload="javascript:initialize()"
topmargin="0" leftmargin="0" marginheight="0"
marginwidth="0">
290
291     <!-- onsubmit="return validate()" -->
292
293     <form name="dataForm" action="builder.php" method="get"
target="OutputFrame">
294
295     <div class="tab-pane" id="tabPanel">

```

```

296 <script type="text/javascript">
297     var tpl = new WebFXTabPane( document.getElementById( "tabPanel" ) );
298 </script>
299     <div class="tab-page" id="tab_intro">
300         <h2 class="tab">Introduction</h2>
301         Welcome to <b>docBuilder</b>.
302         <p>This is the new web-interface for running, in our opinion, the best in-code
documentation compiler there is: <b>phpDocumentor</b>. </p>
303         <p>What's new in this release? Heaps of things, but here are the
headlines:</p>
304         <ul>
305             <li>Much greater support for PEAR on both windows and linux</li>
306             <li>CHM, PDF and XML:DocBook/peardoc2 converters are all stable!</li>
307             <li>New tokenizer-based parser is literally twice as fast as the old parser
(requires PHP 4.3.0+)</li>
308             <li>New external user-level manual parsing and generation allows cross-
linking between API docs and DocBook-format tutorials/manuals!</li>
309             <li>Color syntax source highlighting and cross-referencing with documentation
of source code in HTML, CHM and PDF with customizable templating</li>
310             <li>New Configuration files simplify repetitive and complex documentation
tasks</li>
311             <li>Brand new extensive manual - which can be generated directly from the
source using makedocs.ini!</li>
312             <li>Many improvements to in-code API documentation including new tags, and
better handling of in-code html tags</li>
313             <li>New XML:DocBook/peardoc converter makes generating PEAR manual formats
easy for PEAR developers along with the --pear command-line switch</li>
314             <li>Many new HTML templates, all of them beautiful thanks to Marco von
Ballmoos</li>
315             <li>A brand new web interface thanks to Andrew Eddie!</li>
316         </ul>
317     </div>
318
319     <div class="tab-page" id="tab_config">
320         <h2 class="tab">Config</h2>
321         <table cellspacing="0" cellpadding="3" border="0">
322             <tr>
323                 <td colspan="2"><b>Use a pre-created config file for form
values.</b></td>
324             </tr>
325             <tr>
326                 <td nowrap="nowrap">
327                     <b>change config directory:</b>
328                 </td>
329                 <td width="100%">
330                     <input size="20" type="text" name="altuserdir"
value="" /><input type="SUBMIT" value="Change"
name="submitButton" onclick="document.dataForm.target='DataFrame';
document.dataForm.action = 'config.php';document.dataForm.submit();">
331                     <?php
332                     if (!empty($_REQUEST['altuserdir'])) {
333                         print '<br><i>changed to
<b>' . $_REQUEST['altuserdir'] . '</b></i>' ;
334                     }
335                 ?>
336                 </td>
337             </tr>
338             <tr>
339                 <td nowrap="nowrap">
340                     <b>Choose a config:</b>
341                 </td>
342                 <td>
343
344                     <select name="setting_useconfig">
345                         <option value="" <?php
346 (empty($_REQUEST['altuserdir'])) print 'selected'; ?>>don't use config file</option>
347                         <?php
348                         $dirs = array();
349                         $dirs = phpdDocumentor_ConfigFileList($configdir);
350                         $path = '';
351                         $sel = ' selected';
352                         if (!empty($_REQUEST['altuserdir'])) $path = $configdir . PATH_DELIMITER;
353                         else $sel = '';
354                         foreach($dirs as $configfile)
355                         {
356                             print '<option
value="' . $path . $configfile . '" ' . $sel . '>' . $configfile . ".ini</option>\n";
357                         $sel = '';
358                         ?>

```

```

359             </select>
360             <input type="SUBMIT" value="Go"
361         </td>
362     </tr>
363     <tr>
364         <td colspan="2">
365             Normally, phpDocumentor uses the form values from this form to set up parsing.
366             In version 1.2, phpDocumentor allows you to "save" form values in configuration files so
367             that you can replicate common complicated documentation tasks with only one time. Just choose a
368             config file below or create a new one and refresh this page to choose it.
369         </td>
370     </tr>
371     </table>
372 </div>
373 <div class="tab-page" id="tab_files">
374     <h2 class="tab">Files</h2>
375     <table cellspacing="0" cellpadding="3" border="0">
376         <tr>
377             <td align="right" valign="top" nowrap="nowrap">
378                 <b>Files<br />to parse</b>
379                 <br />
380                 <a href="javascript:addFile(document.dataForm.setting_filename)" title="Add the file in the help box">
381                     <?php
382                         echo showImage( 'images/rc-gui-install-24.png', '24', '24' );
383                     ?></a>
384             </td>
385             <td valign="top">
386                 <textarea rows="5" cols="60" name="setting_filename" class="text"></textarea>
387                 This is a group of comma-separated names of php files or tutorials that will be
388                 processed by phpDocumentor.
389             </td>
390         <tr>
391             <td align="right" valign="top" nowrap="nowrap">
392                 <b>Directory<br />to parse</b>
393                 <br />
394                 <a href="javascript:addFile(document.dataForm.setting_directory)" title="Add the file in the help box">
395                     <?php
396                         echo showImage( 'images/rc-gui-install-24.png', '24', '24' );
397                     ?></a>
398             </td>
399             <td valign="top">
400                 <textarea rows="5" cols="60" name="setting_directory" class="text" title=""></textarea>
401             </td>
402             <td valign="top" class="small">
403                 This is a group of comma-separated directories where php files or tutorials are
404                 found that will be processed by phpDocumentor. phpDocumentor automatically parses subdirectories
405             </td>
406         <tr>
407             <td align="right" valign="top" nowrap="nowrap">
408                 <b>Files<br />to ignore</b>
409                 <br />
410                 <a href="javascript:addFile(document.dataForm.setting_ignore)" title="Add the file in the help box">
411                     <?php
412                         echo showImage( 'images/rc-gui-install-24.png', '24', '24' );
413                     ?></a>
414             </td>
415             <td valign="top">
416                 <textarea rows="5" cols="60" class="text" name="setting_ignore"></textarea>
417             </td>
418             <td valign="top" class="small">
419                 A list of files (full path or filename), and patterns to ignore. Patterns may
420                 use wildcards * and ?. To ignore all subdirectories named "test" for example, using
421                 "test/" To ignore all files and directories with test in their name use "*test*"
422             </td>
423         <tr>
424             <td align="right" valign="top" nowrap="nowrap">
425                 <b>Packages<br />to parse</b>

```

```

425             </td>
426             <td valign="top">
427                 <textarea rows="4" cols="60" class="text"
name="setting_packageoutput"></textarea>
428             </td>
429             <td valign="top" class="small">
430                 The parameter packages is a group of comma separated names of abstract packages
that will be processed by phpDocumentor. All package names must be separated by commas.
431             </td>
432         </tr>
433     </table>
434 </div>
435
436
437     <div class="tab-page" id="tab_output">
438         <h2 class="tab">Output</h2>
439         <table cellspacing="0" cellpadding="3" border="0">
440             <tr>
441                 <td align="right" valign="top" nowrap="nowrap">
442                     <b>Target</b>
443                     <br />
444                     <a href="javascript:addFile(document.dataForm.setting_target)" title="Add the file in the help box">
445                         <?php
446                         echo showImage( 'images/rc-gui-install-24.png', '24', '24' );
447                     ?></a>
448                 </td>
449                 <td valign="top">
450                     <input type="text" name="setting_target"
size="60" class="text" />
451                 </td>
452                 <td valign="top" class="small">
453                     Target is the directory where the output produced by phpDocumentor will reside.
454                 </td>
455             </tr>
456             <tr>
457                 <td align="right" valign="top" nowrap="nowrap">
458                     <b>Output<br />Format</b>
459                 </td>
460                 <td valign="top">
461                     <textarea cols="60" rows="3"
name="setting_output" class="text">HTML:Smarty:default</textarea>
462                     <br />
463                     Output type:Converter name:template name
464                     <br />
465             <?php
466                 echo htmlArraySelect( $converters, 'ConverterSetting', 'size="1'
class="text"
onchange="swapImage(this.options[this.options.selectedIndex].value);"
'HTML:Smarty:default' );
467             ?>
468                     <br />
469                     <a
href="javascript:addConverter(document.dataForm.setting_output)">
470                         Add the converter in the help box
471                     </a>
472                     <br />
473                     <br />
474                     
475                     </td>
476                     <td valign="top" class="small">
477                         Outputformat may be HTML, XML, PDF, or CHM (case-sensitive) in version 1.2.
478                         <br />There is only one Converter for both CHM and PDF:<br
/><i>default</i>.
479                         <br />There are 2 HTML Converters:<br /><i>frames</i>
or <i>Smarty</i>.
480                         <br /><b>frames templates</b> may be any of:
481                         <br />
482                         <i>default, earthli, 10133t, phpdoc.de, phptmllib, phpedit, DOM/default,
DOM/earthli, DOM/10133t, DOM/phptmllib, or DOM/phpdoc.de</i>.
483                         <br />
484                         <b>Smarty templates</b> may be any of:
485                         <br />
486                         <i>default, Hands, or PHP</i>
487                         <br />
488                         <strong>XML:DocBook/pearlbook2:default</strong> is the only choice
for XML in 1.2.2
489                     </td>

```

```

490         </tr>
491     </table>
492   </div>
493
494   <div class="tab-page" id="tab_options">
495     <h2 class="tab">Options</h2>
496     <table cellspacing="0" cellpadding="3" border="0">
497       <tr>
498         <td align="right" nowrap="nowrap">
499           <b>Generated Documentation Title</b>
500         </td>
501         <td>
502           <input type="text" name="setting_title"
size="40" value="Generated Documentation" class="text">
503           </td>
504           <td class="small">
505             Choose a title for the generated documentation
506           </td>
507       </tr>
508       <tr>
509         <td nowrap="nowrap">
510           <b>Default Package Name</b>
511         </td>
512         <td>
513           <input type="TEXT" name="setting_defaultpackagename"
size="40" value="default" class="text" />
514           </td>
515           <td class="small">
516             Choose a name for the default package
517           </td>
518       </tr>
519       <tr>
520         <td nowrap="nowrap">
521           <b>Default Category Name</b>
522         </td>
523         <td>
524           <input type="TEXT" name="setting_defaultcategoryname"
size="40" value="default" class="text" />
525           </td>
526           <td class="small">
527             Choose a name for the default category. This is only used by the peardoc2
converter
528           </td>
529       </tr>
530       <tr>
531         <td nowrap="nowrap">
532           <b>Custom Tags</b>
533         </td>
534         <td>
535           <input type="text" name="setting_customtags"
size="40" class="text" />
536           </td>
537           <td class="small">
538             Custom Tags is a comma-separated list of tags you want phpDocumentor to include
as valid tags in this parse. An example would be 'value, size' to allow @value and @size tags.
539           </td>
540       </tr>
541       <tr>
542         <td nowrap="nowrap">
543           <b>Parse @access private and @internal/{@internal}</b>
544         </td>
545         <td nowrap="nowrap">
546           <input type="checkbox" name="setting_parseprivate"
value="on" />
547           </td>
548           <td class="small">
549             The parameter Parse @access private tells phpDocumentor whether to parse
elements with an '@access private' tag in their docblock. In addition, it will turn on parsing of
@internal tags and inline {@internal} sections
550           </td>
551       </tr>
552       <tr>
553         <td nowrap="nowrap">
554           <b>Generate Highlighted Source Code</b>
555         </td>
556         <td nowrap="nowrap">
557           <input type="checkbox" name="setting_sourcecode"
value="on" />
558           </td>
559           <td class="small">

```

```

560             The parameter Generate Highlighted Source Code tells phpDocumentor whether to
561             generate highlighted XRef source code similar to PHP-XRef output.
562             </td>
563             <tr>
564                 <td nowrap="nowrap">
565                     <b>JavaDoc-compliant<br />Description parsing.</b>
566                 </td>
567                 <td>
568                     <input type="checkbox" name="setting_javadocdesc"
569                     value="on" />
570                     <td class="small">
571                         Normally, phpDocumentor uses several rules to determine the short description.
572                         This switch asks phpDocumentor to simply search for the first period (.) and use it to delineate the
573                         short description. In addition, the short description will not be separated from the long
574                         description.
575                     </td>
576                     </tr>
577                     <tr>
578                         <td nowrap="nowrap">
579                             <b>PEAR package repository parsing</b>
580                         </td>
581                         <td>
582                             PEAR package repositories have specific requirements:
583                             <ol>
584                                 <li>Every package is in a directory with the same name.</li>
585                                 <li>All private data members and methods begin with an underscore
586 (function _privfunction()).</li>
587                                 <li>_Classname() is a destructor</li>
588                             </ol>
589                             This option recognizes these facts and uses them to make assumptions about
590                             packaging and access levels. Note that with PHP 5, the destructor option will be obsolete.
591                         </td>
592                     </tr>
593                 </table>
594             </div>
595             <div class="tab-page" id="tab_credits">
596                 <h2 class="tab">Credits</h2>
597                 phpDocumentor written by Joshua Eichorn
598                 <br />Web Interface originally written by Juan Pablo Morales, enhanced by Greg
599                 Beaver and super-charged by Andrew Eddie
600                 <p>
601                     Joshua Eichorn <a href="mailto:jeichorn@phpdoc.org">jeichorn@phpdoc.org</a>
602                     <br>Juan Pablo Morales <a href="mailto:ju-moral@uniandes.edu.co">ju-moral@uniandes.edu.co</a>
603                     <br>Gregory Beaver <a href="mailto:cellog@php.net">cellog@php.net</a>
604                     <br>Andrew Eddie <a href=
605 "mailto:eddieajau@users.sourceforge.net">eddieajau@users.sourceforge.net</a>
606                     </p>
607                     <p>
608                         If you have any problems with phpDocumentor, please visit the website:
609                         <a href='http://phpdoc.org'>phpdoc.org</a>, or
610                         <a
611 href="http://pear.php.net/bugs/search.php?cmd=display&package_name[ ]=PhpDocumentor&status
612 =Open">
613                     submit a bug at PEAR</a>
614                     </p>
615                     <!-- Created: Tue Jun 26 18:52:40 MEST 2001 -->
616                     <!-- hhmts start -->
617                     <pre>
618                         Last modified: $Date: 2007-04-19 15:20:57 -0500 (Thu, 19 Apr 2007) $
619                         Revision: $Revision: 234145 $
620                     </pre>
621             </div>
622             <div class="tab-page" id="tab_links">
623                 <h2 class="tab">Links</h2>
624                 <ul>
625                     <li><a target="_top"
626 href="http://www.phpdoc.org/manual.php">phpDocumentor manual</a> - Learn how to
627 use phpDocumentor to document your PHP source code</li>
628                     <li><a target="_top"

```

```
href="http://pear.php.net/package/PhpDocumentor ">phpDocumentor homepage</a> on
PEAR</li>
625      <li><a target="_top"
href="http://phpdoc.sourceforge.net/">phpDocumentor homepage</a> on
SourceForge</li>
626          <li><a target="_top"
href="http://freshmeat.net/projects/phpdocu">Freshmeat record</a> - subscribe
here</li>
627      </ul>
628  </div>
629 </div>
630 <input type="hidden" name="interface" value="web">
631 <input type="hidden" name="dataform" value="true">
632
633 </form>
634
635 <script type="text/javascript">
636
637     tp1.addTabPage( document.getElementById( "tab_intro" ) );
638     tp1.addTabPage( document.getElementById( "tab_config" ) );
639     tp1.addTabPage( document.getElementById( "tab_files" ) );
640     tp1.addTabPage( document.getElementById( "tab_output" ) );
641     tp1.addTabPage( document.getElementById( "tab_options" ) );
642     tp1.addTabPage( document.getElementById( "tab_credits" ) );
643     tp1.addTabPage( document.getElementById( "tab_links" ) );
644     setupAllTabs();
645 </script>
646
647 </body>
648 </html>
```

# File Source for top.php

Documentation for this file is available at [top.php](#)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <?php
3  /**
4   * phpDocumentor :: docBuilder Web Interface
5   *
6   * Advanced Web Interface to phpDocumentor
7   *
8   * PHP versions 4 and 5
9   *
10  * Copyright (c) 2003-2006 Andrew Eddie, Greg Beaver
11  *
12  * LICENSE:
13  *
14  * This library is free software; you can redistribute it
15  * and/or modify it under the terms of the GNU Lesser General
16  * Public License as published by the Free Software Foundation;
17  * either version 2.1 of the License, or (at your option) any
18  * later version.
19  *
20  * This library is distributed in the hope that it will be useful,
21  * but WITHOUT ANY WARRANTY; without even the implied warranty of
22  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23  * Lesser General Public License for more details.
24  *
25  * You should have received a copy of the GNU Lesser General Public
26  * License along with this library; if not, write to the Free Software
27  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28  *
29  * @package    phpDocumentor
30  * @author     Andrew Eddie
31  * @author     Greg Beaver <cellog@php.net>
32  * @copyright  2003-2006 Andrew Eddie, Greg Beaver
33  * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version    CVS: $Id: top.php 212211 2006-04-30 22:18:14Z cellog $
35  * @filesource
36  * @see       phpdoc.php
37  */
38  if ('@DATA-DIR@' != '@'. 'DATA-DIR@' )
39  {
40      include_once( "PhpDocumentor/phpDocumentor/common.inc.php" );
41  } else {
42      $path = dirname(dirname(__FILE__));
43      include_once( "      $path/phpDocumentor/common.inc.php" );
44  }
45 //
46 // -----
47 // | phpDocumentor :: docBuilder Web Interface
48 // +-----+
49 // | Copyright (c) 2003 Andrew Eddie, Greg Beaver
50 // +-----+
51 // | This source file is subject to version 3.00 of the PHP License,
52 // | that is available at http://www.php.net/license/3_0.txt.
53 // | If you did not receive a copy of the PHP license and are unable to
54 // | obtain it through the world-wide-web, please send a note to
55 // | license@php.net so we can mail you a copy immediately.
56 // +-----+
57 //
58 ?>
59 <html>
60 <head>
61     <title>
62         Form to submit to phpDocumentor v<?php print      PHPDOCUMENTOR_VER; ?>
63     </title>
64     <style type="text/css">
65         body, td, th {
66             font-family: verdana,sans-serif;
67             font-size: 10pt;
```

```
68      }
69      .title {
70          font-size: 12pt;
71      }
72  </style>
73 </head>
74
75  <body bgcolor="#0099cc" topmargin="0" leftmargin="0"
marginheight="0" marginwidth="0">
76
77  <table width="100%" cellspacing="0" cellpadding="0">
78  <tr>
79      <td bgcolor="#0099cc" height="35" width="100"
nowrap="nowrap">
80          
81      </td>
82      <td bgcolor="#0099cc" width="100%">
83          <span class="title"><strong>docBuilder</strong> ::<br>
phpDocumentor v<?php print      PHPDOCUMENTOR_VER; ?> Web Interface</span>
84      </td>
85  </tr>
86  </table>
87
88  </body>
89  </html>
```

# File Source for new\_phpdoc.php

Documentation for this file is available at [new\\_phpdoc.php](#)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <?php
3  /**
4   * Advanced Web Interface to phpDocumentor
5   *
6   * PHP versions 4 and 5
7   *
8   * @category ToolsAndUtilities
9   * @package phpDocumentor
10  * @author Juan Pablo Morales <ju-moral@uniandes.edu.co>
11  * @author Joshua Eichorn <jeichorn@phpdoc.org>
12  * @author Gregory Beaver <cellog@php.net>
13  * @author Chuck Burgess <ashnazg@php.net>
14  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
15  * @link http://pear.php.net/package/PhpDocumentor
16  * @see phpdoc.php
17  * @filesource
18  * @todo CS cleanup - change package to PhpDocumentor
19  * @deprecated redirects automatically to docbuilder
20  *          (see {@link docbuilder/index.html})
21  */
22 ?>
23
24 <html>
25 <head>
26 <title>PhpDocumentor (deprecated web interface)</title>
27 <META HTTP-EQUIV="Refresh" CONTENT="5; URL=docbuilder/" />
28 </head>
29 <body>
30 This PhpDocumentor web interface is deprecated...
31 you will be redirected to the DocBuilder interface.
32 </body>
33 </html>
```

# File Source for phpdoc.php

*Documentation for this file is available at [phpdoc.php](#)*

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <?php
3  /**
4   * Original Web Interface to phpDocumentor
5   *
6   * PHP versions 4 and 5
7   *
8   * @category ToolsAndUtilities
9   * @package phpDocumentor
10  * @author Juan Pablo Morales <ju-moral@uniandes.edu.co>
11  * @author Joshua Eichorn <jeichorn@phpdoc.org>
12  * @author Gregory Beaver <cellog@php.net>
13  * @author Chuck Burgess <ashnazg@php.net>
14  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
15  * @link http://pear.php.net/package/PhpDocumentor
16  * @filesource
17  * @todo CS cleanup - change package to PhpDocumentor
18  * @deprecated redirects automatically to docbuilder
19  *          (see {@link docbuilder/index.html})
20  */
21 ?>
22
23 <html>
24 <head>
25 <title>PhpDocumentor (deprecated web interface)</title>
26 <META HTTP-EQUIV="Refresh" CONTENT="5; URL=docbuilder/">
27 </head>
28 <body>
29 This PhpDocumentor web interface is deprecated...
30 you will be redirected to the DocBuilder interface.
31 </body>
32 </html>
```

# File Source for Classes.inc

Documentation for this file is available at [Classes.inc](#)

```
1  <?php
2  /**
3   * Intermediate class parsing structure.
4   *
5   * phpDocumentor :: automatic documentation generator
6   *
7   * PHP versions 4 and 5
8   *
9   * Copyright (c) 2001-2007 Gregory Beaver
10  *
11  * LICENSE:
12  *
13  * This library is free software; you can redistribute it
14  * and/or modify it under the terms of the GNU Lesser General
15  * Public License as published by the Free Software Foundation;
16  * either version 2.1 of the License, or (at your option) any
17  * later version.
18  *
19  * This library is distributed in the hope that it will be useful,
20  * but WITHOUT ANY WARRANTY; without even the implied warranty of
21  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22  * Lesser General Public License for more details.
23  *
24  * You should have received a copy of the GNU Lesser General Public
25  * License along with this library; if not, write to the Free Software
26  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27  *
28  * @category ToolsAndUtilities
29  * @package phpDocumentor
30  * @author Greg Beaver <cellog@php.net>
31  * @copyright 2001-2007 Gregory Beaver
32  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33  * @version CVS: $Id: Classes.inc 243933 2007-10-10 01:18:25Z ashnazg $
34  * @filesource
35  * @link http://www.phpdoc.org
36  * @link http://pear.php.net/PhpDocumentor
37  * @see parserDocBlock, parserInclude, parserPage, parserClass
38  * @see parserDefine, parserFunction, parserMethod, parserVar
39  * @since 1.0rc1
40  * @todo CS cleanup - change package to PhpDocumentor
41  */
42 /**
43 * Intermediate class parsing structure.
44 *
45 * The {@link phpDocumentor_IntermediateParser} class uses this class and its
46 * cousin, {@link ProceduralPages} to organize all parsed source code elements.
47 * Data is fed to each immediately after it is parsed, and at conversion time,
48 * everything is organized.
49 *
50 * The Classes class is responsible for all inheritance, including resolving
51 * name conflicts between classes, determining which classes extend other
52 * classes, and is responsible for all inheritance of documentation.
53 * {@internal
54 * This structure parses classes, vars and methods by file, and then iterates
55 * over the class tree to set up inheritance. The {@link Inherit()}
56 * method is the meat of the class, and processes the class trees from root to
57 * branch, ensuring that parsing order is unimportant.}}
58 *
59 * @category ToolsAndUtilities
60 * @package phpDocumentor
61 * @author Greg Beaver <cellog@php.net>
62 * @copyright 2001-2007 Gregory Beaver
63 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
64 * @version Release: @VER@
65 * @link http://www.phpdoc.org
66 * @link http://pear.php.net/PhpDocumentor
67 * @since 1.0rc1
```

```

68 * @todo      CS cleanup - change package to PhpDocumentor
69 */
70 class Classes
71 {
72     /**#@+
73     * @access private
74     */
75     /**
76     * file being parsed, used in every add function to match up elements with
77     * the file that contains them
78     *
79     * This variable is used during parsing to associate class elements added
80     * to the data structures that contain them with the file they reside in
81     * @see addClass(), addMethod(), addVar(), nextFile()
82     * @var string
83     */
84     var $curfile;
85     /**
86     * class being parsed, used to match up methods and vars with their parent
87     * class
88     *
89     * This variable is used during parsing to associate class elements added
90     * to the data structures that contain them with the file they reside in
91     * @see addMethod(), addVar()
92     * @var string
93     */
94     var $curclass;
95
96     /**
97     * Used when a definite match is made between a parent class and a child
98     * class
99     *
100    * This variable is used in post-parsing.
101   *
102   * Format:<pre>
103   * array(
104   *     parent => array(
105   *         parentfile => array(
106   *             child => childfile
107   *         )
108   *     )
109   * )</pre>
110   * @var array
111   */
112   var $definitechild;
113   /**
114   * array of parsed classes organized by the name of the file that contains
115   * the class.
116   *
117   * Format:<pre>
118   * array(
119   *     filename => array(
120   *         classname => {@link parserClass}
121   *     )
122   * )</pre>
123   * @var array
124   */
125   var $classesbyfile = array();
126   /**
127   * array of file names organized by classes that are in the file.
128   *
129   * This structure is designed to handle name conflicts. Two files can
130   * contain classes with the same name, and this array will record both
131   * filenames to help control linking and inheritance errors
132   *
133   * Format:<pre>
134   * array(
135   *     classname => array(
136   *         name of file containing classname,
137   *         name of file 2 containing classname,
138   *         ...
139   *     )
140   * )</pre>
141   * @var array
142   */
143   var $classesbynamefile = array();
144   /**
145   * array of parsed methods organized by the file that contains them.
146   *
147   * Format:<pre>
```

```

148 * array(
149 *     filename => array(
150 *         classname => array(
151 *             {@link parserMethod} 1,
152 *             {@link parserMethod} 2,
153 *             ...
154 *         )
155 *     )
156 * )</pre>
157 * @var array
158 */
159 var $methodsbyfile = array();
160 /**
161 * array of parsed vars organized by the file that contains them.
162 *
163 * Format:<pre>
164 * array(
165 *     filename => array(
166 *         classname => array(
167 *             {@link parserVar} 1,
168 *             {@link parserVar} 2,
169 *             ...
170 *         )
171 *     )
172 * )</pre>
173 * @var array
174 */
175 var $varsbyfile = array();
176 /**
177 * array of parsed class constants organized by the file that contains them.
178 *
179 * Format:<pre>
180 * array(
181 *     filename => array(
182 *         classname => array(
183 *             {@link parserConst} 1,
184 *             {@link parserConst} 2,
185 *             ...
186 *         )
187 *     )
188 * )</pre>
189 * @var array
190 */
191 var $constsbyfile = array();
192 /**
193 * keeps track of extend declarations by file, used to find inheritance
194 *
195 * Format:<pre>
196 * array(
197 *     filename => array(
198 *         classname => parentclassname
199 *     )
200 * )</pre>
201 * @var array
202 */
203 var $extendsbyfile = array();
204 /**
205 * Keeps track of child classes by file.
206 * Since phpDocumentor can document collections of files that contain name
207 * conflicts (PHP would give a fatal error), it
208 * is impossible to assume a class that declares "extends foo" necessarily
209 * extends the class foo in file X. It could be an
210 * extended class of class foo in file Y. Because of this, phpDocumentor
211 * relies on packaging to resolve the name conflict
212 * This array keeps track of the packages of a child class
213 *
214 * Format:<pre>
215 * array(
216 *     parentclassname => array(
217 *         filename => array(
218 *             childclassname => array(
219 *                 packagename,
220 *                 packagename
221 *             )
222 *         )
223 *     )
224 * )</pre>
225 * @var array
226 */
227 var $classchildrenbyfile = array();

```

```

228 /**
229 * Keeps track of class packages found in a file.
230 * This is used in {@link getParentClass()} to determine the number of
231 * packages in a file, in order to resolve inheritance issues
232 * Format:<pre>
233 * array(
234 *     filename => array(
235 *         packagename1,
236 *         packagename2,
237 *         ...
238 *     )
239 * )</pre>
240 * @var array
241 */
242 var $classpackagebyfile = array();
243 /**
244 * a tree of class inheritance by name.
245 *
246 * Format:<pre>
247 * array(
248 *     childname => parentname,
249 *     childname1 => parentname1,
250 *     rootname => 0,
251 *     ...
252 * )</pre>
253 * @var array
254 * @see Converter::generateSortedClassTreeFromClass()
255 */
256 var $classparents = array();
257 /**
258 * Keeps track of package and subpackage for each class name, organized
259 * by package
260 *
261 * Format:<pre>
262 * array(
263 *     classname => array(
264 *         path => array(
265 *             package,
266 *             subpackage
267 *         ),
268 *         path2 => array(
269 *             package,
270 *             subpackage
271 *         ),
272 *         ...
273 *     )
274 * )</pre>
275 * @var array
276 */
277 var $classpathpackages = array();
278 /**
279 * used to delete duplicates in the same package to avoid documentation errors
280 *
281 * Specifically used in {@link Converter::checkKillClass()}
282 */
283 var $killclass = array();
284 /**
285 * array of methods by package and class
286 *
287 * format:<pre>
288 * array(packagename =>
289 *     array(classname =>
290 *         array(methodname1 => {@link parserMethod} class,
291 *               methodname2 => {@link parserMethod} class,...)
292 *         )
293 *     )
294 * )</pre>
295 * @var array
296 * @see Converter
297 */
298 var $methods = array();
299 /**
300 * array of class variables by package and class
301 *
302 * format:<pre>
303 * array(packagename =>
304 *     array(classname =>
305 *         array(variablename1 => {@link parserVar} class,
306 *               variablename2 => {@link parserVar} class,...)

```

```

308         )
309         )
310         )</pre>
311     * @var array
312     * @see Converter
313     */
314 var $vars = array();
315
316 /**
317  * array of class variables by package and class
318  *
319  * format:<pre>
320  * array(packagename =>
321  *         array(classname =>
322  *                 array(constname1 => {@link parserConst} class,
323  *                     constname2 => {@link parserConst} class,...,
324  *                     ...
325  *                 )
326  *             )</pre>
327  * @var array
328  * @see Converter
329  */
330 var $consts = array();
331 /**
332  * Reverse class_packages_by_file, used to prevent duplicates
333  * @var array Format: array(packagename => 1)
334  */
335 var $revcpbf = array();
336 /**
337  * All classes with no parents (no extends clause) are tracked in this array
338  * by the file that contains them.
339  *
340  * Format:<pre>
341  * array(
342  *         classname => array(
343  *             name of file1 that contains root classname,
344  *             name of file2 that contains root classname,
345  *             ...
346  *         )
347  *     )</pre>
348  * @var array
349  */
350 var $roots = array();
351 /**
352  * All classes with a parent that was not parsed are included in this array
353  *
354  * Format:<pre>
355  * array(
356  *         classname => array(
357  *             name of file1 that contains root classname,
358  *             name of file2 that contains root classname,
359  *             ...
360  *         )
361  *     )</pre>
362  * @var array
363  */
364 var $specialRoots = array();
365
366 /**
367  * array of all files that contain classes with the same name
368  * @var array Format: (classname => array(path1, path2,...))
369  */
370 var $potentialclassconflicts = array();
371
372 /**
373  * array of all inter-package name conflicts of classes
374  *
375  * This array allows documentation of PHP namespace conflicts that would
376  * occur should a user try to include these files in the same file
377  * @var array Format: (classname => array(path1, path2,...))
378  */
379 var $classconflicts = array();
380 /**
381  * While parsing, add a class to the list of parsed classes
382  *
383  * sets up the {@link $classesbyfile}, {@link $classesbynamefile}, {@link $extendsbyfile},
384  * {@link $classchildrenbyfile}, {@link $roots} arrays, and sets {@link $curclass}
385  *
386  * @param parserClass &$element      element is a {@link parserClass}

```

```

388 *
389 * @return void
390 * @uses addPackageToFile() marks the current class's package as being
391 * present in a file
392 */
393 function addClass(& $element)
394 {
395     $this-> curclass = $element-> getName();
396     $element-> curfile = $this-> curfile;
397     if (isset($this-> classesbyfile[$this-> curfile][$this-> curclass])) {
398         addWarning(PDERROR_ELEMENT_IGNORED,
399                     'class', $this-> curclass, $this-> curfile);
400         $this-> curclass = false;
401         return;
402     }
403     $this->
404     classesbyfile
405         [$this-> curfile][$this-> curclass]
406         = $element;
407     $this->
408         classesbynamefile[$this-> curclass] []
409         = $this-> curfile;
410     $this->
411         extendsbyfile[$this-> curfile][$this-> curclass]
412         = $element-> getExtends();
413     $this->
414         classchildrenbyfile[$element-> getExtends()]
415             [$this-> curfile][$this-> curclass] []
416             = $element-> docblock-> package;
417     if ($element-> docblock-> getExplicitPackage())
418         $this-> addPackageToFile($element-> docblock-> package);
419     if (!$element-> getExtends()) {
420         $this-> roots[$this-> curclass] [] = $this-> curfile;
421     }
422 }
423 /**
424 * While parsing, add a method to the list of parsed methods
425 *
426 * sets up the {@link $methodsbyfile} array using {@link $curfile} and
427 * {@link $curclass}
428 *
429 * @param parserMethod &$element      element is a {@link parserMethod}
430 *
431 * @return void
432 */
433 function addMethod(& $element)
434 {
435     if (!$this-> curclass) return;
436     $this-> methodsbyfile[$this-> curfile][$this-> curclass] [] = $element;
437 }
438 /**
439 * While parsing, add a variable to the list of parsed variables
440 *
441 * sets up the {@link $varsbyfile} array using {@link $curfile}
442 * and {@link $curclass}
443 *
444 * @param parserVar &$element      element is a {@link parserVar}
445 *
446 * @return void
447 */
448 function addVar(& $element)
449 {
450     if (!$this-> curclass) return;
451     $this-> varsbyfile[$this-> curfile][$this-> curclass] [] = $element;
452 }
453 /**
454 * While parsing, add a variable to the list of parsed variables
455 *
456 * sets up the {@link $constsbyfile} array using {@link $curfile}
457 * and {@link $curclass}
458 *
459 * @param parserConst &$element      element is a {@link parserConst}
460 *
461 * @return void
462 */
463 function addConst(& $element)
464 
```

```

467     {
468         if (!$this-> curclass) return;
469         $this-> constsbyfile[$this-> curfile][$this-> curclass][] = $element;
470     }
471
472 /**
473 * Prepare to parse a new file
474 *
475 * sets {@link $curfile} to $file and {@link $curclass}
476 * to false (no class being parsed)
477 *
478 * @param string $file file currently being parsed
479 *
480 * @return void
481 */
482 function nextFile($file)
483 {
484     $this-> curfile = $file;
485     $this-> curclass = false;
486 }
487
488 /**
489 * Mark a package as being used in a class
490 *
491 * {@source}
492 *
493 * @param string $package package name
494 *
495 * @return void
496 */
497 function addPackageToFile($package)
498 {
499     if (!isset($this-> revcpbf[$this-> curfile][$package]))
500         $this-> classpackagebyfile[$this-> curfile][] = $package;
501     $this-> revcpbf[$this-> curfile][$package] = 1;
502 }
503
504 /**
505 * Find the parent class of $class, and set up structures to note this fact
506 *
507 * Modifies the {@link parserClass} element in {@link $classesbyfile} to use
508 * the parent's package, and inherit methods/vars
509 *
510 * @param string $class child class to find parent class
511 * @param string $file file child class is located in
512 *
513 * @return void
514 * @uses $definitechild if a match is made between a parent class and parameter
515 * $class in file $file, then definitechild is set here
516 * @uses getParentClass() to find the parent class
517 */
518 function setClassParent($class,$file)
519 {
520     if (is_array($par = $this-> getParentClass($class, $file))) {
521         // (for debugging)
522         // phpDocumentor_out("$file class $class extends "
523         // . $par[1] . " file ". $par[0] . "\n");
524
525         $this-> classesbyfile[$file][$class]-> setParent($par[1], $par[0], $this);
526         $this-> definitechild[$par[1][$par[0]][$class]] = $file;
527     } else {
528         $this-> classesbyfile[$file][$class]-> setParentNoClass($par);
529     }
530 }
531
532 /**
533 * Main processing engine for setting up class inheritance.
534 *
535 * This function uses {@link $roots} to traverse the inheritance tree via
536 * {@link processChild()} and returns the data structures
537 * phpDocumentor_IntermediateParser needs to convert parsed data
538 * to output using {@link phpDocumentor_IntermediateParser::Convert()}
539 *
540 * @param phpDocumentor_IntermediateParser &$render the renderer object
541 *
542 * @return void
543 * @uses processChild() set up inheritance
544 * @todo CS Cleanup - rename to "inherit" for CamelCaps naming standard
545 */
546 function Inherit(& $render)

```

```

547
548     {
549         phpDocumentor_out( "\nProcessing Class Inheritance\n\n" );
550         flush();
551         phpDocumentor_out( "\nProcessing Root Trees\n\n" );
552         flush();
553         foreach ($this-> roots as $class =>      $files) {
554             for ($i=0; $i< count($files); $i++) {
555                 $this-> processChild($render, $class, $files[$i]);
556             }
557         }
558         if (0)
559         foreach ($this-> classesbyfile as $i =>      $j) {
560             foreach ($j as $k =>      $m) {
561                 var_dump($i, $k);
562                 if ($i == 'iConverter') {
563                     var_dump($j);
564                 }
565             }
566         }
567         phpDocumentor_out( "\nProcessing leftover classes "
568             . "(classes that extend root classes not found in the same package)\n" );
569         flush();
570         foreach ($this-> classesbyfile as $i =>      $j) {
571             foreach ($j as $k =>      $m) {
572                 $this-> processChild($render, $k, $i, true);
573             }
574         }
575         phpDocumentor_out( "done processing leftover classes\n" );
576         flush();
577         $this-> setupClassConflicts();
578     }
579
580     /**
581      * Transfers actual conflicts from {@link $potentialClassconflicts} to
582      * {@link $classconflicts}
583      *
584      * @return void
585      * @access private
586      * @uses $potentialclassconflicts transfers values to {@link $classconflicts}
587      */
588     function setupClassConflicts()
589     {
590         foreach ($this-> potentialclassconflicts as $class =>      $paths) {
591             if (count($paths) - 1) { //conflict
592                 $package = array();
593                 foreach ($paths as $path) {
594                     // create a list of conflicting classes in each package
595                     if (isset($this-> classpathpackages[$class][$path]))
596                         $package[$this-> classpathpackages[$class][$path][0]][] = $path;
597                 }
598                 foreach ($package as $pathpackages) {
599                     /*
600                      * if at least 2 functions exist in the same package,
601                      * delete all but the first one and add warnings
602                      */
603                     if (count($pathpackages) - 1) {
604                         for ($i=1; $i < count($pathpackages); $i++) {
605                             if (isset($this-> classesbyfile[$pathpackages[$i]])) {
606                                 addWarning(PDERROR_ELEMENT_IGNORED,
607                                         'class', $class, $pathpackages[$i]);
608                                 $this-> killClass($class, $pathpackages[$i]);
609                                 $oth = array_flip($paths);
610                                 unset($paths[$oth[$pathpackages[$i]]]);
611                         }
612                     }
613                 }
614                 $this-> classconflicts[$class] = $paths;
615             }
616         }
617     }
618
619     /**
620      * If a package contains two classes with the same name, this function finds
621      * that conflict
622      *
623      * Returns the {@link $classconflicts} entry for class $class, minus its own path
624      *
625      * @param mixed $class the class name to search for
626      */

```

```

627 * @return mixed returns false if no conflicts,
628 *          or an array of paths containing conflicts
629 */
630 function getConflicts($class)
631 {
632     if (!isset($this-> classconflicts[$class])) return false;
633     $a = array();
634     foreach ($this-> classconflicts[$class] as $conflict) {
635         $a[$this-> classesbyfile[$conflict][$class]-> docblock-> package]
636         = $this-> classesbyfile[$conflict][$class];
637     }
638     return $a;
639 }
640
641 /**
642 * sets up {@link $killclass} for use by Converter::checkKillClass()
643 *
644 * @param mixed $class the class
645 * @param mixed $path the path
646 *
647 * @return void
648 * @access private
649 */
650 function killClass($class,$path)
651 {
652     $this-> killclass[$class][$path] = true;
653 }
654
655 /**
656 * This function recursively climbs up the class tree, setting inherited
657 * information like package and adds the elements to
658 * {@link phpDocumentor_IntermediateParser}.
659 *
660 * Using structures defined in {@link Classes},
661 * the function first sets package information,
662 * and then seeks out child classes.
663 * It uses 3 tests to determine whether a class is a child class.
664 * <ol>
665 *     <li>child class is in the same file as the parent class
666 *         and extends parent class
667 *     </li>
668 *     <li>child class is in a different file and specifies
669 *         the parent's @package in its docblock
670 *     </li>
671 *     <li>child class is in a different file and is in a
672 *         different @package, with one possible parent class
673 *     </li>
674 * </ol>
675 *
676 * @param phpDocumentor_IntermediateParser &$render      the renderer object
677 * @param string                         $class        class to process
678 * @param string                         $file         name of file $class
679 * @param boolean                        $furb        is located in
680 * @param boolean                        $furb        flag used privately
681 * @param boolean                        $furb        to control informational
682 * @param boolean                        $furb        output while parsing
683 * @param boolean                        $furb        (used when processing
684 * @param boolean                        $furb        leftover classes in
685 * @param boolean                        $furb        {@link Inherit()})
686 *
687 * @return void
688 * @global string default package, usually "default"
689 */
690 function processChild(& $render,$class,$file,$furb = false)
691 {
692     global $phpDocumentor_DefaultPackageName;
693     if (isset($this-> classesbyfile[$file][$class]-> processed))
694         return;
695     $this-> potentialclassconflicts[$class][] = $file;
696     if ($furb)
697         phpDocumentor_out("    Processing $class in file $file\n");
698     flush();
699     $this-> classesbyfile[$file][$class]-> processed = true;
700
701     $db = $this-> classesbyfile[$file][$class];
702     $render-> addUses($db, $file);
703     if (!$render-> parsePrivate) {
704         /*
705             * if this class has an @access private,
706             * and parse private is disabled, remove it

```

```

707
708     */
709     if ($db-> docblock-> hasaccess) {
710         $aaa = $db-> docblock-> getKeyword('access');
711         if (is_object($aaa) && $aaa-> getString() == 'private') {
712             if (isset($this-> varsbyfile[$file])
713                 && isset(
714                     $this-> varsbyfile[$file][$class])) {
715                 unset($this-> varsbyfile[$file][$class]);
716             }
717             if (isset($this-> methodsbyfile[$file])
718                 && isset(
719                     $this-> methodsbyfile[$file][$class])) {
720                 unset($this-> methodsbyfile[$file][$class]);
721             }
722             if (isset($this-> constsbyfile[$file])
723                 && isset(
724                     $this-> constsbyfile[$file][$class])) {
725                 unset($this-> constsbyfile[$file][$class]);
726             }
727             $this-> classesbyfile[$file][$class]-> ignore = true;
728             // if this is a root class, remove it from the roots array
729             if (isset($this-> roots[$class])) {
730                 foreach ($this-> roots[$class] as $i => $files) {
731                     // find the file key and unset
732                     if ($files == $file)
733                         unset($this-> roots[$class][$i]);
734                 }
735             /*
736             * if this is a child, remove it from the list
737             * of child classes of its parent
738             */
739             if ($db-> getExtends())
740                 unset($this-> classchildrenbyfile[$db-> getExtends()][$file]);
741         }
742     }
743     if ($render-> packageoutput) {
744         if (!in_array($db-> docblock-> package, $render-> packageoutput)) {
745             if (isset($this-> varsbyfile[$file])
746                 && isset(
747                     $this-> varsbyfile[$file][$class])) {
748                 unset($this-> varsbyfile[$file][$class]);
749             }
750             if (isset($this-> methodsbyfile[$file])
751                 && isset(
752                     $this-> methodsbyfile[$file][$class])) {
753                 unset($this-> methodsbyfile[$file][$class]);
754             }
755             if (isset($this-> constsbyfile[$file])
756                 && isset(
757                     $this-> constsbyfile[$file][$class])) {
758                 unset($this-> constsbyfile[$file][$class]);
759             }
760             $this-> classesbyfile[$file][$class]-> ignore = true;
761             if (isset($this-> roots[$class])) {
762                 foreach ($this-> roots[$class] as $i => $files) {
763                     if ($files == $file) unset($this-> roots[$class][$i]);
764                 }
765             }
766         }
767         $this-> setClassParent($class, $file);
768         $db = $this-> classesbyfile[$file][$class];
769         if ($furb && ! is_array($db-> parent)) {
770             // debug("furb adding $class $file to roots");
771             $this-> specialRoots[$db-> parent][] = array($class, $file);
772         }
773         // fix for 591396
774         if (!$db-> docblock-> getExplicitPackage()) {
775             $a = $render-> proceduralpages-> pagepackages[$file];
776             if ($a[0] != $phpDocumentor_DefaultPackageName) {
777                 // inherit page package
778                 $this-> classesbyfile[$file][$class]-> docblock-> package = $a[0];
779             }
780         }
781         if ($this-> classesbyfile[$file][$class]-> docblock-> package
782             == $render-> proceduralpages-> pagepackages[$file][0]) {
783             if ($this-> classesbyfile[$file][$class]-> docblock-> subpackage == '') {
784                 $this-> classesbyfile[$file][$class]-> docblock-> subpackage
785                     = $render-> proceduralpages-> pagepackages[$file][1];
786             }
787     }

```

```

787 }
788 $db = $this-> classesbyfile[$file][$class];
789 $render-> addPackageParent($db);
790 $render-> addPageIfNecessary($file, $db);
791 if ($access = $db-> docblock-> getKeyword('access')) {
792     if (!is_string($access) && is_object($access))
793         $access = $access-> getString();
794     if (!is_string($access))
795         $access = 'public';
796     if (($access == 'private') && (!$render-> parsePrivate)) {
797         if (isset($this-> varsbyfile[$file])
798             && isset($this-> varsbyfile[$file][$class])) {
799             foreach ($this-> varsbyfile[$file][$class] as $i => $vr) {
800                 $vr-> docblock-> addKeyword('access', 'private');
801                 $this-> varsbyfile[$file][$class][$i] = $vr;
802             }
803         }
804         if (isset($this-> methodsbyfile[$file])
805             && isset($this-> methodsbyfile[$file][$class])) {
806             foreach ($this-> methodsbyfile[$file][$class] as $i => $vr) {
807                 $vr-> docblock-> addKeyword('access', 'private');
808                 $this-> methodsbyfile[$file][$class][$i] = $vr;
809             }
810         }
811         if (isset($this-> constsbyfile[$file])
812             && isset($this-> constsbyfile[$file][$class])) {
813             foreach ($this-> constsbyfile[$file][$class] as $i => $vr) {
814                 $vr-> docblock-> addKeyword('access', 'private');
815                 $this-> constsbyfile[$file][$class][$i] = $vr;
816             }
817         }
818     }
819 }
820 $this-> classpathpackages[$class][$file]
821     = array($db-> docblock-> package, $db-> docblock-> subpackage);
822 if ($db-> docblock-> getExplicitPackage()) {
823     $render-> proceduralpages->
824         addClassPackageToFile($file,
825             $db-> docblock-> package, $db-> docblock-> subpackage);
826 }
827 $render-> addElementToPage($db, $file);
828 if (isset($this-> varsbyfile[$file])
829     && isset($this-> varsbyfile[$file][$class])) {
830     foreach ($this-> varsbyfile[$file][$class] as $i => $vr) {
831         $vr-> docblock-> package = $db-> docblock-> package;
832         $vr-> docblock-> subpackage = $db-> docblock-> subpackage;
833         $render-> addElementToPage($vr, $file);
834         $render-> addUses($vr, $file);
835         $this-> varsbyfile[$file][$class][$i] = $vr;
836         $this-> vars[$db-> docblock-> package][$class][$vr-> getName()] = $vr;
837     }
838 }
839 if (isset($this-> methodsbyfile[$file])
840     && isset($this-> methodsbyfile[$file][$class])) {
841     foreach ($this-> methodsbyfile[$file][$class] as $i => $vr) {
842         $vr-> docblock-> package = $db-> docblock-> package;
843         $vr-> docblock-> subpackage = $db-> docblock-> subpackage;
844         $render-> addElementToPage($vr, $file);
845         $render-> addUses($vr, $file);
846         $this-> methodsbyfile[$file][$class][$i] = $vr;
847         $this-> methods[$db-> docblock-> package][$class][$vr-> getName()] =
$vr;
848     }
849 }
850 if (isset($this-> constsbyfile[$file])
851     && isset($this-> constsbyfile[$file][$class])) {
852     foreach ($this-> constsbyfile[$file][$class] as $i => $vr) {
853         $vr-> docblock-> package = $db-> docblock-> package;
854         $vr-> docblock-> subpackage = $db-> docblock-> subpackage;
855         $render-> addElementToPage($vr, $file);
856         $render-> addUses($vr, $file);
857         $this-> constsbyfile[$file][$class][$i] = $vr;
858         $this-> methods[$db-> docblock-> package][$class][$vr-> getName()] =
$vr;
859     }
860 }
861 $this-> classpackages[$class][] =
array($db-> docblock-> package, $db-> docblock-> subpackage);
862 if (is_array($db-> parent))
863     $this-> classparents[$db-> docblock-> package][$class] = $db-> parent[1];

```

```

865     else
866         $this-> classparents[$db-> docblock-> package][$class] = $db-> getExtends();
867     if (is_array($db-> parent)) {
868         $z = $this-> getClass($db-> parent[1], $db-> parent[0]);
869     }
870     $this-> classchildren[$z-> docblock-> package][$db-> parent[1]][] = $db;
871 }
872 if (isset($this-> classchildrenbyfile[$class])) {
873     foreach ($this-> classchildrenbyfile[$class] as $childfile => $other) {
874         // test 1, inherits in same file (must be same package)
875         if ($childfile == $file) {
876             foreach ($other as $child => $packages) {
877                 // debug("parent $class same file $child");
878                 $this-> processChild($render, $child, $childfile);
879                 $x = $this-> getClass($child, $childfile);
880                 if ($x-> docblock-> package
881                     != $GLOBALS['phpDocumentor_DefaultPackageName']) {
882                     // child package need root for class trees
883                     if ($x-> docblock-> package != $db-> docblock-> package) {
884                         // debug("adding $child in $childfile 1");
885                         $this-> roots[$child][] = $childfile;
886                     }
887                 }
888             }
889         } else {
890             // test 2, different file, same package
891             foreach ($other as $child => $packages) {
892                 for ($j=0; $j< count($packages); $j++) {
893                     if ($this-> classesbyfile[$file][$class]->
894                         docblock-> package == $packages[$j]) {
895                         $this-> processChild($render, $child, $childfile);
896                         // debug("$childfile diff file $child, parent $class,
897                         // same package ".$packages[$j]);
898                     } else {
899                         /*
900                         * test 3, different file, different package,
901                         * only 1 parent is possible
902                         */
903                         if (isset($this-> classesbynamefile[$child])) {
904                             // 1 possible parent
905                             if (count($this-> classesbynamefile[$class])
906                                 == 1) {
907                                 // debug("$childfile diff file $child,
908                                 // diff package,
909                                 // 1 possible parent root $class");
910                                 $this-> processChild($render,
911                                     $child, $childfile);
912                                 $x = $this-> getClass($child, $childfile);
913                                 if ($x-> docblock-> package
914                                     != $GLOBALS
915                                     ['phpDocumentor_DefaultPackageName']) {
916                                     // child package need root
917                                     //for class trees
918                                     if ($x-> docblock-> package
919                                         != $db-> docblock-> package) {
920                                         // debug("adding roots
921                                         // $child in $childfile 2");
922                                         $this-> roots[$child][] = $childfile;
923                                     }
924                                 }
925                             }
926                         }
927                     }
928                 }
929             }
930         }
931     }
932 }
933 /**
934  * Get the parserClass representation of a class from its name and file
935  *
936  * @param string $class classname
937  * @param string $file file classname is located in
938  *
939  * @return parserClass
940  */
941 function & getClass($class, $file)
942 {

```

```

945     // debug("getClass called with class $class file $file");
946     return $this-> classesbyfile[$file][$class];
947 }
948 /**
949 * Used by {@link parserData::getClasses()}
950 * to retrieve classes defined in file $path
951 *
952 * retrieves the array entry from {@link $classesbyfile} for $path
953 *
954 * @param string $path full path to filename
955 *
956 * @return mixed returns false if no classes defined in the file,
957 *         otherwise returns an array of {@link parserClass}es
958 */
959
960 function getClassesInPath($path)
961 {
962     if (!isset($this-> classesbyfile[$path])) return false;
963     return $this-> classesbyfile[$path];
964 }
965
966 /**
967 * called by {@link parserClass::hasMethods()}. Should not be directly called
968 *
969 * @param string $file file classname is located in
970 * @param string $class classname
971 *
972 * @return bool
973 * @access private
974 */
975 function hasMethods($file, $class)
976 {
977     return isset($this-> methodsbyfile[$file][$class]);
978 }
979
980 /**
981 * called by {@link parserClass::hasConsts()}.
982 * Should not be directly called
983 *
984 * @param string $file file classname is located in
985 * @param string $class classname
986 *
987 * @return bool
988 * @access private
989 */
990 function hasConsts($file, $class)
991 {
992     return isset($this-> constsbyfile[$file][$class]);
993 }
994
995 /**
996 * called by {@link parserClass::hasVars()}. Should not be directly called
997 *
998 * @param string $file file classname is located in
999 * @param string $class classname
1000 *
1001 * @return bool
1002 * @access private
1003 */
1004 function hasVars($file, $class)
1005 {
1006     return isset($this-> varsbyfile[$file][$class]);
1007 }
1008
1009 /**
1010 * called by {@link parserClass::hasMethod()}. Should not be directly called
1011 *
1012 * @param string $class classname
1013 * @param string $file file classname is located in
1014 * @param string $name method name
1015 *
1016 * @return bool
1017 * @access private
1018 */
1019 function hasMethod($class, $file, $name)
1020 {
1021     if (!$this-> hasMethods($file, $class)) return false;
1022     for ($i=0; $i< count($this-> methodsbyfile[$file][$class]); $i++) {
1023         if ($this-> methodsbyfile[$file][$class][$i]-> getName() == $name)
1024             return true;

```

```

1025         }
1026     return false;
1027 }
1028 /**
1029 * called by {@link parserClass::hasVar()}. Should not be directly called
1030 *
1031 * @param string $class classname
1032 * @param string $file file classname is located in
1033 * @param string $name var name
1034 *
1035 * @return bool
1036 * @access private
1037 */
1038 function hasVar($class, $file, $name)
1039 {
1040     if (!$this-> hasVars($file, $class)) return false;
1041     for ($i=0; $i< count($this-> varsbyfile[$file][$class]); $i++) {
1042         if ($this-> varsbyfile[$file][$class][$i]-> getName() == $name)
1043             return true;
1044     }
1045     return false;
1046 }
1047 /**
1048 * called by {@link parserClass::hasConst()}. Should not be directly called
1049 *
1050 * @param string $class classname
1051 * @param string $file file classname is located in
1052 * @param string $name constant name
1053 *
1054 * @return bool
1055 * @access private
1056 */
1057 function hasConst($class, $file, $name)
1058 {
1059     if (!$this-> hasConsts($file, $class)) return false;
1060     for ($i=0; $i< count($this-> constsbyfile[$file][$class]); $i++) {
1061         if ($this-> constsbyfile[$file][$class][$i]-> getName() == $name)
1062             return true;
1063     }
1064     return false;
1065 }
1066 /**
1067 * called by {@link parserClass::getMethods()}. Should not be directly called
1068 *
1069 * @param string $class classname
1070 * @param string $file file classname is located in
1071 *
1072 * @return mixed
1073 * @access private
1074 */
1075 function & getMethods($class, $file)
1076 {
1077     if (!isset($this-> methodsbyfile[$file][$class])) {
1078         $flag = false;
1079         return $flag;
1080     }
1081     return $this-> methodsbyfile[$file][$class];
1082 }
1083 /**
1084 * called by {@link parserClass::getVars()}. Should not be directly called
1085 *
1086 * @param string $class classname
1087 * @param string $file file classname is located in
1088 *
1089 * @return mixed
1090 * @access private
1091 */
1092 function & getVars($class, $file)
1093 {
1094     if (!isset($this-> varsbyfile[$file][$class])) {
1095         $flag = false;
1096         return $flag;
1097     }
1098     return $this-> varsbyfile[$file][$class];
1099 }
1100
1101
1102
1103
1104

```

```

1105 /**
1106 * called by {@link parserClass::getConsts()}. Should not be directly called
1107 *
1108 * @param string $class classname
1109 * @param string $file file classname is located in
1110 *
1111 * @return mixed
1112 * @access private
1113 */
1114 function & getConsts($class, $file)
1115 {
1116     if (!isset($this-> constsbyfile[$file][$class])) {
1117         $flag = false;
1118         return $flag;
1119     }
1120     return $this-> constsbyfile[$file][$class];
1121 }
1122
1123 /**
1124 * called by {@link parserClass::getMethod()}. Should not be directly called
1125 *
1126 * @param string $class classname
1127 * @param string $file file classname is located in
1128 * @param string $name method name
1129 *
1130 * @return mixed
1131 * @access private
1132 */
1133 function getMethod($class, $file, $name)
1134 {
1135     if (!$this-> hasMethod($class, $file, $name)) return false;
1136     for ($i=0; $i< count($this-> methodsbyfile[$file][$class]); $i++) {
1137         if ($this-> methodsbyfile[$file][$class][$i]-> getName() == $name)
1138             return $this-> methodsbyfile[$file][$class][$i];
1139     }
1140 }
1141
1142 /**
1143 * called by {@link parserClass::getVar()}. Should not be directly called
1144 *
1145 * @param string $class classname
1146 * @param string $file file classname is located in
1147 * @param string $name var name
1148 *
1149 * @return mixed
1150 * @access private
1151 */
1152 function getVar($class, $file, $name)
1153 {
1154     if (!$this-> hasVar($class, $file, $name)) return false;
1155     for ($i=0; $i< count($this-> varsbyfile[$file][$class]); $i++) {
1156         if ($this-> varsbyfile[$file][$class][$i]-> getName() == $name)
1157             return $this-> varsbyfile[$file][$class][$i];
1158     }
1159 }
1160
1161 /**
1162 * called by {@link parserClass::getConst()}. Should not be directly called
1163 *
1164 * @param string $class classname
1165 * @param string $file file classname is located in
1166 * @param string $name const name
1167 *
1168 * @return mixed
1169 * @access private
1170 */
1171 function getConst($class, $file, $name)
1172 {
1173     if (!$this-> hasConst($class, $file, $name)) return false;
1174     for ($i=0; $i< count($this-> constsbyfile[$file][$class]); $i++) {
1175         if ($this-> constsbyfile[$file][$class][$i]-> getName() == $name)
1176             return $this-> constsbyfile[$file][$class][$i];
1177     }
1178 }
1179
1180 /**
1181 * Search for a class in a package
1182 *
1183 * @param string $class classname
1184 * @param string $package package classname is in

```

```

1185 *
1186 * @return mixed returns false if no class in $package,
1187 * otherwise returns a {@link parserClass}
1188 */
1189 function & getClassByPackage($class, $package)
1190 {
1191     if (!isset($this-> classesbynamefile[$class])) {
1192         // removed, too many warnings, not very useful
1193         // addWarning(PDERROR_CLASS_NOT_IN_PACKAGE,$class,$package);
1194
1195         $flag = false;
1196         return $flag;
1197     }
1198     for ($i=0; $i < count($this-> classesbynamefile[$class]); $i++) {
1199         $cls =
1200             $this-> classesbynamefile[$this-> classesbynamefile[$class][$i]][$class];
1201         $pkg = $cls-> getPackage();
1202         if ($pkg == $package)
1203             return $cls;
1204     }
1205     // addWarning(PDERROR_CLASS_NOT_IN_PACKAGE,$class,$package);
1206
1207     $flag = false;
1208     return $flag;
1209 }
1210 /**
1211 * Find the parent class of a class in file $file
1212 * uses 3 tests to find the parent classname:
1213 * <ol>
1214 *   <li>only one class with the parent classname</li>
1215 *   <li>more than one class, but only one in the same file as the child</li>
1216 *   <li>only one parent class in the same package as the child</li>
1217 * </ol>
1218 *
1219 * @param string $class classname
1220 * @param string $file file classname is located in
1221 *
1222 * @return mixed false if no parent class,
1223 *             a string if no parent class found by that name,
1224 *             and an array(file parentclass is in, parentclassname)
1225 */
1226 function getParentClass($class,$file)
1227 {
1228     if (!isset($this-> classesbyfile[$file][$class])) {
1229         return false;
1230     }
1231     $element = $this-> classesbyfile[$file][$class];
1232     if (!$ex = $element-> getExtends()) return false;
1233     // first check to see if there is one and only one
1234     // class with the parent class's name
1235     if (isset($this-> classesbynamefile[$ex])) {
1236         if (count($this-> classesbynamefile[$ex]) == 1) {
1237             if ($this-> classesbyfile
1238                 [$this-> classesbynamefile[$ex][0]][$ex]-> ignore) {
1239                 return $ex;
1240             }
1241             return array($this-> classesbynamefile[$ex][0],$ex);
1242         } else {
1243             // next check to see if there is a parent class in the same file
1244             if (isset($this-> classesbyfile[$file][$ex])) {
1245                 if ($this-> classesbyfile[$file][$ex]-> ignore) {
1246                     return $ex;
1247                 }
1248                 return array($file,$ex);
1249             }
1250             // next check to see if there is only one package
1251             // used in the file, try to resolve it that way
1252             if (isset($this-> classpackagebyfile[$file])) {
1253                 if (count($this-> classpackagebyfile[$file]) == 1) {
1254                     for ($i=0;$i< count($this-> classesbynamefile[$ex]);$i++) {
1255                         if ($this-> classesbyfile
1256                             [$this-> classesbynamefile[$ex][$i]][$ex]-> getPackage()
1257                             == $this-> classpackagebyfile[$file][0]) {
1258                             if ($this-> classesbyfile
1259                                 [$this-> classesbynamefile[$ex][$i]][$ex]-> ignore)
1260                                 return $ex;
1261                             return array($this-> classesbynamefile[$ex][$i],$ex);
1262                         }
1263                     }
1264                 }
1265             }
1266         }
1267     }
1268 }

```

```

1265             }
1266         }
1267         // name conflict
1268         addWarning(PDERROR_INHERITANCE_CONFLICT, $class, $file, $ex);
1269         return $ex;
1270     }
1271 } else {
1272     if (class_exists('ReflectionClass') && class_exists($ex)) {
1273         $r = new ReflectionClass($ex);
1274         if ($r->isInternal()) {
1275             return $ex; // no warning
1276         }
1277     }
1278     addWarning(PDERROR_PARENT_NOT_FOUND, $class, $ex);
1279     return $ex;
1280 }
1281
1282 /**
1283 * Get a list of all root classes indexed by package. Used to generate
1284 * class trees by {@link Converter}
1285 *
1286 * @param boolean $all [since phpDocumentor 1.3.0RC6] determines whether to
1287 *                      return class trees that extend non-parsed classes
1288 *
1289 * @return array array(package => array(rootclassname, rootclassname,...),...)
1290 */
1291 function getRoots($all = false)
1292 {
1293     $roots      = array();
1294     $temproots = $this->roots;
1295     if (!$all) {
1296         foreach ($this->specialRoots as $package => $root) {
1297             foreach ($root as $parent => $info) {
1298                 $temproots[$info[0]][] = $info[1];
1299             }
1300         }
1301     }
1302     foreach ($temproots as $class => $files) {
1303         if (count($files)) {
1304             foreach ($files as $i => $boofou) {
1305                 $x = $this->getClass($class, $files[$i]);
1306
1307                 $roots[$x->getPackage()][] = $class;
1308             }
1309         }
1310     }
1311     foreach ($roots as $package => $root) {
1312         usort($roots[$package], "strnatcasecmp");
1313     }
1314     if ($all) {
1315         $specialRoots = array();
1316         foreach ($this->specialRoots as $parent => $classinfo) {
1317             if (count($classinfo)) {
1318                 foreach ($classinfo as $i => $info) {
1319                     $x = $this->getClass($info[0], $info[1]);
1320
1321                     $specialRoots[$x->getPackage()][$parent][] = $info[0];
1322                 }
1323             }
1324         }
1325         foreach ($specialRoots as $package => $root) {
1326             uksort($specialRoots[$package], "strnatcasecmp");
1327             foreach ($specialRoots[$package] as $parent => $classes) {
1328                 usort($specialRoots[$package][$parent], 'strnatcasecmp');
1329             }
1330         }
1331     }
1332     return array('special' => $specialRoots, 'normal' => $roots);
1333 }
1334
1335     return $roots;
1336 }
1337 /**
1338 * Get all classes confirmed in parsing
1339 * to be descended class $parclass in file $file
1340 *
1341 * @param string $parclass name of parent class
1342 * @param string $file    file parent class is found in
1343 *
1344 * @return mixed either false if no children, or array of format

```

```
1345 *      array(childname => childfile,childname2 => childfile2,...)
1346 * @see parserClass::getChildClassList()
1347 * @uses $definitechild
1348 */
1349 function getDefiniteChildren($parclass, $file)
1350 {
1351     if (isset($this-> definitechild[$parclass][$file]))
1352         return $this-> definitechild[$parclass][$file];
1353     return false;
1354 }
1355 ?
1356 ?>
```

# File Source for clone.inc.php

Documentation for this file is available at [clone.inc.php](#)

```
1  <?php
2  /**
3  * Object clone method
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2001-2006 Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @category ToolsAndUtilities
29 * @package phpDocumentor
30 * @author Greg Beaver <cellog@php.net>
31 * @copyright 2001-2006 Gregory Beaver
32 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33 * @version CVS: $Id: clone.inc.php 243202 2007-09-30 02:08:08Z ashnazg $
34 * @filesource
35 * @link http://www.phpdoc.org
36 * @link http://pear.php.net/PhpDocumentor
37 * @since 1.0rc1
38 * @todo CS cleanup - change package to PhpDocumentor
39 */
40 /**
41 * Clone an object in PHP 4
42 *
43 * @param object $obj the object to be cloned
44 *
45 * @return object the new clone
46 * @todo CS cleanup - rename function to PhpDocumentor_clone
47 */
48 function phpDocumentor_clone($obj)
49 {
50     return $obj;
51 }
52
53 ?>
```

# File Source for clone5.inc.php

Documentation for this file is available at [clone5.inc.php](#)

```
1  <?php
2  /**
3  * Object clone method
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2001-2006 Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @category ToolsAndUtilities
29 * @package phpDocumentor
30 * @author Greg Beaver <cellog@php.net>
31 * @copyright 2001-2006 Gregory Beaver
32 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33 * @version CVS: $Id: clone5.inc.php 243202 2007-09-30 02:08:08Z ashnazg $
34 * @filesource
35 * @link http://www.phpdoc.org
36 * @link http://pear.php.net/PhpDocumentor
37 * @since 1.0rc1
38 * @todo CS cleanup - change package to PhpDocumentor
39 */
40 /**
41 * Clone an object in PHP 5
42 *
43 * @param object $obj the object to be cloned
44 *
45 * @return object the new clone
46 * @ignore
47 * @todo CS cleanup - rename function to PhpDocumentor_clone
48 */
49 function phpDocumentor_clone($obj)
50 {
51     return clone $obj;
52 }
53
54 ?>
```

# File Source for common.inc.php

Documentation for this file is available at [common.inc.php](#)

```
1  <?php
2  /**
3   * Common information needed by all portions of the application
4   *
5   * phpDocumentor :: automatic documentation generator
6   *
7   * PHP versions 4 and 5
8   *
9   * Copyright (c) 2001-2008 Gregory Beaver
10  *
11  * LICENSE:
12  *
13  * This library is free software; you can redistribute it
14  * and/or modify it under the terms of the GNU Lesser General
15  * Public License as published by the Free Software Foundation;
16  * either version 2.1 of the License, or (at your option) any
17  * later version.
18  *
19  * This library is distributed in the hope that it will be useful,
20  * but WITHOUT ANY WARRANTY; without even the implied warranty of
21  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22  * Lesser General Public License for more details.
23  *
24  * You should have received a copy of the GNU Lesser General Public
25  * License along with this library; if not, write to the Free Software
26  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27  *
28  * @category ToolsAndUtilities
29  * @package phpDocumentor
30  * @author Greg Beaver <cellog@php.net>
31  * @copyright 2001-2008 Gregory Beaver
32  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33  * @version CVS: $Id: common.inc.php 288074 2009-09-05 02:16:26Z ashnazg $
```

\* @filesource

```
34  * @link http://www.phpdoc.org
35  * @link http://pear.php.net/PhpDocumentor
36  * @see parserDocBlock, parserInclude, parserPage, parserClass
37  * @see parserDefine, parserFunction, parserMethod, parserVar
38  * @since 1.0rc1
39  * @todo CS cleanup - change package to PhpDocumentor
40  * @todo CS cleanup - rename constant to TOKENIZER_EXT
41  */
42
43
44  /* phpDocumentor version */
45  if ('@PEAR-DIR@' != '@'. 'PEAR-DIR@') {
46      /** @ignore */
47      define("PHPDOCUMENTOR_VER" , "@VER@");
48  } else {
49      define("PHPDOCUMENTOR_VER" , "1.4.3");
50  }
51
52  /* phpDocumentor URL */
53  define("PHPDOCUMENTOR_WEBSITE" , "http://www.phpdoc.org");
54
55  // set the correct path delimiter
56  define('SMART_PATH_DELIMITER', DIRECTORY_SEPARATOR);
57
58  define('tokenizer_ext', extension_loaded('tokenizer')
59      && version_compare(phpversion(), "4.3.0" , ">="));
60
61  // we just replace all the \ with / so that we can just operate on /
62  define('PATH_DELIMITER', '/'); // set the correct path delimiter
63
64  define('PHPDOCUMENTOR_WINDOWS', substr(PHP_OS, 0, 3) == 'WIN');
65
66  define('_IN_PHP5',
67      phpversion() == '5.0.0RC1-dev' || phpversion() == '5.0.0RC2-dev'
```

```

68     || version_compare/phpversion(), '5.0.0', 'ge'));
69
70 // determine which "clone" class to set, based on PHP major version
71 $cloneClassDir = 'PhpDocumentor' . DIRECTORY_SEPARATOR . 'phpDocumentor';
72 $cloneClassFile = 'clone.inc.php';
73 if ('@VER@' == '@'. 'VER@') {
74     // we're _not_ in a PEAR installation
75     $cloneClassDir = dirname(__FILE__);
76 }
77 if (_IN_PHP5) {
78     // we _are_ in PHP5
79     $cloneClassFile = 'clone5.inc.php';
80 }
81 require_once $cloneClassDir . DIRECTORY_SEPARATOR . $cloneClassFile;
82
83 // make arg arrays available
84 if (isset($_SERVER['argv'])) {
85     $argv = $_SERVER['argv'];
86     $argc = $_SERVER['argc'];
87 }
88
89 /**
90 * used in phpdoc.php and new_phpdoc.php
91 *
92 * @param string $directory a directory string
93 *
94 * @return array an array of directory contents
95 * @todo CS cleanup - rename function to PhpDocumentor_ConfigFileList
96 */
97 function phpDocumentor_ConfigFileList($directory)
98 {
99     $ret = array();
100    if (@is_dir($directory)) {
101        $ret = array();
102
103        // thanks to Jason E Sweat (jsweat@users.sourceforge.net) for fix
104        $d = @dir($directory);
105
106        while ($d && $entry=$d-> read()) {
107            $getentry = false;
108            if (strcmp($entry, ".") != 0 && strcmp($entry, "..") != 0) {
109                if (substr($entry, 0, 1) != ".") $getentry = true;
110            }
111            if ($getentry == true) {
112                if (strpos($entry, '.ini'))
113                    if (is_file($directory . PATH_DELIMITER . $entry)) {
114                        $ret[] = str_replace('.ini', '', $entry);
115                    }
116            }
117            if ($d) $d-> close();
118        } else {
119        }
120    }
121    return $ret;
122 }
123
124
125 /**
126 * Parse an .ini file
127 *
128 * Works like {@link parse_ini_file}, except it will take a section like:
129 *
130 * <pre>
131 * [MYVAR]
132 * value1
133 * value2
134 * value3
135 * </pre>
136 *
137 * and return an associative array(MYVAR => array(value1, value2, value3))
138 *
139 * @param string $filename full path to the ini file
140 * @param bool   $process_sections add an associative index
141 *                                for each section [in brackets]
142 *
143 * @return array
144 * @todo CS cleanup - rename function to PhpDocumentor_parse_ini_file
145 */
146 function phpDocumentor_parse_ini_file($filename, $process_sections = false)

```

```

147  {
148      $ini_array = array();
149      $sec_name = "";
150      $lines = @file($filename);
151      if (!$lines) return $lines;
152      foreach ($lines as $line) {
153          // code by Greg Beaver, ignore comments
154          if ($line[0] == ';') continue;
155          $line = trim($line);
156
157          if ($line == "") continue;
158
159          if ($line[0] == "[" && $line[strlen($line) - 1] == "]") {
160              $sec_name = substr($line, 1, strlen($line) - 2);
161          } else {
162              if (strpos($line, "=")) {
163                  $pos = strpos($line, "=");
164                  $property = trim(substr($line, 0, $pos));
165                  // code by Greg Beaver
166                  if (substr($property, 0, 1) == " " && substr($property, -1) ==
167                      " ") {
168                      $property =
169                          stripslashes(substr($property, 1, count($property) - 2));
170
171                  $value = trim(substr($line, $pos + 1));
172                  if ($value == 'false') $value = false;
173                  if ($value == 'true') $value = true;
174                  if (substr($value, 0, 1) == " " && substr($value, -1) == " ")
175                      $value = stripslashes(substr($value, 1, count($value) - 2));
176
177                  // done additions
178
179                  if ($process_sections) {
180                      if ($sec_name != '') {
181                          $ini_array[$sec_name][$property] = $value;
182                      } else {
183                          $ini_array[$property] = $value;
184                      }
185                  } else {
186                      // code by Greg Beaver
187                      if (trim($line[0]) == ';') continue;
188                      if ($process_sections) {
189                          $ini_array[$sec_name][] = trim($line);
190                      }
191                      // done additions
192
193                  }
194              }
195          }
196      }
197      return $ini_array;
198  }
199
200 /**
201 * construct an "array_key_exists()" method
202 * if the runtime PHP version doesn't have one
203 *
204 * @todo CS Cleanup - can't avoid "prefixed by package" error
205 * @todo depend on PHP_Compat for this?
206 */
207 if (!function_exists('array_key_exists')) {
208     /**
209      * Determines if a given key exists in a given array
210      *
211      * @param mixed $key    key to search for
212      * @param array $search the array of keys to search
213      *
214      * @return bool whether or not the key was found
215      * @ignore
216      */
217     function array_key_exists($key, $search)
218     {
219         foreach ($search as $keys => $nul) {
220             if ($key == $keys) return true;
221         }
222         return false;
223     }
224 }

```

```

225     }
226
227 /**
228 * construct an "is_a()" method
229 * if the runtime PHP version doesn't have one
230 *
231 * @todo CS Cleanup - can't avoid "prefixed by package" error
232 * @todo depend on PHP_Compat for this?
233 */
234 if (!function_exists('is_a')) {
235     /**
236      * Determines if one item "is" an object of the other item
237      *
238      * @param string $classname the class in question
239      * @param string $classquery the "is it a" class
240      *
241      * @return bool whether or not the class "is" one
242      * @ignore
243     */
244     function is_a($classname, $classquery)
245     {
246         $father = get_parent_class($classname);
247         if (strtolower($father) == strtolower($classquery)) {
248             return true;
249         } elseif (!empty($father)) {
250             return is_a($father, $classquery);
251         } else {
252             return false;
253         }
254     }
255 }
256
257 /**
258 * Debugging output
259 *
260 * @param string $s the "debug message" string to echo out
261 *
262 * @return void
263 * @todo CS Cleanup - can't avoid "prefixed by package" error
264 */
265 function debug($s)
266 {
267     echo "      $s\n"      ;
268 }
269
270 /**
271 * Returns a formatted var_dump for debugging purposes.
272 *
273 * @param string $s string to display
274 * @param mixed $v unlimited number of variables to display with var_dump()
275 *
276 * @return void
277 */
278 function fancy_debug($s,$v)
279 {
280     if (isset($GLOBALS['dont_debug']) && $GLOBALS['dont_debug']) return;
281     debug($s."\n</pre><blockquote><pre>" );
282     var_dump($v);
283     if (func_num_args() > 2) {
284         for ($i=2;$i< func_num_args();$i++) {
285             $a = func_get_arg($i);
286             // debug(" ");
287             var_dump($a);
288         }
289     }
290     debug("</pre></blockquote><pre>\n\n");
291 }
292
293 /**
294 * Returns a lower-cased version of get_class for PHP 5
295 *
296 * get_class() returns case as declared in the file in PHP 5
297 *
298 * @param object $object the object to get the classname for
299 *
300 * @return string the class name of the given object
301 * @todo CS cleanup - rename function to PhpDocumentor_get_class
302 */
303 function phpDocumentor_get_class($object)

```

```
305  {
306      if (is_object($object)) {
307          return strtolower(get_class($object));
308      }
309      return false;
310  }
311
312 ?>
```

# File Source for EventStack.inc

Documentation for this file is available at [EventStack.inc](#)

```
1  <?php
2  /**
3  * An Event Stack for inter-program communication, particularly for parsing
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2000-2007 Joshua Eichorn
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @category ToolsAndUtilities
29 * @package phpDocumentor
30 * @author Joshua Eichorn <jeichorn@phpdoc.org>
31 * @copyright 2000-2007 Joshua Eichorn
32 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
33 * @version CVS: $Id: EventStack.inc 243937 2007-10-10 02:27:42Z ashnazg $
34 * @filesource
35 * @link http://www.phpdoc.org
36 * @link http://pear.php.net/PhpDocumentor
37 * @since 0.1
38 * @todo CS cleanup - change package to PhpDocumentor
39 */
40 /**
41 * An event Stack
42 *
43 * @category ToolsAndUtilities
44 * @package phpDocumentor
45 * @author Joshua Eichorn <jeichorn@phpdoc.org>
46 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
47 * @version Release: @VER@
48 * @link http://www.phpdoc.org
49 * @link http://pear.php.net/PhpDocumentor
50 * @todo CS cleanup - change package to PhpDocumentor
51 */
52 class EventStack
53 {
54     /**
55      * The stack
56      * @var array
57      */
58     var $stack = array(PARSER_EVENT_NOEVENTS);
59     /**
60      * The number of events in the stack
61      * @var integer
62      */
63     var $num = 0;
64     /**
65      * Push an event onto the stack
66
67 
```

```
68
69      *
70      * @param int $event All events must be constants
71      *
72      * @return void
73      */
74      function pushEvent($event)
75      {
76          $this-> num = array_push($this-> stack, $event) - 1;
77      }
78
79      /**
80      * Pop an event from the stack
81      *
82      * @return int An event
83      */
84      function popEvent()
85      {
86          $this-> num--;
87          return array_pop($this-> stack);
88      }
89
90      /**
91      * Get the current event
92      *
93      * @return int An event
94      */
95      function getEvent()
96      {
97          return $this-> stack[$this-> num];
98      }
}
```

# File Source for IntermediateParser.inc

Documentation for this file is available at [IntermediateParser.inc](#)

```
1  <?php
2  /**
3  * The phpDocumentor_IntermediateParser Class
4  *
5  * The Intermediary Data Parser (intermediate between Parse and Converter)
6  *
7  * phpDocumentor :: automatic documentation generator
8  *
9  * PHP versions 4 and 5
10 *
11 * Copyright (c) 2002-2006 Gregory Beaver
12 *
13 * LICENSE:
14 *
15 * This library is free software; you can redistribute it
16 * and/or modify it under the terms of the GNU Lesser General
17 * Public License as published by the Free Software Foundation;
18 * either version 2.1 of the License, or (at your option) any
19 * later version.
20 *
21 * This library is distributed in the hope that it will be useful,
22 * but WITHOUT ANY WARRANTY; without even the implied warranty of
23 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
24 * Lesser General Public License for more details.
25 *
26 * You should have received a copy of the GNU Lesser General Public
27 * License along with this library; if not, write to the Free Software
28 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
29 *
30 * @package    phpDocumentor
31 * @author     Gregory Beaver <cellog@php.net>
32 * @copyright  2002-2006 Gregory Beaver
33 * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
34 * @version    CVS: $Id: IntermediateParser.inc 247821 2007-12-09 06:11:35Z ashnazg $
35 * @filesource
36 * @link       http://www.phpdoc.org
37 * @link       http://pear.php.net/PhpDocumentor
38 * @since      1.1
39 */
40 /**
41 * The phpDocumentor_IntermediateParser Class
42 *
43 * This class performs the work of organizing raw data from the parser in the
44 * format of descendants of the {@link parserElement} class. This is also where
45 * processing of package pages occurs, in
46 * {@link phpDocumentor_IntermediateParser::handleClass()} for class-level
47 * packages and {@link phpDocumentor_IntermediateParser::handleDocBlock()} for
48 * page-level packages.
49 *
50 * Most of the work of this parser goes to matching up
51 * DocBlocks with the elements that they are documenting. Since DocBlocks are
52 * passed before the element they document, the last DocBlock is stored in
53 * {@link phpDocumentor_IntermediateParser::$last} and then placed into the
54 * $docblock parameter of the parserElement
55 * descendant object.
56 * @author Gregory Beaver
57 * @version $Id: IntermediateParser.inc 247821 2007-12-09 06:11:35Z ashnazg $
58 * @copyright 2002 Gregory Beaver
59 * @package    phpDocumentor
60 */
61 class phpDocumentor_IntermediateParser
62 {
63     /**
64     * @var parserDocBlock
65     */
66     var $last;
67     /**
```

```

68 * type of the last parser Element handled
69 *
70 * This is used in handleDocBlock to determine whether a DocBlock is a
71 * page-level DocBlock in conjunction with the {@link parserData::$clean}
72 * var. A page-level DocBlock is always the first DocBlock in a file, and
73 * must be followed by another DocBlock. The first test is handled by
74 * parserData::$clean, which is set to false on the first encounter of an
75 * element, and the second test is handled by this variable, which must be
76 * equal to "docblock"
77 * @see handleDocBlock()
78 * @var string
79 */
80 var $lasttype = '';
81
82 /**
83 * Name of the class currently being parsed.
84 * It is only used (and only valid) when phpDocumentor_IntermediateParser is
85 * parsing a class
86 * @var string
87 */
88 var $cur_class = '';
89
90 /**
91 * type of the current parser Element being handled
92 *
93 * This is used by {@link HandleEvent()} to set the {@link $lasttype} var,
94 * which is used to detect page-level DocBlocks
95 * @var string
96 */
97 var $type = '';
98
99 /**
100 * set in {@link Setup.inc.php} to the value of the parseprivate commandline
101 * option. If this option is true, elements with an @access private tag
102 * will be parsed and displayed
103 * @tutorial phpDocumentor.howto.pkg#using.command-line.parseprivate
104 * @var boolean
105 */
106 var $parsePrivate = false;
107
108 /**
109 * this variable is used to prevent parsing of elements with an @ignore tag
110 * @see $packageoutput
111 * @see $parsePrivate
112 */
113 var $private_class = false;
114
115 /**
116 * used to set the output directory
117 * @see setTargetDir()
118 */
119 var $targetDir;
120
121 /**
122 * used to set the template base directory
123 * @see setTemplateBase()
124 */
125 var $templateBase;
126
127 /**
128 * array of parsed package pages
129 *
130 * used by {@link Convert()} to convert all package pages into output
131 * @var array
132 */
133 var $package_pages = array();
134
135 /**
136 * @var array array of all {@link parserData} containing page information
137 */
138 var $pages = array();
139
140 * Put away a page that has been @ignored or @access private if
141 * !{@link $parsePrivate}
142 *
143 * When a page has @access private in its DocBlock, it is placed here
144 * instead of in {@link $pages}, to allow for proper Class parsing. Since
145 * classes and pages are parsed as if they were separate, this array allows
146 * public classes on private pages to retrieve information needed about the
147 * page that holds the class and to {@link addPageIfNecessary()} to the

```

```

148 * $pages array
149 * @var array
150 */
151 var $privatepages = array();
152 /**
153 * Keeps track of packages of classes that have parent classes in another
154 * package. Used in automatic linking.
155 *
156 * This array is updated by {@link addPackageParent()}, which is called in
157 * {@link Classes::processChild()} to keep track of classes that descend
158 * from classes in different packages. In other words, if class foo is in
159 * package one, and class bar is in package two, an entry
160 * $package_parents['two'] = 'one' will be made.
161 * @var array Format: packagename => parentpackagename
162 * @see Converter::getLink()
163 */
164 var $package_parents = array();
165
166 /**
167 * Used to determine the category for tutorials.
168 *
169 * <b>WARNING:</b> If more than one category exists, the last category
170 * encountered will overwrite the previous and will raise a big warning
171 * @var array Format: packagename => categoryname
172 */
173 var $packagecategories = array();
174
175 /**
176 * list of all packages encountered while documenting. Used in automatic
177 * linking.
178 *
179 * Converter::getLink() first checks if an ambiguous link is found in the
180 * current package. If not, it then checks in parent packages, and if still
181 * not found, uses this array to check in the rest of the packages before
182 * giving up
183 * @var array Format: array(packagename => 1, packagename => 1,...)
184 * @see Converter::getLink()
185 */
186 var $all_packages = array();
187
188 /**
189 * array of packages to parser and output documentation for, if not all
190 * packages should be documented
191 *
192 * Format:<br />
193 * array(package1,package2,...)<br />
194 * or false if not set
195 *
196 * Use this option to limit output similar to ignoring files. If you have
197 * some temporary files that you don't want to specify by name but don't
198 * want included in output, set a package name for all the elements in your
199 * project, and set packageoutput to that name. the default package will be
200 * ignored. Parsing speed does not improve. If you want to ignore files
201 * for speed reasons, use the ignore command-line option
202 * @tutorial phpDocumentor.howto.pkg#using.command-line.packageoutput
203 * @see Io
204 * @var false|array
205 */
206 var $packageoutput = false;
207
208 /**
209 * the functions which handle output from the {@link Parser}
210 * @see handleEvent(), handleDocBlock(), handlePage(), handleClass()
211 * @see handleDefine(), handleFunction(), handleMethod(), handleVar()
212 * @see handlePackagePage(), handleInclude(), handleTutorial()
213 */
214 var $event_handlers = array(
215     'docblock' => 'handleDocBlock',
216     'page' => 'handlePage',
217     'class' => 'handleClass',
218     'define' => 'handleDefine',
219     'function' => 'handleFunction',
220     'method' => 'handleMethod',
221     'var' => 'handleVar',
222     'const' => 'handleConst',
223     'packagepage' => 'handlePackagePage',
224     'include' => 'handleInclude',
225     'global' => 'handleGlobal',
226     'tutorial' => 'handleTutorial',
227 );

```

```

228
229  /**
230   * $data contains parsed structures for the current page being parsed
231   *
232   * In version 1.1+, $data is only used to store the current page information.
233   * All handling of documented elements is handled by the
234   * {@link ProceduralPages} and {@link Classes} classes.
235   * @var parserData
236   */
237 var $data;
238
239 /**
240  * set in {@link Setup.inc.php} to the value of the quitemode commandline
241  * option.
242  *
243  * If this option is true, informative output while parsing will not be
244  * displayed (documentation is unaffected)
245  * @var boolean
246  * @tutorial phpDocumentor.howto.pkg#using.command-line.quiet
247  */
248 var $quietMode = false;
249
250 /**
251  * set in {@link Setup.inc.php} to the value of the undocumentedElementWarnings commandline
252  * option.
253  *
254  * If this option is true, warnings about certain elements (classes, methods)
255  * that are not documented with DocBlocks will be shown while parsing,
256  * and will also be displayed in the errors.html page
257  * (other documentation is unaffected)
258  * @var boolean
259  * @tutorial phpDocumentor.howto.pkg#using.command-line.undocumentedelements
260  */
261 var $undocumentedElementWarnings = false;
262
263 /**
264  * used to keep track of inheritance at the smartest level possible for a
265  * dumb computer
266  * @var Classes
267  */
268 var $classes = false;
269
270 /**
271  * used to keep track of all elements in a procedural page. Handles name
272  * conflicts with elegance
273  * @since 1.1
274  * @var ProceduralPages
275  */
276 var $proceduralpages = false;
277
278 /**
279  * an array of template names indexed by converter name
280  *
281  * For example, if the default HTMLframesConverter is using the DOM/10133t
282  * template, the array will be
283  * <code>$converters['frames'] = 'DOM/10133t'</code>
284  * @var array Format: array(Converternamel => templatename)
285  * @see Converter
286  */
287 var $converters = false;
288 /**
289  * @var string Title of generated documentation, passed to Converters
290  */
291 var $title = '';
292
293 var $uses = array();
294
295 var $db_template;
296
297 /**
298  * Stores parsed CHANGELOG/INSTALL/README files
299  * @var array Format: array(CHANGELOG => contents,
300  *                         INSTALL => contents,
301  *                         README => contents)
302  */
303 var $ric = array();
304
305 /**
306  * Flag used to determine whether the last docblock
307  * was a page-level docblock.

```

```

308     * @var boolean
309     * @access private
310     */
311     var $_lastDocBlockWasPageLevel = false;
312
313     /**
314      * Flag used to determine whether the Page-level
315      * DocBlock was declared in old or new style
316      * @var boolean
317      * @access private
318      */
319     var $_oldPageLevel = false;
320
321     /**
322      * sets up basic data structures
323      * @param string Title of generated documentation, passed to Converters
324      * @see $title, $data, $classes, $proceduralpages
325      */
326     function phpDocumentor_IntermediateParser($title='Generated Documentation')
327     {
328         $this-> title = $title;
329         $this-> data = new parserData;
330         $this-> classes = new Classes;
331         $this-> proceduralpages = new ProceduralPages;
332     }
333
334     /**
335      * Retrieve the relative path. If the path contains "pear/" it will
336      * be used as the base, otherwise the Program_Root string will be used.
337      * @global array uses 'Program_Root' option to replace it with '' for
338      *          retrieving the source location of a file
339      * @param string path to file
340      * @return string
341      * @see $sourceLocation
342      * @access private
343      */
344     function _getSourceLocation($sl, $sourcedloc)
345     {
346         global $_phpDocumentor_options;
347         if (empty($sl)) return false;
348         $sl = str_replace('\\','/',$sl);
349         if (strpos($sl,'pear/'))
350         {
351             $sl = substr($sl,strpos($sl,'pear/') + 5);
352             if (dirname($sl) == '.')
353             {
354                 return 'PEAR';
355             }
356             return dirname($sl);
357         } else
358         {
359             if (strpos(str_replace($_phpDocumentor_options['Program_Root'] .
PATH_DELIMITER,'',$sourcedloc),PATH_DELIMITER) === false)
360                 return '';
361             return dirname(str_replace($_phpDocumentor_options['Program_Root'] .
PATH_DELIMITER,'',$sourcedloc));
362         }
363     }
364
365     /**
366      * Guess the package/subpackage based on subdirectory if the --pear option
367      *
368      * A file in pear/dir/file.php will be in package "dir." A file in
369      * pear/dir/subdir/file.php will be in package "dir," subpackage
370      * "subdir."
371      * @param string full path of file
372      * @param template-ready source location Program_Root/dir/file.php
373      * @global array uses the 'pear' option to determine whether to guess based
374      *          on subdirectory
375      * @tutorial phpDocumentor.howto.pkg#using.command-line.pear
376      */
377     function _guessPackage($path, $sourcedloc)
378     {
379         global $_phpDocumentor_setting;
380         if ($_phpDocumentor_setting['pear'])
381         {
382             $subpath = explode(PATH_DELIMITER, $this-> _getSourceLocation($path, $sourcedloc));
383             if (!empty($subpath[0]))
384             { // can only have package and subpackage in this version
385                 $package = $subpath[0];

```

```

385             $subpackage = '';
386             if (isset($subpath[1])) $subpackage = $subpath[1];
387             return array($package,$subpackage);
388         } else return array($this-> package, $this-> subpackage);
389     } else return array($this-> package, $this-> subpackage);
390 }
391 /**
392 * handles post-parsing of include/require/include_once/require_once
393 *
394 * This function sets {@link $data}->clean to false to tell the
395 * phpDocumentor_IntermediateParser that a page-level DocBlock can't be
396 * found after this point on this page. It then sets the package
397 * to be the same as the page, and adds itself to the
398 * {@link ProceduralPages} class
399 * @param integer $event Event number from {@link Parser.inc}
400 * @param parserInclude $data
401 */
402 function handleInclude($event,$data)
403 {
404     if ($this-> _lastDocBlockWasPageLevel)
405     {
406         addWarning(PDERROR_DOCBLOCK_CONFLICT, $data-> getName(), $data-> getValue());
407         if (!$this-> _oldPageLevel)
408         {
409             unset($this-> last);
410         }
411     }
412     $this-> _lastDocBlockWasPageLevel = false;
413     $this-> data-> clean = false;
414     // page was @ignored
415     if ($this-> private_page)
416     {
417         unset($this-> last);
418         return;
419     }
420     if (empty($this-> last))
421     {
422         if (isset($this-> db_template))
423         {
424             // use the docblock template
425             $this-> last = phpDocumentor_clone($this-> db_template);
426         }
427         else
428         {
429             // we don't have a docblock, create an empty one to get rid of errors
430             $this-> last = new parserDocblock();
431         }
432     }
433     // $this->last->setLineNumber($data->getLineNumber());
434     if ($this-> last-> getKeyword('ignore'))
435     {
436         $this-> last = false;
437         return;
438     }
439     // addWarning(PDERROR_IGNORE_TAG_IGNORED,'include',$data->getName().'.'.$data->getValue().'');
440     }
441
442     $this-> last-> overridePackage($this-> category,$this-> package,$this-
443 > subpackage,$data-> getName(),'include');
444     $data-> setDocBlock($this-> last);
445     $this-> proceduralpages-> addInclude($data);
446     $this-> last = false;
447 }
448 /**
449 * handles post-parsing of global variables
450 *
451 * This function sets {@link $data}->clean to false to tell the
452 * phpDocumentor_IntermediateParser that a page-level DocBlock can't be
453 * found after this point on this page. It then sets the package
454 * to be the same as the page, and adds itself to the
455 * {@link ProceduralPages} class
456 * @param integer $event Event number from {@link Parser.inc}
457 * @param parserGlobal $data
458 */
459 function handleGlobal($event,$data)
460 {
461     if ($this-> _lastDocBlockWasPageLevel)
462     {

```

```

463         addWarning(PDERROR_DOCBLOCK_CONFLICT, 'global variable', $data->    getName());
464         if (!$this->  _oldPageLevel)
465         {
466             unset($this->  last);
467         }
468     }
469     $this->  _lastDocBlockWasPageLevel = false;
470     $this->  data->  clean = false;
471     if ($this->  private_page)
472     {
473         unset($this->  last);
474         return;
475     }
476     if (empty($this->  last))
477     {
478         if (isset($this->  db_template))
479         {
480             // use the docblock template
481             $this->  last = phpDocumentor_clone($this->  db_template);
482         }
483         else
484         {
485             // we don't have a docblock, create an empty one to get rid of errors
486             $this->  last = new parserDocblock();
487         }
488     }
489     // $this->last->setLineNumber($data->getLineNumber());
490     if ($this->  last->  getKeyword('ignore'))
491     {
492         addWarning(PDERROR_IGNORE_TAG_IGNORED,'global variable - just don\'t document
the',$data->  getName());
493         $this->  last = false;
494         return;
495     }
496     $this->  last->  overridePackage($this->  category,$this->  package,$this-
>  subpackage,$data->  getName(),'global');
497     $data->  setDocBlock($this->  last);
498     if ($data->  docblock->  getKeyword('name'))
499     {
500         $a = $data->  docblock->  getKeyword('name');
501         if (is_object($a)) $a = $a->  value;
502         $data->  setName($a);
503     }
504     $this->  proceduralpages->  addGlobal($data);
505     $this->  last = false;
506 }
507 /**
508 * handles post-parsing of Package-level documentation pages.
509 *
510 * sets the {@link $package_pages}[$data->package] to $data
511 * @param integer $event Event number from {@link Parser.inc}
512 * @param parserPackagePage $data
513 */
514 function handlePackagePage($event,$data)
515 {
516     $this->  package_pages[$data->  package] = &      $data;
517     $this->  last = false;
518 }
519 /**
520 * handle post-parsing of Tutorials.
521 *
522 * This adds the parsed tutorial to the tutorial tree
523 * @uses $tutorials sets the value of tutorials to parameter $data
524 * @param integer $event Event Number
525 * @param parserTutorial $data
526 * @since 1.2
527 */
528 function handleTutorial($event,$data)
529 {
530     if (isset($this->  packagecategories[$data->  package]))
531     {
532         $data->  category = $this->  packagecategories[$data->  package];
533     }
534     else
535     {
536         $data->  category = $GLOBALS['phpDocumentor_DefaultCategoryName'];
537     }
538     $this->  tutorials[$data->  package][$data->  subpackage][$data-
>  tutorial_type][$data->  name] = $data;
539 }

```

```

540 }
541 /**
542 * handles post-parsing of class vars
543 *
544 * This function sets up a @var tag if none is found, and aligns $data's
545 * $path var and packages to match the parent object
546 * @param integer $event Event number from {@link Parser.inc}
547 * @param parserVar $data
548 */
549 function handleVar($event,$data)
550 {
551     global $_phpDocumentor_setting;
552     if ($this-> private_class)
553     {
554         unset($this-> last);
555         return;
556     }
557     if (empty($this-> last))
558     {
559         if (isset($this-> db_template))
560         {
561             // use the docblock template
562             $this-> last = phpDocumentor_clone($this-> db_template);
563         }
564         else
565         {
566             // we don't have a docblock, create an empty one to get rid of errors
567             $this-> last = new parserDocblock();
568         }
569     }
570     // $this->last->setLineNumber($data->getLineNumber());
571     $this-> last-> overridePackage($this-> category,$this-> package,$this-
572 > subpackage,$data-> getName(),'var');
573     $this-> last-> updateModifiers($data-> getModifiers());
574
575     if ($this-> last-> getKeyword('ignore'))
576     {
577         $this-> last = false;
578         return;
579     // addWarning(PDERROR_IGNORE_TAG_IGNORED,'var',$this->cur_class.'::'.$data-
580 > getName());
581     if (!$this-> last-> var)
582     {
583         $this-> last-> addVar('mixed',new parserStringWithInlineTags());
584     }
585
586     if ($_phpDocumentor_setting['pear'])
587     {
588         if (strpos($data-> getName(), '_') == 1 && !
589 > getKeyword('access'))
590         {
591             addWarning(PDERROR_PRIVATE_ASSUMED,'class variable',$data-> class.'::'.$data-
592 > getName());
593             $this-> last-> addKeyword('access','private');
594             $data-> setDocBlock($this-> last);
595         }
596         $data-> setDocBlock($this-> last);
597         $data-> path = $this-> data-> parent-> path;
598         $this-> classes-> addVar($data);
599         $this-> last = false;
600     }
601 /**
602 * handles post-parsing of class constants
603 *
604 * This function aligns $data's
605 * $path var and packages to match the parent object
606 * @param integer $event Event number from {@link Parser.inc}
607 * @param parserVar $data
608 */
609 function handleConst($event,$data)
610 {
611     global $_phpDocumentor_setting;
612     if ($this-> private_class)
613     {
614         unset($this-> last);
615         return;

```

```

616 }
617 if (empty($this-> last))
618 {
619     if (isset($this-> db_template))
620     {
621         // use the docblock template
622         $this-> last = phpDocumentor_clone($this-> db_template);
623     }
624     else
625     {
626         // we don't have a docblock, create an empty one to get rid of errors
627         $this-> last = new parserDocblock();
628     }
629 }
630 // $this->last->setLineNumber($data->getLineNumber());
631 $this-> last-> overridePackage($this-> category,$this-> package,$this-
> subpackage,$data-> getName(),'const');
632
633 if ($this-> last-> getKeyword('ignore'))
634 {
635     $this-> last = false;
636     return;
637 // addWarning(PDERROR_IGNORE_TAG_IGNORED,'var',$this->cur_class.'::' . $data-
>getName());
638 }
639 $data-> setDocBlock($this-> last);
640 $data-> path = $this-> data-> parent-> path;
641 $this-> classes-> addConst($data);
642 $this-> last = false;
643 }
644
645 /**
646 * handles post-parsing of class methods
647 *
648 * This function first aligns $data's path and package to match the parent
649 * object, and also aligns the docblock's @param, @global, and @staticvar
650 * tags with the information parsed from the method source code. It also
651 * checks to see if the method is a constructor and sets the $isConstructor
652 * flag. If source code has been parsed by a {@source} tag, the source is
653 * added to its docblock
654 *
655 * Finally, it adds the method to the {@link Classes} class.
656 * @param integer $event Event number from {@link Parser.inc}
657 * @param parserMethod $data
658 */
659 function handleMethod($event,$data)
660 {
661     global $phpDocumentor_setting;
662     if ($this-> private_class)
663     {
664         unset($this-> last);
665         return;
666     }
667
668 if (empty($this-> last))
669 {
670     if ($this-> undocumentedElementWarnings)
671     {
672         addWarning(PDERROR_UNDOCUMENTED_ELEMENT,'Method',$data-> getName(),'method');
673     }
674     if (isset($this-> db_template))
675     {
676         // use the docblock template
677         $this-> last = phpDocumentor_clone($this-> db_template);
678     }
679     else
680     {
681         // we don't have a docblock, create an empty one to get rid of errors
682         $this-> last = new parserDocblock();
683     }
684 }
685 // $this->last->setLineNumber($data->getLineNumber());
686 if ($this-> last-> getKeyword('ignore'))
687 {
688     $this-> last = false;
689     return;
690 // addWarning(PDERROR_IGNORE_TAG_IGNORED,'method',$this->cur_class.'::' . $data-
>getName());
691 }
692 $this-> last-> overridePackage($this-> category,$this-> package,$this-

```

```

> subpackage,$data-> getName(),'method');
693     if ($data-> hasSource())
694     {
695         $this-> last-> setSource($data-> getSource(), $data-> getClass());
696     }
697     foreach($data-> listParams() as $key => $param)
698     {
699         $update_params[$key] = $param;
700     }
701     foreach($data-> listGlobals() as $param)
702     {
703         $update_globals[] = $param[1];
704     }
705     foreach($data-> listStatics() as $param)
706     {
707         $update_statics[] = $param[0];
708     }
709     if (isset($update_params))
710     $this-> last-> updateParams($update_params);
711     if (isset($update_globals))
712     $this-> last-> updateGlobals($update_globals);
713     if (isset($update_statics))
714     $this-> last-> updateStatics($update_statics);
715     $this-> last-> updateModifiers($data-> getModifiers());
716     unset($update_params);
717     unset($update_globals);
718     unset($update_statics);
719
720     if ($data-> getName() == $this-> cur_class) $data-> setConstructor();
721     if ($data-> getName() == '__construct') {
722         $data-> setConstructor();
723     }
724     if ($data-> getName() == '__destruct') {
725         $data-> setDestructor();
726     }
727
728     if ($phpDocumentor_setting['pear'])
729     {
730         if (strpos($data-> getName(), '_') === 0 && substr($data-> getName(),
1) == $data-> class)
731         {
732             // is destructor
733             $data-> setDestructor();
734         } elseif (strpos($data-> getName(), '_') === 0 && ! $this-> last-
> getKeyword('access'))
735         {
736             if (strpos($data-> getName(), '_') !== 0) {
737                 addWarning(PDERROR_PRIVATE_ASSUMED,'method',$data-> class.'::'.$data-
> getName().'()');
738             $this-> last-> addKeyword('access','private');
739             $data-> setDocBlock($this-> last);
740         }
741     }
742     $data-> setDocBlock($this-> last);
743     $data-> path = $this-> data-> parent-> path;
744     $this-> classes-> addMethod($data);
745     $this-> last = false;
746 }
747 /**
748 * handles post-parsing of functions
749 *
750 * This function sets {@link $data}->clean to false to tell the
751 * phpDocumentor_IntermediateParser that a page-level DocBlock can't be
752 * found after this point on this page. It then sets the package to be the
753 * same as the page, aligns the docblock's @param, @global, and @staticvar
754 * tags with the information parsed from the function source code.
755 *
756 * If source code has been parsed by a {@source} tag, the source is added
757 * to its docblock, and then the parserFunction adds itself to the
758 * {@link ProceduralPages} class
759 * @param integer $event Event number from {@link Parser.inc}
760 * @param parserFunction $data
761 */
762
763 function handleFunction($event,$data)
764 {
765     if ($this-> _lastDocBlockWasPageLevel)
766     {
767         addWarning(PDERROR_DOCBLOCK_CONFLICT, 'function', $data-> getName());
768         if (!$this-> _oldPageLevel)

```

```

769         {
770             unset($this-> last);
771         }
772     }
773     $this-> _lastDocBlockWasPageLevel = false;
774     $this-> data-> clean = false;
775     if ($this-> private_page)
776     {
777         unset($this-> last);
778         return;
779     }
780     if (empty($this-> last))
781     {
782         if (isset($this-> db_template))
783         {
784             // use the docblock template
785             $this-> last = phpDocumentor_clone($this-> db_template);
786         }
787         else
788         {
789             // we don't have a docblock, create an empty one to get rid of errors
790             $this-> last = new parserDocblock();
791         }
792     }
793 }
794 // $this->last->setLineNumber($data->getLineNumber());
795 if ($this-> last-> getKeyword('ignore'))
796 {
797     unset($this-> last);
798     return;
799 }
800 $this-> last-> overridePackage($this-> category,$this-> package,$this-
> subpackage,$data-> getName(),'function');
801
802     foreach($data-> listParams() as $key => $param)
803     {
804         $update_params[$key] = $param;
805     }
806     foreach($data-> listGlobals() as $param)
807     {
808         $update_globals[] = $param[1];
809     }
810     foreach($data-> listStatics() as $param)
811     {
812         $update_statics[] = $param[0];
813     }
814     if (isset($update_params))
815     $this-> last-> updateParams($update_params);
816     if (isset($update_globals))
817     $this-> last-> updateGlobals($update_globals);
818     if (isset($update_statics))
819     $this-> last-> updateStatics($update_statics);
820     unset($update_params);
821     unset($update_globals);
822     unset($update_statics);
823
824     if ($data-> hasSource())
825     {
826         $this-> last-> setSource($data-> getSource());
827     }
828     if (count($this-> last-> params) == 1 && !
829     {
830         // if the function has no parameters, and 1 @param, add it to the list as optional,
831         default value is description from @param
832         $pars = $this-> last-> listParams();
833         $data-> addParam($pars[0]['var'],$pars[0]['data']-> getString());
834     }
835     $data-> setDocBlock($this-> last);
836     $this-> proceduralpages-> addFunction($data);
837     $this-> last = false;
838 }
839 /**
840 * handles post-parsing of defines
841 *
842 * This function sets {@link $data}->clean to false to tell the
843 * phpDocumentor_IntermediateParser that a page-level DocBlock can't be
844 * found after this point on this page. It then sets the package to be the
845 * same as the page and adds itself to the {@link ProceduralPages} class
846 * @param integer $event Event number from {@link Parser.inc}

```

```

847     * @param parserDefine $data
848     */
849     function handleDefine($event,$data)
850     {
851         if ($this-> _lastDocBlockWasPageLevel)
852         {
853             addWarning(PDERROR_DOCBLOCK_CONFLICT, 'define', $data-> getName());
854             if (!$this-> _oldPageLevel)
855             {
856                 unset($this-> last);
857             }
858         }
859         $this-> _lastDocBlockWasPageLevel = false;
860         $this-> data-> clean = false;
861         if ($this-> private_page)
862         {
863             unset($this-> last);
864             return;
865         }
866         if (empty($this-> last))
867         {
868             if (isset($this-> db_template))
869             {
870                 // use the docblock template
871                 $this-> last = phpDocumentor_clone($this-> db_template);
872             }
873             else
874             {
875                 // we don't have a docblock, create an empty one to get rid of errors
876                 $this-> last = new parserDocblock();
877             }
878         }
879         // $this->last->setLineNumber($data->getLineNumber());
880         if ($this-> last-> getKeyword('ignore'))
881         {
882             unset($this-> last);
883             return;
884         }
885         $this-> last-> overridePackage($this-> category,$this-> package,$this-
886 > subpackage,$data-> getName(),'define');
887         $data-> setDocBlock($this-> last);
888         $this-> proceduralpages-> addDefine($data);
889         $this-> last = false;
890     }
891
892 /**
893 * handles post-parsing of classes
894 *
895 * This function sets {@link $data}->clean to false to tell the
896 * phpDocumentor_IntermediateParser that a page-level DocBlock can't be
897 * found after this point on this page. It sets {@link $cur_class} to its
898 * name, and if an @ignore tag is found in the DocBlock, it sets
899 * {@link $private_class} to true, to prevent post-parsing of any of the
900 * class's vars or methods. Then it checks for the existence of a package
901 * page for the class's package
902 * @param integer $event Event number from {@link Parser.inc}
903 * @param parserClass $data
904 */
905 function handleClass($event,$data)
906 {
907     global $_phpDocumentor_setting;
908     if ($data-> isInterface())
909     {
910         $objectType = 'interface';
911     }
912     else
913     {
914         $objectType = 'class';
915     }
916     if ($this-> _lastDocBlockWasPageLevel)
917     {
918         if (!$this-> _oldPageLevel)
919         {
920             addWarning(PDERROR_DOCBLOCK_Goes_CLASS, $data-> getName());
921             $doc = new parserDocBlock;
922             $doc-> category = $this-> category;
923             $doc-> package = $this-> package;
924             $doc-> subpackage = $this-> subpackage;
925             if ($_phpDocumentor_setting['sourcocode']) {

```

```

926             $doc-> canSource();
927             $doc-> addFileSource($this-> data-> parent-> path, $this-> data-
> parent-> source);
928         }
929         $this-> data-> setDocBlock($doc);
930         unset($doc);
931         if ($this-> last) {
932             $this-> last-> cantSource();
933         }
934     }
935 }
936 $this-> lastDocBlockWasPageLevel = false;
937 $this-> data-> clean = false;
938 if (empty($this-> last))
939 {
940     if ($this-> undocumentedElementWarnings)
941     {
942         addWarning(PDERROR_UNDOCUMENTED_ELEMENT,'Class',$data-> getName(),'Class');
943     }
944     if (isset($this-> db_template))
945     {
946         // use the docblock template
947         $this-> last = phpDocumentor_clone($this-> db_template);
948     }
949     else
950     {
951         // we don't have a docblock, create an empty one to get rid of errors
952         $this-> last = new parserDocblock();
953     }
954     list($this-> last-> package, $this-> last-> subpackage) = $this-
> _guessPackage($this-> data-> parent-> path, $this-> data-> parent-
> getSourceLocation('dummy'));
955     addWarning(PDERROR_NO_PACKAGE_TAG,$objectType,$data-> getName(),$this-> last-
> package);
956 } else
957 {
958     if (!$this-> last-> getExplicitPackage())
959     {
960         list($this-> last-> package, $this-> last-> subpackage) = $this-
> _guessPackage($this-> data-> parent-> path, $this-> data-> parent-
> getSourceLocation('dummy'));
961         addWarning(PDERROR_NO_PACKAGE_TAG,$objectType,$data-> getName(),$this-
> last-> package);
962     } else
963     {
964         if (isset($this-> packageCategories[$this-> package])
965             && $this-> packageCategories[$this-> package] != $this-
> category)
966             addWarning(PDERROR_PACKAGECAT_SET,$this-> package,
967                         $this-> packageCategories[$this-> package],
968                         $this-> category);
969         $this-> packageCategories[$this-> package] = $this-> category;
970     }
971 }
972 $this-> last-> updateModifiers($data-> getModifiers());
973 // $this->last->setLineNumber($data->getLineNumber());
974 $data-> setDocBlock($this-> last);
975 $this-> cur_class = $name = $data-> getName();
976 if ($this-> last-> getKeyword('ignore'))
977 {
978     $this-> private_class = true;
979     unset($this-> last);
980     return;
981 }
982 $data-> path = $this-> data-> parent-> path;
983 $this-> classes-> addClass($data);
984 $this-> private_class = false;
985 if ($this-> last-> package)
986 {
987     $this-> parsePackagePage($this-> last-> package, $this-> data-> parent-
> getPath());
988 }
989 $this-> last = false;
990 }
991 /**
992 * handles post-parsing of procedural pages
993 *
994 * this event is called at the start of a new page, before the Parser knows
995 * whether the page will contain any procedural pages or not

```

```

997 * @param integer $event Event number from {@link Parser.inc}
998 * @param parserPage $data
999 */
1000 function handlePage($event,$data)
1001 {
1002     $type = 'page';
1003     $this-> private_page = false;
1004     $this-> data = new parserData;
1005     $data-> category = $this-> category = $GLOBALS['phpDocumentor_DefaultCategoryName'];
1006     $this-> package = $GLOBALS['phpDocumentor_DefaultPackageName'];
1007     $this-> subpackage = '';
1008     $this-> proceduralpages-> addPage($data);
1009     $this-> data-> setParent($data);
1010     $this-> pages[$data-> getPath()] = $this-> data;
1011     $this-> classes-> nextFile($data-> getPath());
1012     $this-> packageoutput = $data-> getPackageOutput();
1013 }
1014
1015 /**
1016 * handles post-parsing of DocBlocks
1017 *
1018 * This function sets {@link $last} to the DocBlock represented by $data, to
1019 * allow the next documentable element passed to
1020 * phpDocumentor_IntermediateParser to link the DocBlock into its $docblock
1021 * property. This function also checks for two special cases of DocBlocks:
1022 * <ol>
1023 *   <li>First DocBlock in the file contains a @package tag</li>
1024 *   <li>First DocBlock in the file is immediately followed by another
1025 *     DocBlock</li>
1026 * </ol>
1027 * In both cases, the function extracts this tag and uses it as the
1028 * page-level package. If the @package tag is in the DocBlock of an
1029 * element (function, global variable, whatever) that isn't a page-level
1030 * DocBlock, a warning will be raised to notify the author that a @package
1031 * tag belongs in a page-level DocBlock.
1032 *
1033 * <b>New</b> in version 1.2.2, if the first DocBlock in a file contains
1034 * a @package tag, it is a page-level DocBlock.
1035 *
1036 * If the DocBlock is page-level, it is processed with
1037 * {@link _processPageLevelDocBlock}
1038 *
1039 * Finally, the function replaces the old parserPage in
1040 * {@link parserData::$data}->parent with the new one containing information
1041 * from the DocBlock by calling {@link addPage()}, and checks for
1042 * package-level docs.
1043 * @param integer $event Event number from {@link Parser.inc}
1044 * @param parserDocBlock $data
1045 */
1046 function handleDocBlock($event,$data)
1047 {
1048     $type = 'docblock';
1049     $data-> postProcess();
1050     // Zend desc support
1051     if ($tdesc = $data-> getKeyword('desc'))
1052     {
1053         $data-> setShortDesc($tdesc);
1054         unset($data-> tags['desc']);
1055     }
1056     $this-> lastDocBlockWasPageLevel = false;
1057     // 1st docblock in file, check for @package
1058     if ($this-> data-> isClean() && !isset($this-> last))
1059     {
1060         if ($data-> getExplicitPackage())
1061         {
1062             // new with 1.2.2:
1063             // if the first docblock in a file
1064             // contains a @package tag, then it is
1065             // a page-level docblock
1066             $this-> _processPageLevelDocBlock($data);
1067             $this-> lastDocBlockWasPageLevel = true;
1068             $this-> all_packages[$data-> package] = 1;
1069             $this-> last = $data;
1070             return;
1071         }
1072         $doc = new parserDocBlock;
1073         $doc-> category = $this-> category;
1074         $doc-> package = $this-> package;
1075         $doc-> subpackage = $this-> subpackage;
1076         $this-> data-> setDocBlock($doc);

```

```

1077         $this-> proceduralpages-> addPagePackage($this-> data-> parent-
1078 > getPath(),$this-> package,$this-> subpackage);
1079         unset($doc);
1080     }
1081     // 2nd docblock in a row, and it's at the top of the file, page-level docblock
1082     if ($this-> lasttype == "docblock" && $this-> data-> isClean())
1083     {
1084         $this-> _processPageLevelDocBlock($this-> last);
1085         $this-> _oldPageLevel = true;
1086         $this-> _lastDocBlockWasPageLevel = false;
1087     }
1088     $this-> all_packages[$data-> package] = 1;
1089     $this-> last = $data;
1090 }
1091 /**
1092 * Process a Page-level DocBlock
1093 *
1094 * First, it checks for an @ignore tag,
1095 * and if found, calls {@link ProceduralPages::ignorePage()}. An @ignore
1096 * tag in a page-level DocBlock will ignore all functions, defines, global
1097 * variables, and includes. It will not ignore classes! The function next
1098 * checks for an @access private, and if --parseprivate is off, performs the
1099 * same actions as @ignore.
1100 * Next, it checks for the @name tag, which is used to rename the page.
1101 * This is also a PEAR compatibility issue, and may not be very useful in
1102 * the long run. Documentation is best when it refers to real entities in
1103 * the package, and not to aliases.
1104 * @access private
1105 */
1106 function _processPageLevelDocBlock($data)
1107 {
1108     global $_phpDocumentor_setting;
1109     // can only have 1 package-level docblock, others are ignored
1110     if (!$this-> data-> isClean())
1111     {
1112         return;
1113     }
1114     $this-> data-> clean = false;
1115     $this-> data-> explicitDocBlock();
1116     $data-> canSource();
1117     if ($_phpDocumentor_setting['sourcecode'])
1118     {
1119         $data-> addFileSource($this-> data-> parent-> path, $this-> data-
1120 > parent-> source);
1121     }
1122     if (!$data-> getExplicitPackage())
1123     {
1124         list($data-> package,$data-> subpackage) = $this-> _guessPackage($this-
1125 > data-> parent-> getPath(), $this-> data-> parent-> getSourceLocation('dummy'));
1126         addWarning(PDERROR_NO_PACKAGE_TAG,'file',$this-> data-> parent-
1127 > getPath(),$this-> last-> package);
1128         if (isset($this-> packagecategories[$this-> package])
1129             && $this-> packagecategories[$this-> package] != $data-> category)
1130             addWarning(PDERROR_PACKAGECAT_SET,$this-> package,
1131                         $this-> packagecategories[$this-> package],
1132                         $data-> category);
1133         $this-> packagecategories[$this-> package] = $data-> category;
1134         $this-> category = $this-> data-> parent-> category = $data-> category;
1135         $this-> packagecategories[$this-> package] = $this-> category;
1136         $this-> subpackage = $this-> data-> parent-> subpackage = $data-> subpackage;
1137         $this-> data-> setDocBlock($data);
1138         if ($data-> getKeyword('ignore'))
1139         {
1140             $this-> proceduralpages-> ignorePage($this-> data-> parent);
1141             $this-> private_page = true;
1142             unset($this-> last);
1143             $this-> privatepages[$this-> data-> parent-> getPath()] = $this-> data;
1144             unset($this-> pages[$this-> data-> parent-> getPath()]);
1145             return;
1146         }
1147         $this-> package = $this-> data-> parent-> package = $data-> package;
1148         $this-> subpackage = $this-> data-> parent-> subpackage = $data-> subpackage;
1149         $this-> proceduralpages-> addPagePackage($this-> data-> parent-
1150 > getPath(),$this-> package,$this-> subpackage);
1151         if ($access = $data-> getKeyword('access'))
1152         {
1153             if (is_object($access) && ($access-> getString() == 'private') &&
1154 (!$this-> parsePrivate))

```

```

1151         {
1152             $this-> proceduralpages-> ignorePage($this-> data-> parent);
1153             $this-> private_page = true;
1154             unset($this-> last);
1155             $this-> privatepages[$this-> data-> parent-> getPath()] = $this-
> data;
1156             unset($this-> pages[$this-> data-> parent-> getPath()]);
1157             return;
1158         }
1159     }
1160     if ($data-> getKeyword('name'))
1161     {
1162         $a = $data-> getKeyword('name');
1163         if (is_object($a)) $a = $a-> value;
1164         $this-> data-> parent-> setFile($a);
1165         $this-> proceduralpages-> setName($a);
1166     }
1167     $this-> addPage($this-> data-> parent, $this-> data-> parent-> getPath());
1168     if ($this-> package)
1169     {
1170         $this-> parsePackagePage($this-> package, $this-> data-> parent-
> getPath());
1171     }
1172 }
1173 /**
1174 * Backward-compatibility only, use the new tutorials for more power
1175 * @tutorial tutorials.pkg
1176 * @param string package name of package file to parse
1177 * @param string directory of file that contains package name
1178 */
1179 function parsePackagePage($package, $path)
1180 {
1181     if (!isset($this-> package_pages[$package]))
1182     {
1183         if (file_exists(dirname($path) . SMART_PATH_DELIMITER . $package . '.html'))
1184         {
1185             if ($this-> quietMode === false)
1186             {
1187                 phpDocumentor_out("Reading package-level file " . $package .
'.html');
1188                 flush();
1189             }
1190             $fp = fopen(dirname($path) . SMART_PATH_DELIMITER . $package .
'.html', "r");
1191             $ret = fread($fp, filesize(dirname($path) . SMART_PATH_DELIMITER . $package .
'.html'));
1192             fclose($fp);
1193             unset($fp);
1194             if ($this-> quietMode === false)
1195             {
1196                 phpDocumentor_out(" -- Parsing File\n");
1197                 flush();
1198             }
1199             $pageParser = new ppageParser;
1200             $temp = $this-> package;
1201             $lp = $this-> last;
1202             $pageParser-> subscribe('*', $this);
1203             $pageParser-> parse($ret, false, $package);
1204             $this-> package = $temp;
1205             $this-> last = $lp;
1206             unset($temp);
1207             unset($pageParser);
1208         }
1209     }
1210 }
1211 /**
1212 * called via {@link Parser::parse()} and Parser's inherited method
1213 * {@link Publisher::publishEvent()}
1214 *
1215 * $event is one of the PHPDOC constants from Parser.inc. If it is not
1216 * PHPDOCUMENTOR_EVENT_NEWSTATE, then a function name is retrieved from the
1217 * {@link $event_handlers} array and called to handle the $data
1218 * @param integer $event event number from {@link Parser.inc}
1219 * @param mixed $data if $event is {@link PHPDOCUMENTOR_EVENT_NEWSTATE}, $data is a {@link
1220 PHP_DOC_EVENT_END_PAGE} or {@link STATE_END_CLASS},
1221 * otherwise $data is either a {@link parserDocBlock}, {@link
1222 parserPage} or descendant of {@link parserElement}
1223 * @global array we use 'sourcecode' to determine whether to highlight the source

```

```

1224 *          of the current file if it has no file-level docblock
1225 */
1226 function HandleEvent($event,$data)
1227 {
1228     global $phpDocumentor_setting;
1229     global $phpDocumentor_DefaultPackageName, $phpDocumentor_DefaultCategoryName;
1230     if (empty($this-> packageCategories))
1231     $this-> packageCategories[$phpDocumentor_DefaultPackageName] =
$phpDocumentor_DefaultCategoryName;
1232     if ($event == PHPDOCUMENTOR_EVENT_NEWSSTATE)
1233     {
1234         if ($data == STATE_END_CLASS)
1235         {
1236             elseif ($data == PHPDOCUMENTOR_EVENT_END_PAGE)
1237             {
1238                 if (!$this-> private_page)
1239                 {
1240                     $this-> all_packages[$this-> package] = 1;
1241                     if (!$this-> data-> hasExplicitDocBlock())
1242                     {
1243                         $doc = $this-> data-> docblock;
1244                         if (!$this-> data-> docblock)
1245                         {
1246                             $doc = new parserDocBlock;
1247                         }
1248                         if ($phpDocumentor_setting['sourcecode'])
1249                         {
1250                             $doc-> canSource();
1251                             $doc-> addFileSource($this-> data-> parent-> path, $this-
> data-> parent-> source);
1252                         }
1253                         list($doc-> package,$doc-> subpackage) = $this-
> _guessPackage($this-> data-> parent-> getPath(), $this-> data-> parent-
> getSourceLocation('dummy'));
1254                         addWarning(PDERROR_NO_PAGE_LEVELDOCBLOCK,$this-> data-> parent-
> getPath());
1255                         $this-> data-> setDocBlock($doc);
1256                         $this-> proceduralpages-> addPage($this-> data-> parent,$doc-
> package,$doc-> subpackage);
1257                         $this-> pages[$this-> data-> parent-> getPath()] = $this-> data;
1258                     }
1259                     $this-> private_page = false;
1260                     $this-> private_class = false;
1261                     if (isset($this-> db_template))
1262                     {
1263                         addWarning(PDERROR_DB_TEMPLATE_UNTERMINATED);
1264                     }
1265                     unset($this-> db_template);
1266                     unset($this-> last);
1267                 } elseif ($data == PHPDOCUMENTOR_EVENT_END_DOCBLOCK_TEMPLATE)
1268                 {
1269                     unset($this-> db_template);
1270                 }
1271                 //echo $this->state_lookup[$data] . "\n";
1272                 //echo $data."\n";
1273             }
1274         }
1275     }
1276     else
1277     {
1278         if ($event == PHPDOCUMENTOR_EVENT_README_INSTALL_CHANGELOG)
1279         {
1280             $this-> ric[$data[0]] = $data[1];
1281             return;
1282         }
1283         if ($event == PHPDOCUMENTOR_EVENT_DOCBLOCK_TEMPLATE)
1284         {
1285             $data-> postProcess();
1286             $this-> db_template = $data;
1287             $this-> _lastDocBlockWasPageLevel = false;
1288             // 2nd docblock in a row, and it's at the top of the file, page-level docblock
1289             if ($this-> type == "docblock" && $this-> data-
> isClean())
1290             {
1291                 // can only have 1 package-level docblock, others are ignored
1292                 $this-> data-> clean = false;
1293                 if ($this-> last-> getKeyword('ignore'))
1294                 {
1295                     $this-> proceduralpages-> ignorePage($this-> data-> parent);
1296                     $this-> private_page = true;
1297                     unset($this-> last);
1298                 }
1299             }
1300         }
1301     }
1302 }

```

```

1297             $this-> privatepages[$this-> data-> parent-> getPath()] = $this-
> data;
1298             unset($this-> pages[$this-> data-> parent-> getPath()]);
1299             return;
1300         }
1301         $this-> data-> setDocBlock($this-> last);
1302         $this-> package = $this-> data-> parent-> package = $this-> last-
> package;
1303         $this-> subpackage = $this-> data-> parent-> subpackage = $this-
> last-> subpackage;
1304         $this-> proceduralpages-> addPagePackage($this-> data-> parent-
package,$this-> subpackage);
1305         if ($access = $this-> last-> getKeyword('access'))
1306         {
1307             if (is_object($access) && (
1308                 $this-> parsePrivate))
1309             {
1310                 addWarning(PDERROR_PARSEPRIVATE, $this-> data-> parent-
$this-> proceduralpages-> ignorePage($this-> data-> parent));
1311                 $this-> private_page = true;
1312                 unset($this-> last);
1313                 $this-> privatepages[$this-> data-> parent-> getPath()] =
1314                     unset($this-> pages[$this-> data-> parent-> getPath()]);
1315                 return;
1316             }
1317         }
1318         if ($this-> last-> getKeyword('name'))
1319         {
1320             $a = $this-> last-> getKeyword('name');
1321             if (is_object($a)) $a = $a-> value;
1322             $this-> data-> parent-> setFile($a);
1323             $this-> proceduralpages-> setName($a);
1324         }
1325         $this-> addPage($this-> data-> parent, $this-> data-> parent-
> getPath());
1326         if ($this-> package)
1327         {
1328             $this-> parsePackagePage($this-> package, $this-> data-> parent-
> getPath());
1329         }
1330         unset($this-> last);
1331     } else
1332     {
1333         $this-> lasttype = $this-> type;
1334         $type = $data-> getType();
1335         fancy_debug($type,$data);
1336         // if (( $type != 'page' ) && ( $type != 'docblock' ) && ( $type !=
1337         'packagepage' ) && (
1338             $data-> setFile($this-> data-> parent-> getFile()));
1339         $this-> type = $type;
1340         //echo $type . "\n";
1341         if (isset($this-> event_handlers[$type]))
1342         {
1343             $handle = $this-> event_handlers[$type];
1344             $this-> $handle($event,$data);
1345         }
1346     }
1347 }
1348 }
1349 }
1350 }
1351 }
1352 }
1353 /**
1354 * Replaces the {@link parserPage} represented by $this->pages[$path] with
1355 * $page
1356 *
1357 * Called by {@link addPageIfNecessary()}, handleDocBlock()} and
1358 * {@link ProceduralPages::setupPages()}, this method first checks to see if
1359 * the page has been added. If not, it assumes that the page has either
1360 * been @ignored or set with @access private with --parseprivate off, and
1361 * returns {@link addPrivatePage()}. Otherwise, it sets the pages[$path] to
1362 * be the parserPage $page and sets the package and subpackage to that of
1363 * $page
1364 * @see $pages
1365 * @param parserPage
1366 * @param string full path to the file

```

```

1367 */
1368 function addPage($page, $path)
1369 {
1370     if (!isset($this-> pages[$path])) return $this-> addPrivatePage($page, $path);
1371     $this-> pages[$path]-> setParent($page);
1372     if ($page-> package != $GLOBALS['phpDocumentor_DefaultPackageName'])
1373     {
1374         if (!$this-> pages[$path]-> docblock)
1375         {
1376             $docblock = new parserDocBlock;
1377             $docblock-> package = $page-> package;
1378             $docblock-> subpackage = $page-> subpackage;
1379             $this-> pages[$path]-> docblock = $docblock;
1380         } else
1381         {
1382             $this-> pages[$path]-> docblock-> package = $page-> package;
1383             $this-> pages[$path]-> docblock-> subpackage = $page-> subpackage;
1384         }
1385     }
1386 }
1387 /**
1388 * add a new {@link parserPage} to the $pages array if none is found
1389 *
1390 * This method is used when a page has been @ignored or marked with @access
1391 * private, and a public class is in the page (a class with no @access
1392 * private in its DocBlock). The method first creates a new page in the
1393 * {@link $pages} array and then copies path information, and calls
1394 * {@link addPage()} to set up packages
1395 * @param string full path of page
1396 */
1397 function addPageIfNecessary($path, & $class)
1398 {
1399     global $_phpDocumentor_setting;
1400     if (!$this-> parsePrivate)
1401     {
1402         if (!isset($this-> pages[$path]))
1403         {
1404             $this-> pages[$path] = new parserData;
1405             $this-> pages[$path]-> docblock = new parserDocBlock;
1406             $this-> pages[$path]-> docblock-> package = $this-> privatepages[$path]-
1407 > docblock-> package;
1408             $this-> pages[$path]-> docblock-> subpackage = $this-
1409 > privatepages[$path]-> docblock-> subpackage;
1410             $par = $this-> privatepages[$path]-> parent;
1411             $this-> pages[$path]-> setParent($par);
1412             $this-> proceduralpages-> addPage($par);
1413         }
1414         if (!empty($_phpDocumentor_setting['packageoutput']))
1415             $packages = explode(',', $_phpDocumentor_setting['packageoutput']);
1416         if (!empty($_phpDocumentor_setting['packageoutput']) &&
1417             $this-> pages[$path]-> parent-> package != $class-> docblock-> package
1418 &&
1419             !in_array($this-> pages[$path]-> parent-> package, $packages))
1420         {
1421             $this-> pages[$path]-> parent-> package = $class-> docblock-> package;
1422             $this-> addPage($this-> pages[$path]-> parent, $path);
1423             $this-> proceduralpages-> addPage($this-> pages[$path]-> parent);
1424         }
1425     }
1426 /**
1427 * Adds a {@link parserPage} element to the {@link parserData} element in
1428 * $this->privatepages[$path]
1429 *
1430 * Performs a similar function to addPage, but adds to the
1431 * {@link $privatePages} array
1432 * @param parserPage $page
1433 * @param string $path full path to the page
1434 * @see addPage()
1435 */
1436 function addPrivatePage($page, $path)
1437 {
1438     /*
1439      * if privatepages is still empty,
1440      * we need to initialize it with an empty
1441      * path=>ParserData element, so that it has
1442      * a top-level element... otherwise the setParent() call
1443      * below will crap out.

```

```

1444     */
1445     if (count($this-> privatepages) == 0) {
1446         $this-> privatepages[$path] = new ParserData();
1447     }
1448     $this-> privatepages[$path]-> setParent($page);
1449     if (isset($page-> package) && $page-> package != $GLOBALS['phpDocumentor_DefaultPackageName'])
1450     {
1451         if (!$this-> privatepages[$path]-> docblock)
1452         {
1453             $docblock = new parserDocBlock;
1454             $docblock-> package = $page-> package;
1455             $docblock-> subpackage = $page-> subpackage;
1456             $this-> privatepages[$path]-> docblock = $docblock;
1457         } else
1458         {
1459             $this-> privatepages[$path]-> docblock-> package = $page-> package;
1460             $this-> privatepages[$path]-> docblock-> subpackage = $page-
1461             > subpackage;
1462         }
1463     }
1464
1465 /**
1466 * adds a processed descendant of {@link parserElement} to the {@link $pages}
1467 * array or {@link $privatepages} array
1468 *
1469 * This function expects the page to exist in either $pages or $privatepages. It calls the
1470 * {@link parserData::addElement()} method to add $element to the page.
1471 * @param parserElement $element this will actually be a descendant of parserElement
1472 * @param string $path
1473 */
1474 function addElementToPage($element, $path)
1475 {
1476     if (isset($this-> privatepages[$path]))
1477     {
1478         if (isset($this-> pages[$path]))
1479         {
1480             if ($element-> type == 'class' || $element-> type == 'method'
1481                 || $element-> type == 'var' || $element-> type == 'const')
1482             {
1483                 $this-> pages[$path]-> addElement($element);
1484             } else
1485                 $this-> privatepages[$path]-> addElement($element);
1486         } else
1487             $this-> privatepages[$path]-> addElement($element);
1488     } else
1489     {
1490         if (isset($this-> pages[$path]))
1491         {
1492             $this-> pages[$path]-> addElement($element);
1493         }
1494     }
1495 }
1496
1497 /**
1498 * Add all the @uses tags from $element to the $uses array so that @usedby
1499 * virtual tags can be added
1500 * @uses parserUsesTag::getSeeElement() used to initialize {@link $uses}
1501 * @uses parserUsesTag::getDescription() used to initialize {@link $uses}
1502 * @param parserElement descendant of parserElement
1503 * @param string full path to the file
1504 */
1505 function addUses($element, $path)
1506 {
1507     if (isset($element-> type) && $element-> type == 'page')
1508     {
1509         $element = $this-> pages[$element-> path];
1510     }
1511     if (!$this-> parsePrivate && isset($element-> docblock-> hasaccess)
1512         && $element-> docblock-> hasaccess)
1513     {
1514         $a = $element-> docblock-> getKeyword('access');
1515         if (is_object($a) && $a-> getString() == 'private') return;
1516     }
1517     if (isset($this-> privatepages[$path]))
1518     {
1519         if (isset($this-> pages[$path]))
1520         {
1521             $uses = $element-> docblock-> getKeyword('uses');

```

```

1521     if ($uses)
1522     {
1523         if (!is_array($uses)) $uses = array($uses);
1524         foreach($uses as $use)
1525         {
1526             if (!is_object($use)) continue;
1527             $el = $use-> getSeeElement();
1528             $description = $use-> getDescription();
1529             $this-> uses[$el][] = array($element, $description);
1530         }
1531     }
1532 }
1533 {
1534     if (isset($this-> pages[$path]))
1535     {
1536         $uses = $element-> docblock-> getKeyword('uses');
1537         if ($uses)
1538         {
1539             if (!is_array($uses)) $uses = array($uses);
1540             foreach($uses as $use)
1541             {
1542                 if (!is_object($use)) continue;
1543                 $el = $use-> getSeeElement();
1544                 $description = $use-> getDescription();
1545                 $this-> uses[$el][] = array($element, $description);
1546             }
1547         }
1548     }
1549 }
1550 }
1551 }
1552 /**
1553 * Add a {@link parserUsedByTag} link to every element referred to by @uses
1554 * @param Converter temporary converter used to retrieve abstract links
1555 * @uses phpDocumentor_IntermediateParser::addUses() indirectly, as
1556 *       addUses() sets up $uses, which is iterated over here
1557 * @uses $pages sets up all @usedby tags from here
1558 * @access private
1559 */
1560 function _setupUsesList(& $converter)
1561 {
1562     ob_start();
1563     $converter-> _createPkgElements($this-> pages);
1564     ob_end_clean();
1565     ksort($this-> uses);
1566     foreach($this-> uses as $link => $elements)
1567     {
1568         foreach($elements as $element)
1569         {
1570             if ($element[0]-> type == 'method' || $element[0]-> type == 'var' ||
1571                 $element[0]-> type == 'const')
1572             {
1573                 $converter-> class = $element[0]-> getClass();
1574             }
1575             if ($element[0]-> type == 'class')
1576             {
1577                 $converter-> class = $element[0]-> getName();
1578             }
1579             $reallink = $converter-> getLink($link, $element[0]-> docblock-> package);
1580             if (is_object($reallink))
1581             {
1582                 // add a used by tag to the docblock of the destination
1583                 switch(phiDocumentor_get_class($reallink))
1584                 {
1585                     case 'pagelink' :
1586                         $this-> pages[$reallink-> path]-> docblock-> addUsedBy(
1587                             $element[0]-> getLink($converter, false, true),
1588                             $element[1]);
1589                         break;
1590                     case 'functionlink' :
1591                     case 'definelink' :
1592                     case 'globallink' :
1593                         if (isset($this-> pages[$reallink-> path]))
1594                         {
1595                             for ($i=0;
1596                                 $i< count($this-> pages[$reallink-> path]-> elements);
1597                                 $i++) {
1598                                 if ($this-> pages[$reallink-> path]-> elements[$i]-
1599 > type ==

```

```

1600
1601
1602
1603
1604
1605
1606
1607
1608
1609 // .str_replace('link','',$phpDocumentor->get_class($reallink)) &&
1610 str_replace('link', '',
1611     $this-> pages[$reallink-> path]->
1612     elements[$i]-> getName()
1613     == $reallink-> name) {
1614     $this-> pages[$reallink-> path]-> elements[$i]->
1615     docblock-> addUsedBy(
1616         $element[0]-> getLink($converter, false, true),
1617         $element[1]);
1618     debug('added @usedby to
1619     '.str_replace('link','',$phpDocumentor->get_class($reallink)).' '.$reallink->name);
1620     }
1621     }
1622     break;
1623     case 'classlink' :
1624     case 'methodlink' :
1625     case 'varlink' :
1626     case 'constlink' :
1627     if (isset($this-> pages[$reallink-> path]))
1628     {
1629         for ($i=0;$i< count($this-> pages[$reallink-> path])->
1630 > classelements);$i++)
1631         {
1632             if ($this-> pages[$reallink-> path]-> classelements[$i]->
1633                 str_replace('link','',$phpDocumentor->get_class($reallink))
1634                 &&
1635                 $this-> pages[$reallink-> path]-> classelements[$i]->
1636                 !isset($reallink-> class) ||
1637                 $this-> pages[$reallink-> path]-> classelements[$i]->
1638                 getClass() == $reallink-> class))
1639                 {
1640                     $this-> pages[$reallink-> path]-> classelements[$i]->
1641                     docblock-> addUsedBy($element[0]-> getLink($converter, false, true), $element[1]);
1642                     debug('added @usedby to
1643                     '.str_replace('link','',$phpDocumentor->get_class($reallink)).' '.$reallink->name);
1644                     }
1645                     }
1646                     break;
1647                 }
1648             }
1649         }
1650     }
1651 /**
1652 * Interface to the Converter
1653 *
1654 * This function simply passes {@link $pages} and {@link package_pages} to
1655 * the walk() method, and then calls the Output() method. Note that
1656 * Output() is not required to do anything, and in fact doesn't in
1657 * HTMLframesConverter.
1658 * @uses Converter::walk() passes {@link $pages} and {@link $package_pages}
1659 * @uses Converter::Output()
1660 */
1661 function Convert($title, $converter)
1662 {
1663     $converter-> walk($this-> pages, $this-> package_pages);
1664     $converter-> Output($title);
1665     $converter-> cleanup();
1666 }
1667 /**
1668 * Clean up classes
1669 *
1670 * {@source}
1671 * @access private
1672 * @uses Classes::Inherit() passes $this
1673 */
1674 function fixClasses()
1675 {
1676     $this-> classes-> Inherit($this);
1677 }
1678 /**
1679 * Clean up Procedural Pages
1680 * {@source}

```

```

1672     * @access private
1673     * @uses ProceduralPages::setupPages() passes $this
1674     */
1675     function fixProcPages()
1676     {
1677         $this-> proceduralpages-> setupPages($this);
1678     }
1679
1680 /**
1681 * If the parent class of $class is in a different package, adds it to the
1682 * {@link $package_parents} array
1683 * @param parserClass &$class
1684 */
1685 function addPackageParent(& $class)
1686 {
1687     if (!is_array($class-> parent)) return;
1688     $par = $this-> classes-> getClass($class-> parent[1], $class-> parent[0]);
1689     if ($class-> docblock-> package == $par-> docblock-> package) return;
1690     $this-> package_parents[$class-> docblock-> package] = $par-> docblock-
> package;
1691     if (!isset($this-> package_parents[$par-> docblock-> package]) || !$this-
> package_parents[$par-> docblock-> package]) $this-> package_parents[$par-> docblock-
> package] = false;
1692 }
1693
1694 /**
1695 * Add a converter name to use to the list of converters
1696 *
1697 * Sets up the {@link $converters} array.
1698 * {@internal
1699 * First, the Converter's file is included, and then, if successful,
1700 * the converter classname is tested for existance. If all is good,
1701 * then the templates are added to the list of converters/templates to use}}
1702 * @param string $output output format (HTML, PDF, XML). Must be all caps
1703 * @param string $name Converter name (frames, for example, is the name of
1704 *                     HTMLframesConverter)
1705 * @param string $template template to use, should be a relative path to the
1706 *                         templates dir (like DOM/default)
1707 */
1708 function addConverter($output,$name,$template)
1709 {
1710     if ($this-> templateBase) {
1711         $templateBase = str_replace('\\','/', $this-> templateBase) . '/Converters';
1712     } else {
1713         if ('@PEAR-DIR@' != '@'. 'PEAR-DIR@') {
1714             $templateBase = 'PhpDocumentor/phpDocumentor/Converters';
1715         } else {
1716             $templateBase = str_replace('\\','/',$GLOBALS['_phpDocumentor_install_dir']) .
1717             '/phpDocumentor/Converters';
1718         }
1719         if (strpos($name,PATH_DELIMITER))
1720         {
1721             // include the parent template
1722             $parent = explode(PATH_DELIMITER,$name);
1723             $parent = $parent[0];
1724             if (!class_exists($output . $parent . 'Converter')) {
1725                 $filename = $templateBase . '/' . $output . '/' . $parent . '/' . $output
1726                 . $parent . 'Converter.inc';
1727                 if ($o::isIncludeable($filename))
1728                 {
1729                     include_once($filename);
1730                 }
1731             }
1732             if (!class_exists($output . $parent . 'Converter'))
1733             {
1734                 addError(PDERROR_CONVERTER_NOT_FOUND,"parent Converter " . $output .
1735                 $parent . "Converter or child Converter " . $output . str_replace(PATH_DELIMITER,'',$name) .
1736                 "Converter");
1737             }
1738             $filename = $templateBase .
1739             PATH_DELIMITER . $output . PATH_DELIMITER . $name . PATH_DELIMITER . $output .
1740             str_replace(PATH_DELIMITER, '', $name) . "Converter" . ".inc";
1741             if ($o::isIncludeable($filename))
1742             {
1743                 include_once($filename);
1744             }
1745             if (class_exists($output . str_replace(PATH_DELIMITER,'',$name) . 'Converter'))
1746             {

```

```

1746      $this-> converters[$output][$output . str_replace(PATH_DELIMITER, '', $name) . .
1747      "Converter" ][] = $template;
1748    } else
1749    {
1750      addError(PDERROR_CONVERTER_NOT_FOUND,$output . str_replace(PATH_DELIMITER, '', $name)
1751      . "Converter" );
1752    }
1753  /**
1754   * does a natural case sort on two {@link parserElement} descendants
1755   *
1756   * @param mixed $a
1757   * @param mixed $b
1758   * @return int
1759   * @see generateElementIndex()
1760   */
1761   function elementCmp ($a, $b)
1762   {
1763     return strnatcasecmp($a-> getName(), $b-> getName());
1764   }
1765 /**
1766  * does a natural case sort on two class elements (either
1767  * {@link parserClass}, parserMethod} or {@link parserVar}
1768  *
1769  * @param mixed $a
1770  * @param mixed $b
1771  * @return int
1772  * @see generateElementIndex()
1773  */
1774   function ClasselementCmp ($a, $b)
1775   {
1776     if (phpDocumentor_get_class($a) == 'parserclass') $atest = $a-> name; else $atest =
1777     $a-> class;
1778     if (phpDocumentor_get_class($b) == 'parserclass') $btest = $b-> name; else $btest =
1779     $b-> class;
1780     if((&$c = strnatcasecmp($atest, $btest)) != 0) return $c;
1781     if (phpDocumentor_get_class($a) != 'parserclass') $atest .= $a-> name;
1782     if (phpDocumentor_get_class($b) != 'parserclass') $btest .= $b-> name;
1783     if (phpDocumentor_get_class($a) == 'parsermethod' &&
1784     phpDocumentor_get_class($b) == 'parsermethod')
1785     {
1786       if ($a-> isConstructor) return -1;
1787       if ($b-> isConstructor) return 1;
1788       if ($a-> isDestructor) return -1;
1789       if ($b-> isDestructor) return 1;
1790     }
1791     return strnatcasecmp($atest,$btest);
1792   }
1793 /**
1794  * call this method once parsing has completed.
1795  *
1796  * This method calls the private methods fixClasses and fixProcPages, both
1797  * of which adjust inheritance and package information based on complicated
1798  * post-parsing rules described in {@link ProceduralPages::setupPages()}
1799  * and {@link Classes::Inherit()}. Then, it sorts elements of the $pages
1800  * array and calls Convert for each Converter in the $converters array
1801  * @see $converters
1802  * @see $pages
1803  * @see Convert()
1804  */
1805   function Output ($title = "Generated Documentation" )
1806   {
1807     $GLOBALS['phpDocumentor_errors']-> curfile = false;
1808     $this-> fixClasses();
1809     $this-> fixProcPages();
1810 //    var_dump($this->uses);
1811 //    exit;
1812     phpDocumentor_out("\nSorting page elements..." );
1813     flush();
1814     uasort($this-> pages,'pagesort');
1815     foreach($this-> pages as $i => $page)
1816     {
1817       usort($this-> pages[$i]-> elements,array($this,'elementCmp'));
1818       usort($this-> pages[$i]-> classelements,array($this,'ClasselementCmp'));
1819     }
1820     phpDocumentor_out("done\n");
1821   }

```

```

1821     flush();
1822     $complicatedout = false;
1823     if (is_array($this-> converters))
1824     {
1825         if (count($this-> converters) > 1)
1826         {
1827             $complicatedout = true;
1828         }
1829         phpDocumentor_out("Formatting @uses list..." );
1830         flush();
1831         $a = new __dummyConverter($this-> all_packages, $this-> package_parents, $this-
> classes, $this-> proceduralpages, $this-> packageoutput, $this-> parsePrivate, $this-
> quietMode, $this-> targetDir , '', $this-> title);
1832         $this-> _setupUsesList($a);
1833         unset($a);
1834         phpDocumentor_out("done\n\n");
1835         flush();
1836         foreach($this-> converters as $converter => $blah)
1837         {
1838             if (is_array($blah))
1839             {
1840                 if (count($blah) > 1)
1841                 {
1842                     $complicatedout = true;
1843                 }
1844                 foreach($blah as $converter => $templates)
1845                 {
1846                     foreach($templates as $template)
1847                     {
1848                         $extraout = '';
1849                         if ($complicatedout)
1850                         {
1851                             $extraout = SMART_PATH_DELIMITER . $converter;
1852                         }
1853                         if (count($templates) > 1)
1854                         {
1855                             $extraout .= SMART_PATH_DELIMITER . str_replace(PATH_DELIMITER,
SMART_PATH_DELIMITER, substr($template,0,strlen($template) - 1));
1856                         }
1857                         $a = new $converter($this-> all_packages, $this-
> package_parents, $this-> classes, $this-> proceduralpages, $this-> packageoutput, $this-
> parsePrivate, $this-> quietMode, $this-> targetDir . $extraout, $template, $this-> title);
1858                         if (isset($this-> templateBase))
1859                         {
1860                             $a-> setTemplateBase($this-> templateBase, $template);
1861                         }
1862                         $a-> ric = $this-> ric;
1863                         $a-> packagecategories = $this-> packagecategories;
1864                         if (isset($this-> tutorials)) $a-> setTutorials($this-
> tutorials);
1865                         $this-> Convert($title, $a);
1866                         unset($a);
1867                     }
1868                 }
1869             }
1870         }
1871     } else
1872     {
1873         addErrorDie(PDERROR_NO_CONVERTERS);
1874     }
1875 }
1876 /**
1877 * Sets the output directory
1878 *
1879 * @param string $dir the output directory
1880 */
1881 function setTargetDir($dir)
1882 {
1883     $this-> targetDir = $dir;
1884 }
1885
1886 /**
1887 * Sets the template base directory
1888 *
1889 * @param string $dir the template base directory
1890 * @tutorial phpDocumentor.howto.pkg#using.command-line.templatebase
1891 */
1892 function setTemplateBase($dir)
1893 {

```

```

1895     $this-> templateBase = $dir;
1896 }
1897
1898 /**
1899 * set parsing information output mode (quiet or verbose)
1900 *
1901 * If set to false, no parsing information (parsing /php/file/thisfile.php,
1902 * Converting etc.) will be displayed.
1903 * Useful for cron jobs
1904 * @param bool $quietMode
1905 */
1906 function setQuietMode($quietMode)
1907 {
1908     $this-> quietMode = $quietMode;
1909 }
1910
1911 /**
1912 * show warnings for undocumented elements
1913 *
1914 * If set to false, no warnings will be shown for undocumented elements.
1915 * Useful for identifying classes and methods that haven't yet been documented.
1916 * @param bool $undocumentedElementWarnings
1917 */
1918 function setUndocumentedElementWarningsMode($undocumentedElementWarnings)
1919 {
1920     $this-> undocumentedElementWarnings = $undocumentedElementWarnings;
1921 }
1922
1923 /**
1924 * set display of elements marked with @access private
1925 *
1926 * If set to true, elements will be displayed
1927 * @param bool $parse
1928 */
1929 function setParsePrivate($parse)
1930 {
1931     $this-> parsePrivate = $parse;
1932 }
1933 }
1934
1935 /** @access private */
1936 function pagesort($a, $b)
1937 {
1938     return strnatcasecmp($a-> parent-> file,$b-> parent-> file);
1939 }
1940 ?>

```

# File Source for Io.inc

Documentation for this file is available at [Io.inc](#)

```
1  <?php
2  /**
3  * File and input handling routines
4  *
5  * This class parses command-line options, and works with files to
6  * generate lists of files to parse based on the ignore/include options
7  *
8  * phpDocumentor :: automatic documentation generator
9  *
10 * PHP versions 4 and 5
11 *
12 * Copyright (c) 2000-2006 Joshua Eichorn, Gregory Beaver
13 *
14 * LICENSE:
15 *
16 * This library is free software; you can redistribute it
17 * and/or modify it under the terms of the GNU Lesser General
18 * Public License as published by the Free Software Foundation;
19 * either version 2.1 of the License, or (at your option) any
20 * later version.
21 *
22 * This library is distributed in the hope that it will be useful,
23 * but WITHOUT ANY WARRANTY; without even the implied warranty of
24 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
25 * Lesser General Public License for more details.
26 *
27 * You should have received a copy of the GNU Lesser General Public
28 * License along with this library; if not, write to the Free Software
29 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
30 *
31 * @package    phpDocumentor
32 * @author     Joshua Eichorn <jeichorn@phpdoc.org>
33 * @author     Gregory Beaver <cellog@php.net>
34 * @copyright  2000-2006 Joshua Eichorn, Gregory Beaver
35 * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
36 * @version    CVS: $Id: Io.inc 286921 2009-08-08 05:01:24Z ashnazg $
37 * @filesource
38 * @link       http://www.phpdoc.org
39 * @link       http://pear.php.net/PhpDocumentor
40 * @since      0.1
41 */
42 /**
43 * Class to handle file and user io operations
44 *
45 * @author     Joshua Eichorn <jeichorn@phpdoc.org>
46 * @author     Gregory Beaver <cellog@php.net>
47 * @version    $Id: Io.inc 286921 2009-08-08 05:01:24Z ashnazg $
48 * @package    phpDocumentor
49 */
50 class Io
51 {
52
53     /**
54     * Holds all the options that are available to the cmd line interface
55     * and to the different web interfaces
56     */
57     var $phpDocOptions;
58     /**
59     * Format: array(array(regexp-ready string to search for whole path,
60     * regexp-ready string to search for basename of ignore strings),...)
61     * @var false|array
62     */
63     var $ignore;
64     /**
65     * A specific array of values that boolean-based arguments can understand,
66     * aided by the {@link decideOnOrOff()} helper method.
67     *
```

```

68     * Use lowercase letters always, to simplify string comparisons
69     * @var array
70     */
71 var $valid_booleans = array
72 (
73     '' , ' ', 'on' , 'y' , 'yes' , 'true' , '1' ,
74     'off' , 'n' , 'no' , 'false' , '0'
75
76 );
77 /**
78     * creates an array $this->phpDocOptions and sets program options in it.
79     * Array is in the format of:
80     * <pre>
81     * [filename][tag][] = "f";
82     * [filename][tag][] = "-file";
83     * [filename][desc] "name of file to parse"
84     * </pre>
85     */
86
87 function Io()
88 {
89     $this-> phpDocOptions['filename'][tag] = array( "-f" , "--"
filename" );
90     $this-> phpDocOptions['filename'][desc] = "name of file(s) to parse , "
file1,file2.
Can contain complete path and * ? wildcards" ;
91     $this-> phpDocOptions['filename'][type] = "path" ;
92
93     $this-> phpDocOptions['directory'][tag] = array( "-d" , "--"
directory" );
94     $this-> phpDocOptions['directory'][desc] = "name of a directory(s) to parse
directory1,directory2" ;
95     $this-> phpDocOptions['directory'][type] = "path" ;
96
97     $this-> phpDocOptions['examplesdir'][tag] = array( "-ed" , "--"
examplesdir" );
98     $this-> phpDocOptions['examplesdir'][desc] = "full path of the directory to
look for example files from @example tags" ;
99     $this-> phpDocOptions['examplesdir'][type] = "path" ;
100
101    $this-> phpDocOptions['templatebase'][tag] = array( "-tb" , "--"
templatebase" );
102    $this-> phpDocOptions['templatebase'][desc] = "base location of all templates
for this parse." ;
103    $this-> phpDocOptions['templatebase'][type] = "path" ;
104
105    $this-> phpDocOptions['target'][tag] = array( "-t" , "--target" );
106    $this-> phpDocOptions['target'][desc] = "path where to save the generated
files" ;
107    $this-> phpDocOptions['target'][type] = "path" ;
108
109    $this-> phpDocOptions['ignore'][tag] = array( "-i" , "--ignore" );
110    $this-> phpDocOptions['ignore'][desc] = "file(s) that will be ignored,
multiple separated by , . Wildcards * and ? are ok" ;
111    $this-> phpDocOptions['ignore'][type] = "path" ;
112
113    $this-> phpDocOptions['ignoresymlinks'][tag] = array( "-is" , "--"
ignoresymlinks" );
114    $this-> phpDocOptions['ignoresymlinks'][desc] = "ignore symlinks to other
files or directories, default is off" ;
115    $this-> phpDocOptions['ignoresymlinks'][type] = "set" ;
116    $this-> phpDocOptions['ignoresymlinks'][validvalues] = $this-> valid_booleans;
117
118    $this-> phpDocOptions['ignoretags'][tag] = array( "-it" , "--ignore-
tags" );
119    $this-> phpDocOptions['ignoretags'][desc] = "tags to ignore for this parse.
@package, @subpackage, @access and @ignore may not be ignored." ;
120    $this-> phpDocOptions['ignoretags'][type] = "value" ;
121
122    $this-> phpDocOptions['hidden'][tag] = array( "-dh" , "--hidden" );
123    $this-> phpDocOptions['hidden'][desc] = "set equal to on (-dh on) to descend
into hidden directories (directories starting with '.'), default is off" ;
124    $this-> phpDocOptions['hidden'][type] = "set" ;
125    $this-> phpDocOptions['hidden'][validvalues] = $this-> valid_booleans;
126
127    $this-> phpDocOptions['quiet'][tag] = array( "-q" , "--quiet" );
128    $this-> phpDocOptions['quiet'][desc] = "do not display parsing/conversion
messages. Useful for cron jobs on/off default off" ;
129    $this-> phpDocOptions['quiet'][type] = "set" ;
130    $this-> phpDocOptions['quiet'][validvalues] = $this-> valid_booleans;
131

```

```

132     $this-> phpDocOptions['undocumentedelements']['tag'] = array("-ue" , "--"
133     undocumentedelements");
133     $this-> phpDocOptions['undocumentedelements']['desc'] = "Control whether or not
134     warnings will be shown for undocumented elements. Useful for identifying classes and methods that
135     haven't yet been documented on/off default off";
136     $this-> phpDocOptions['undocumentedelements']['type'] = "set" ;
135     $this-> phpDocOptions['undocumentedelements']['validvalues'] = $this-
136 > valid_booleans;
136
137     $this-> phpDocOptions['title']['tag'] = array("-ti" , "--title" );
138     $this-> phpDocOptions['title']['desc'] = "title of generated documentation,
139     default is 'Generated Documentation'";
139     $this-> phpDocOptions['title']['type'] = "value" ;
140
141     $this-> phpDocOptions['help']['tag'] = array("-h" , "--help" );
142     $this-> phpDocOptions['help']['desc'] = " show this help message";
143
144     $this-> phpDocOptions['useconfig']['tag'] = array("-c" , "--"
144 useconfig");
145     $this-> phpDocOptions['useconfig']['desc'] = "Use a Config file in the users/
145 subdirectory for all command-line options";
146     $this-> phpDocOptions['useconfig']['type'] = "value" ;
147
148     $this-> phpDocOptions['parseprivate']['tag'] = array("-pp" , "--"
148 parseprivate");
149     $this-> phpDocOptions['parseprivate']['desc'] = "parse @internal and elements
149 marked private with @access. Use on/off, default off";
150     $this-> phpDocOptions['parseprivate']['type'] = "set" ;
151     $this-> phpDocOptions['parseprivate']['validvalues'] = array('on' , 'off');
152
153     $this-> phpDocOptions['packageoutput']['tag'] = array("-po" , "--"
153 packageoutput");
154     $this-> phpDocOptions['packageoutput']['desc'] = "output documentation only for
154 selected packages. Use a comma-delimited list";
155     $this-> phpDocOptions['packageoutput']['type'] = "value" ;
156
157     $this-> phpDocOptions['defaultpackagename']['tag'] = array("-dn" , "--"
157 defaultpackagename");
158     $this-> phpDocOptions['defaultpackagename']['desc'] = "name to use for the
158 default package. If not specified, uses 'default'";
159     $this-> phpDocOptions['defaultpackagename']['type'] = "value" ;
160
161     $this-> phpDocOptions['defaultcategoryname']['tag'] = array("-dc" , "--"
161 defaultcategoryname");
162     $this-> phpDocOptions['defaultcategoryname']['desc'] = "name to use for the
162 default category. If not specified, uses 'default'";
163     $this-> phpDocOptions['defaultcategoryname']['type'] = "value" ;
164
165     $this-> phpDocOptions['output']['tag'] = array("-o" , "--output" );
166     $this-> phpDocOptions['output']['desc'] = "output information to use separated
166 by ','. Format: output:converter:templatedir like \"HTML:frames:phpedit\"";
167     $this-> phpDocOptions['output']['type'] = "value" ;
168
169     $this-> phpDocOptions['converterparams']['tag'] = array("-cp" , "--"
169 converterparams");
170     $this-> phpDocOptions['converterparams']['desc'] = "dynamic parameters for a
170 converter, separate values with commas";
171     $this-> phpDocOptions['converterparams']['type'] = "value" ;
172
173     $this-> phpDocOptions['customtags']['tag'] = array("-ct" , "--"
173 customtags");
174     $this-> phpDocOptions['customtags']['desc'] = "custom tags, will be recognized
174 and put in tags[] instead of unknowntags[]";
175     $this-> phpDocOptions['customtags']['type'] = "value" ;
176
177     $this-> phpDocOptions['sourcecode']['tag'] = array("-s" , "--"
177 sourcecode");
178     $this-> phpDocOptions['sourcecode']['desc'] = "generate highlighted sourcecode
178 for every parsed file (PHP 4.3.0+ only) on/off default off";
179     $this-> phpDocOptions['sourcecode']['type'] = "set" ;
180     $this-> phpDocOptions['sourcecode']['validvalues'] = array('on' , 'off');
181
182     $this-> phpDocOptions['javadocdesc']['tag'] = array("-j" , "--"
182 javadocdesc");
183     $this-> phpDocOptions['javadocdesc']['desc'] = "JavaDoc-compliant description
183 parsing. Use on/off, default off (more flexibility)";
184     $this-> phpDocOptions['javadocdesc']['type'] = "set" ;
185     $this-> phpDocOptions['javadocdesc']['validvalues'] = array('on' , 'off');
186
187     $this-> phpDocOptions['pear']['tag'] = array("-p" , "--pear" );

```

```

188      $this-> phpDocOptions['pear']['desc'] = "Parse a PEAR-style repository (package
is directory, _members are @access private) on/off default off" ;
189      $this-> phpDocOptions['pear']['type'] = "set" ;
190      $this-> phpDocOptions['pear']['validvalues'] = array('on', 'off') ;
191
192      $this-> phpDocOptions['readmeinstallchangelog']['tag'] = array("-ric",
193      "-readmeinstallchangelog") ;
194      $this-> phpDocOptions['readmeinstallchangelog']['desc'] = "Specify custom
filenames to parse like README, INSTALL or CHANGELOG files" ;
195      $this-> phpDocOptions['readmeinstallchangelog']['type'] = "value" ;
196
197      $this-> phpDocOptions['general']['message'] = "You can have multiple directories
and multiple files, as well as a combination of both options" ;
198
199
200 /**
201 * create the help message for display on the command-line
202 * @return string a string containing a help message
203 */
204 function displayHelpMsg()
205 {
206     unset($ret);
207     $ret = "\n";
208     foreach($this-> phpDocOptions as $data)
209     {
210         unset($tag);
211         $tag = "" ;
212         if (isset($data['tag']))
213         {
214             if (is_array($data['tag'])) {
215                 foreach($data['tag'] as $param) {
216                     $tag .= "    $param" ;
217                 }
218             }
219             $taglen = 34;
220             $outputwidth = 79;
221             $tagspace = str_repeat(" ", $taglen);
222             $tmp = " " . trim($tag).$tagspace;
223             $tmp = substr($tmp,0,$taglen);
224             $d = wordwrap(trim($data['desc']),($outputwidth-$taglen));
225             $dt = explode("\n", $d);
226             $dt[0] = $tmp . $dt[0];
227             for($i=1;$i< count($dt);$i++)
228             {
229                 $dt[$i] = $tagspace.$dt[$i];
230             }
231             $ret .= implode("\n", $dt). "\n\n" ;
232
233         }
234     }
235     $ret .= "\n" . wordwrap($data['message'],$outputwidth). "\n" ;
236     return $ret;
237 }
238
239 /**
240 * calls {@link file_exists()} for each value in include_path,
241 * then calls {@link is_readable()} when it finds the file
242 * @param string
243 * @return boolean
244 */
245 function isIncludeable($filename)
246 {
247     $test = realpath($filename);
248     if ($test && is_readable($test)) {
249         return true; // for absolute paths
250     }
251     $ip = get_include_path();
252     if (PHPDOCUMENTOR_WINDOWS)
253     {
254         $ip = explode(';', $ip);
255     } else {
256         $ip = explode(':', $ip);
257     }
258     foreach($ip as $path)
259     {
260         if ($a = realpath($path . DIRECTORY_SEPARATOR . $filename))
261         {
262             if (is_readable($a))
263         }

```

```

264             return true;
265         }
266     }
267 }
268 return false;
269 }
270
271 /**
272 * Parses $_SERVER['argv'] and creates a setup array
273 * @return array a setup array
274 * @global array command-line arguments
275 * @todo replace with Console_*
276 */
277 function parseArgv()
278 {
279     global $argv;
280
281     // defaults for setting
282     $setting['hidden'] = "off";
283     $setting['ignoresymlinks'] = 'off';
284     $setting['template'] = 'templates' . PATH_DELIMITER . 'default' . PATH_DELIMITER;
285
286     $valnext = "junk";
287     $data = array();
288     if(isset($argv) && is_array($argv))
289     {
290         foreach ($argv as $cmd)
291         {
292             if ($cmd == '--') {
293                 continue;
294             }
295             if ($cmd == '-h' || $cmd == '--help')
296             {
297                 echo $this-> displayHelpMsg();
298                 die();
299             }
300
301             // at first, set the arg value as if we
302             // already know it's formatted normally, e.g.
303             // -q on
304             $setting[$valnext] = $cmd;
305
306             if (isset($data['type']) && $data['type'] == 'set') {
307
308                 if ($valnext !== 'junk' && strpos(trim($cmd), '-') === 0) {
309                     // if valnext isn't 'junk' (i.e it was an arg option)
310                     // then the first arg needs an implicit "" as its value, e.g.
311                     // ... -q -pp ... ==> ... -q '' -pp ...
312                     $setting[$valnext] = '';
313
314                 } else if (!in_array(strtolower($cmd), $data['validvalues'], true)) {
315                     // the arg value is not a valid value
316                     addErrorDie(PDERROR_INVALID_VALUES, $valnext, $cmd,
317                               '(' . implode(', ', $data['validvalues']) . ')');
318                 }
319             }
320
321             foreach( $this-> phpDocOptions as $name => $data )
322             {
323                 if (!empty($data['tag']))
324                 {
325                     if (in_array($cmd,$data['tag']))
326                     {
327                         $valnext = $name;
328                         break;
329                     }
330                     else
331                     {
332                         $valnext = "junk";
333                     }
334                 }
335
336                 if ($valnext == 'junk' && (strpos(trim($cmd), '-') === 0)) {
337                     // this indicates the last arg of the command
338                     // is an arg option (-) that was preceded by unrecognized "junk"
339                     addErrorDie(PDERROR_UNKNOWN_COMMANDLINE,$cmd);
340
341                 } else if ($valnext != 'junk' && (strpos(trim($cmd), '-') === 0)) {
342                     // this indicates the last arg of the command

```

```

344                         // is an arg option (-) without an arg value
345
346                         // add an empty arg "value" for this arg "option"
347                         $setting[$valnext] = '';
348
349
350                         }
351             } else
352             {
353                 echo "Please use php-cli.exe in windows, or set register_argc_argv On";
354                 die;
355             }
356
357             /* $setting will always have at least 3 elements
358             [$hidden] => off
359             [ignoresymlinks] => 'off'
360             [template] => templates/default
361             */
362             if (count($setting) < 4) {
363                 echo $this-> displayhelpMsg();
364                 die();
365             }
366
367             return $setting;
368         }
369
370
371         /**
372          * @return array list of files in a directory
373          * @param string $directory full path to the directory you want the list of
374          * @param bool whether to list files that begin with . like .bash_history
375          * @param bool whether to ignore symlinks
376          */
377         function dirList($orig_directory, $hidden = false, $ignore_symlinks = false)
378     {
379         $directory = realpath($orig_directory);
380         $ret = false;
381         if (!@is_dir($directory))
382         {
383             die("      directory: '$directory'  not found\n");
384         }
385         $ret = array();
386         $d = @dir($directory); // thanks to Jason E Sweat (jsweat@users.sourceforge.net) for fix
387         while($d && ($entry=$d-> read()) != false) {
388             if (strcmp($entry, ".") == 0 || strcmp($entry, "..") == 0) {
389                 continue;
390             }
391
392             // skip hidden files, if we're supposed to
393             if (!$hidden)
394             {
395                 if (substr($entry,0,1) == ".")
396                 {
397                     phpDocumentor_out("Hidden " . $directory . PATH_DELIMITER .
398 $entry . " Ignored\n");
399                     continue;
400                 }
401
402             // skip symlink files, if we're supposed to
403             if ($ignore_symlinks)
404             {
405                 if (is_link($directory . PATH_DELIMITER . $entry))
406                 {
407                     phpDocumentor_out("Symlink " . $directory . PATH_DELIMITER .
408 $entry . " Ignored\n");
409                     continue;
410                 }
411
412                 if (is_file($directory . PATH_DELIMITER . $entry)) {
413                     $ret[] = $directory . PATH_DELIMITER . $entry;
414                 }
415                 if (is_dir($directory . PATH_DELIMITER . $entry)) {
416                     $tmp = $this-> dirList($directory . PATH_DELIMITER . $entry, $hidden,
417 $ignore_symlinks);
418                     if (is_array($tmp)) {
419                         foreach($tmp as $ent) {
420                             $ret[] = $ent;
421                         }
422                     }
423                 }
424             }
425         }
426     }

```

```

421         }
422     }
423 }
424 if ($d) $d-> close();
425 return $ret;
426 }
427
428 /**
429 * Retrieve common directory (case-insensitive in windows)
430 *
431 * takes the list of files, and returns the subdirectory they share in common,
432 * so in this list:
433 *
434 * <code>
435 * array(
436 *   "/dir1/dir2/subdir/dir3/filename.ext",
437 *   "/dir1/dir2/subdir/dir4/filename.ext",
438 *   "/dir1/dir2/mydir/dir5/filename.ext");
439 * </code>
440 *
441 * getBase will return "/dir1/dir2"
442 * @param array array of strings
443 */
444 function getBase($filelist)
445 {
446     $masterPath = false;
447     foreach($filelist as $path)
448     {
449         if (!$masterPath)
450         {
451             $masterPath = str_replace('\\', '/', dirname($path));
452         } else
453         {
454             if (dirname($path) != $masterPath)
455             {
456                 $mp = explode(PATH_DELIMITER, $masterPath);
457                 $np = explode(PATH_DELIMITER, str_replace('\\', '/', dirname($path)));
458                 if (count($np) < count($mp))
459                 {
460                     $masterPath = join($np, PATH_DELIMITER);
461                 } else
462                 {
463                     $test = false;
464                     $found = false;
465                     for($i=0;$i < count($mp) && $i < count($np);$i++)
466                     {
467                         if (PHPDOCUMENTOR_WINDOWS)
468                         {
469                             if (strtolower($mp[$i]) != strtolower($np[$i])) $found = $i;
470                         } else
471                         {
472                             if ($mp[$i] != $np[$i]) $found = $i;
473                         }
474                     }
475                     if ($found !== false)
476                     {
477                         $mp = array_slice($mp, 0,$found);
478                         $masterPath = join($mp,PATH_DELIMITER);
479                     }
480                 }
481             }
482         }
483     }
484     return $masterPath;
485 }
486
487 /**
488 * Retrieve tutorial subdirectories and their contents from the list of
489 * files to parse
490 * @param array array of paths (strings)
491 * @return array array(filelist - tutorials, tutorials)
492 */
493 function getTutorials($filelist)
494 {
495     $list = $tutorials = array();
496     foreach($filelist as $file)
497     {
498         if (strpos($file,'tutorials/') !== false)
499         {
500             $tutedir = explode('/', substr($file,strpos($file,'tutorials/')));

```

```

501     array_shift($tutedir);
502     if (count($tutedir) <=      3)
503     {
504         $res = array();
505         // kludge - will need to fix for 2.0
506         $res['category'] = $GLOBALS['_phpDocumentor_DefaultCategoryName'];
507         $res['package'] = array_shift($tutedir);
508         $res['subpackage'] = '';
509         if (count($tutedir) >      1)
510             $res['subpackage'] = array_shift($tutedir);
511         $f = array_shift($tutedir);
512         $f = $f[0];
513         $f = explode('.',$f);
514         $res['tutename'] = $f[0];
515         $res['tutetype'] = array_pop($f);
516         if ($res['tutetype'] == 'ini') continue;
517         $res['path'] = $file;
518         if (@file_exists($file . '.ini'))
519         {
520             $res['ini'] = phpDocumentor_parse_ini_file($file . '.ini', true);
521         } else
522         {
523             $res['ini'] = false;
524         }
525         $tutorials[] = $res;
526     } else $list[] = $file;
527 }
528 return array($list,$tutorials);
529 }
530 /**
531 * @param string base directory from {@link getBase()}
532 * @param array file list from {@link dirList()}
533 * @return array array(filelist - README/INSTALL/CHANGELOG,
534 *                      README/INSTALL/CHANGELOG)
535 */
536 function getReadmeInstallChangelog($base,$filelist)
537 {
538     $list = $ric = array();
539     $names = $GLOBALS['_phpDocumentor_RIC_files'];
540     foreach($filelist as $file)
541     {
542         if ((dirname($file) == $base) && in_array(strtoupper(basename($file)),
543 $names))
544         {
545             $ric[] = $file;
546         } else
547         {
548             $list[] = $file;
549         }
550     }
551     return array($list,$ric);
552 }
553 /**
554 * @param string directory
555 * @param string base directory
556 * @param array array of ignored items
557 * @param boolean the "hidden" flag
558 * @param boolean the "ignoresymlinks" flag
559 * @uses dirList
560 * @uses checkIgnore
561 * @uses setup_dirs
562 */
563 function getDirTree($dir, $base_dir, $ignore = array(), $hidden = false, $ignoresymlinks =
false)
564 {
565     $allfiles = $this-> dirList($dir,$hidden,$ignoresymlinks);
566     $struc = array();
567     foreach($allfiles as $file)
568     {
569         if ($this-
570 > checkIgnore(basename($file),dirname(realpath($file)), $ignore, $ignoresymlinks)) continue;
571         if (PHPDOCUMENTOR_WINDOWS) {
572             $path =
573             substr(dirname($file),strlen(str_replace('\\','/',$base_dir)));
574         } else {
575             $path =
576             substr(dirname($file),strlen(str_replace('\\','/',$base_dir))+1);
577         }

```

```

576     if (!$path) $path = '/';
577     $parts = pathinfo($file);
578     if (!isset($parts['extension']))
579     {
580         $parts['extension'] = '';
581     }
582     $struc[$path][] = array(
583         'file' => $parts['basename'],
584         'ext' => $parts['extension'],
585         'path' => $file);
586 }
587 uksort($struc, 'strnatcasecmp');
588 foreach($struc as $key => $ind)
589 {
590     usort($ind, 'Ioinc_sortfiles');
591     $struc[$key] = $ind;
592     $save = $key;
593     if ($key != '/')
594     {
595         $key = explode('/', $key);
596         while (count($key))
597         {
598             array_pop($key);
599             if (isset($struc[join('/', $key)]))
600             {
601                 $struc[join('/', $key)][substr($save, strlen(join('/', $key)) + 1)] = $ind;
602                 unset($struc[$save]);
603             }
604         }
605     }
606 }
607 foreach($struc as $key => $ind)
608 {
609     if ($key != '/')
610     {
611         if (count(explode('/', $key)) == 1)
612         {
613             $struc['/'][$key] = $struc[$key];
614             unset($struc[$key]);
615         }
616     }
617 }
618 $tempstruc = $struc;
619 unset($tempstruc['/']);
620 $leftover_dirs = array_keys($tempstruc);
621 $splitdirs = array();
622 foreach($leftover_dirs as $dir)
623 {
624     $splitdirs[] = explode('/', $dir);
625 }
626 $leftover_dirs = array();
627 foreach($splitdirs as $dir)
628 {
629     $save = join($dir, '/');
630     $struc['/'] = setup_dirs($struc['/'], $dir, $tempstruc[$save]);
631     unset($struc[$save]);
632 }
633 @uksort($struc['/'], 'Ioinc_mystrucsort');
634 return $struc;
635 }
636 }
637 /**
638 * Reads a file and returns it as a string
639 * Does basic error checking
640 *
641 * file extensions are set in {@link phpdoc.inc}
642 *
643 * @global array PHP File extensions, used to validate that $path is a PHP File
644 * @global array PHP File extensions in a CVS repository, used to validate that $path is a
645 PHP File
646 * @param string $path
647 */
648 function readPhpFile($path, $quietMode = false)
649 {
650     global $_phpDocumentor_cvsphpfile_exts, $_phpDocumentor_phpfile_exts;
651     // tiberiusblue addition
652     $cvsExt = $_phpDocumentor_cvsphpfile_exts;
653     $ext = $_phpDocumentor_phpfile_exts;
654     if (file_exists($path))

```

```

655
656     {
657         if (is_file($path))
658         {
659             // check extension
660             $tmp = explode(".", $path);
661             // tiberiusblue addition
662             $tmp2 = $tmp;
663             if (in_array(array_pop($tmp), $ext))
664             {
665                 phpDocumentor_out(" -- Parsing file\n");
666                 flush();
667                 if (function_exists('file_get_contents')) {
668                     return file_get_contents($path);
669                 }
670                 $fp = fopen($path, "r");
671                 $ret = fread($fp, filesize($path));
672                 fclose($fp);
673                 return $ret;
674             } elseif (in_array(array_pop($tmp2), $cvsExt))
675             {
676                 phpDocumentor_out(" CVS file [EXPERIMENTAL]\n");
677                 flush();
678                 if (function_exists('file_get_contents')) {
679                     $ret = file_get_contents($path);
680                 } else {
681                     $fp = fopen($path, "r");
682                     $ret = fread($fp, filesize($path));
683                     fclose($fp);
684                 }
685                 $ret = strstr($ret, "<?");
686                 $ret = substr($ret, 0, strpos($ret, "@\n"));
687                 $ret = str_replace("@@", "@", $ret);
688                 return $ret;
689             } else
690             {
691                 phpDocumentor_out(" -- File not parsed, not a php file\n");
692                 flush();
693             } else {
694                 phpDocumentor_out(" -- Unable to read file, not a file\n");
695                 flush();
696             }
697         } else {
698             phpDocumentor_out(" -- Unable to read file, file does not exist\n");
699             flush();
700         }
701     }
702
703 /**
704 * Tell whether to ignore a file or a directory
705 * allows * and ? wildcards
706 *
707 * @author Greg Beaver <cellog@php.net>
708 * @param string $file just the file name of the file or directory,
709 *                      in the case of directories this is the last dir
710 * @param string $path the path to consider (should be checked by
711 *                      realpath() before, and may be relative)
712 * @param array $ignore
713 * @param bool $ignore Ignore symlinks?
714 * @return bool true if $path should be ignored, false if it should not
715 */
716 function checkIgnore($file, $path, $ignore, $ignore_no_ext = true, $ignoresymlinks = false)
717 {
718     global $_phpDocumentor_RIC_files;
719
720     if ($ignoresymlinks && is_link($path . PATH_DELIMITER . $file)) return false;
721
722     if (!count($ignore)) return false;
723
724     if (!isset($this-> ignore))
725     {
726         $this-> _setupIgnore($ignore);
727     }
728     if (!$this-> ignore)
729     {
730         return false;
731     }
732
733     if ($ignore_no_ext &&

```

```

735     !in_array(strtoupper($file), $phpDocumentor_RIC_files))
736     {
737         if (!is_numeric(strpos($file,'.'))) return true;
738     }
739     if (is_array($this-> ignore))
740     {
741         foreach($this-> ignore as $match)
742         {
743             // match is an array if the ignore parameter was a /path/to/pattern
744             if (is_array($match))
745             {
746                 // check to see if the path matches with a path delimiter appended
747                 preg_match('/^' . strtoupper($match[0]).'$', strtoupper($path));
748
749                 if (!count($find))
750                 {
751                     // check to see if it matches without an appended path delimiter
752                     preg_match('/^' . strtoupper($match[0]).'$', strtoupper($path), $find);
753                 }
754                 if (count($find))
755                 {
756                     // check to see if the file matches the file portion of the regex string
757                     preg_match('/^' . strtoupper($match[1]).'$', strtoupper($file), $find);
758                     if (count($find))
759                     {
760                         // return true;
761                     }
762                     // check to see if the full path matches the regex
763                     preg_match('/^' . strtoupper($match[0]).'$',
764                             strtoupper($path . DIRECTORY_SEPARATOR . $file), $find);
765                     if (count($find))
766                     {
767                         return true;
768                     }
769                 } else
770                 {
771                     // ignore parameter was just a pattern with no path delimiters
772                     // check it against the path
773                     preg_match('/^' . strtoupper($match).'$', strtoupper($path), $find);
774                     if (count($find))
775                     {
776                         return true;
777                     }
778                     // check it against the file only
779                     preg_match('/^' . strtoupper($match).'$', strtoupper($file), $find);
780                     if (count($find))
781                     {
782                         return true;
783                     }
784                 }
785             }
786         }
787     }
788     return false;
789 }
790 /**
791 * Construct the {@link $ignore} array
792 * @author Greg Beaver <cellog@php.net>
793 * @param array strings of files/paths/wildcards to ignore
794 * @access protected
795 */
796 function _setupIgnore($ignore)
797 {
798     $ig = array();
799     if ( ! is_array($ignore))
800     {
801         $this-> ignore = false;
802         return;
803     }
804     for($i=0; $i< count($ignore);$i++)
805     {
806         if (empty($ignore[$i]))
807             continue;
808
809         $ignore[$i] = strtr($ignore[$i], '\\', '/');
810         $ignore[$i] = str_replace('/', PATH_DELIMITER, $ignore[$i]);
811
812         if (!is_numeric(strpos($ignore[$i],PATH_DELIMITER)))
813         {

```

```

814         $sig[] = $this-> getRegExpableSearchString($ignore[$i]);
815     } else
816     {
817         if (basename($ignore[$i]) . PATH_DELIMITER == $ignore[$i])
818             $sig[] = $this-> getRegExpableSearchString($ignore[$i]);
819         else
820             $sig[] = array($this-> getRegExpableSearchString($ignore[$i]),$this-
> getRegExpableSearchString(basename($ignore[$i])));
821     }
822     if (count($sig)) $this-> ignore = $sig;
823 }
824
825 /**
826  * Converts $s into a string that can be used with preg_match
827  * @param string $s string with wildcards ? and *
828  * @author Greg Beaver <cellog@php.net>
829  * @return string converts * to .*, ? to ., etc.
830  */
831 function getRegExpableSearchString($s)
832 {
833     $y = '/\/';
834     if (DIRECTORY_SEPARATOR == '\\')
835     {
836         $y = '\\\\\\';
837     }
838     $s = str_replace('/', DIRECTORY_SEPARATOR, $s);
839     $x = strtr($s, array('?' => '.', '*' => '.*', '.' => '\\.', '\\' => '\\\\',
840 => '\\\\', '[' => '\\[', ']' => '\\]', '-' => '\\-'));
841     if (strpos($s, DIRECTORY_SEPARATOR) !== false &&
842         strpos($s, DIRECTORY_SEPARATOR) === strlen($s) - 1)
843     {
844         $x = "(?:.*$y$x?.*|$x.*)";
845     }
846     return $x;
847 }
848
849 /**
850  * Removes files from the $dir array that do not match the search string in
851  * $match
852  * @param array $dir array of filenames (full path)
853  * @param string $match search string with wildcards
854  * @author Greg Beaver <cellog@php.net>
855  * @return string/array listing of every file in a directory that matches
856  *          the search string
857  */
858 function removeNonMatches($dir, $match)
859 {
860     $match = $this-> getRegExpableSearchString($match);
861     $nodir = false;
862     if (!is_array($dir))
863     {
864         $dir = array($dir);
865         $nodir = true;
866     }
867     foreach($dir as $i => $file)
868     {
869         preg_match('/^'. $match .' /', basename($file), $find);
870         if (!count($find)) unset($dir[$i]);
871     }
872     if ($nodir) return $dir[0];
873     return $dir;
874 }
875
876 /**
877  * Take a filename with wildcards and return all files that match the
878  * wildcards
879  * @param string $file a full path from the -f command-line parameter, with
880  *          potential * and ? wildcards.
881  * @return mixed if $file contains wildcards, returns an array of matching
882  *          files, otherwise returns false
883  * @author Greg Beaver <cellog@php.net>
884  * @uses dirList
885  * @uses removeNonMatches
886  */
887 function getAllFiles($file)
888 {
889     $path = realpath(dirname($file));
890     $file = basename($file);

```

```

892     // any wildcards?
893     if (is_numeric(strpos($file, '?')) || is_numeric(strpos($file, '*')))
894     {
895         $files = $this-> dirList($path);
896         $a = $this-> removeNonMatches($files,$file);
897         return $a;
898     }
899     return false;
900 }
901 }
902 /**
903 * Sorting functions for the file list
904 * @param string
905 * @param string
906 */
907 function Ioinc_sortfiles($a, $b)
908 {
909     return strnatcasecmp($a['file'], $b['file']);
910 }
911
912 function Ioinc_mystrucsort($a, $b)
913 {
914     if (is_numeric($a) && is_string($b)) return 1;
915     if (is_numeric($b) && is_string($a)) return -1;
916     if (is_numeric($a) && is_numeric($b))
917     {
918         if ($a > $b) return 1;
919         if ($a < $b) return -1;
920         if ($a == $b) return 0;
921     }
922     return strnatcasecmp($a,$b);
923 }
924 /**
925 * Recursively add all the subdirectories of $contents to $dir without erasing anything in
926 * $dir
927 * @param array
928 * @param array
929 * @return array processed $dir
930 */
931 function set_dir($dir,$contents)
932 {
933     while(list($one,$two) = each($contents))
934     {
935         if (isset($dir[$one]))
936         {
937             $dir[$one] = set_dir($dir[$one],$contents[$one]);
938         } else $dir[$one] = $two;
939     }
940     return $dir;
941 }
942 /**
943 * Recursively move contents of $struc into associative array
944 *
945 * The contents of $struc have many indexes like 'dir/subdir/subdir2'.
946 * This function converts them to
947 * array('dir' => array('subdir' => array('subdir2')))
948 * @param array struc is array('dir' => array of files in dir,'dir/subdir' => array of
949 * files in dir/subdir,...)
950 * @param array array form of 'dir/subdir/subdir2' array('dir','subdir','subdir2')
951 * @return array same as struc but with array('dir' => array(file1,file2,'subdir' =>
952 * array(file1,...)))
953 */
954 function setup_dirs($struc,$dir,$contents)
955 {
956     if (!count($dir))
957     {
958         foreach($contents as $dir => $files)
959         {
960             if (is_string($dir))
961             {
962                 if (strpos($dir, '/'))
963                 {
964                     $test = true;
965                     $a = $contents[$dir];
966                     unset($contents[$dir]);
967                     $b = explode('/', $dir);
968
969

```

```
970     $c = array_shift($b);
971     if (isset($contents[$c]))
972     {
973         $contents[$c] = set_dir($contents[$c],setup_dirs(array(),$b,$a));
974     } else $contents[$c] = setup_dirs(array(),$b,$a);
975     }
976 }
977 return $contents;
978 }
979 $me = array_shift($dir);
980 if (!isset($struc[$me])) $struc[$me] = array();
981 $struc[$me] = setup_dirs($struc[$me],$dir,$contents);
982 return $struc;
983 }
984 }
985 if (!function_exists('get_include_path')) {
986 function get_include_path()
987 {
988     return ini_get('include_path');
989 }
990 }
991 }
992 ?>
```

# File Source for phpdoc.inc

Documentation for this file is available at [phpdoc.inc](#)

```
1  <?php
2  /**
3  * startup file
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2000-2007 Joshua Eichorn, Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @category ToolsAndUtilities
29 * @package phpdDocumentor
30 * @author Joshua Eichorn <jeichorn@phpdoc.org>
31 * @author Gregory Beaver <celflog@php.net>
32 * @copyright 2000-2007 Joshua Eichorn, Gregory Beaver
33 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34 * @version CVS: $Id: phpdoc.inc 243933 2007-10-10 01:18:25Z ashnazg $
35 * @link http://www.phpdoc.org
36 * @link http://pear.php.net/PhpDocumentor
37 * @since 0.1
38 * @filesource
39 * @todo CS cleanup - change package to PhpDocumentor
40 */
41
42
43 // set up include path so we can find all files, no matter what
44 $a = explode('/', str_replace('\\', '/', dirname(realpath(__FILE__))));
45 array_pop($a);
46 $GLOBALS['_phpDocumentor_install_dir'] = join('/', $a);
47 // add my directory to the include path, and make it first, should fix any errors
48 if (substr(PHP_OS, 0, 3) == 'WIN')
49 ini_set('include_path',
50         $GLOBALS['_phpDocumentor_install_dir'].';'.ini_get('include_path'));
51 else
52 ini_set('include_path',
53         $GLOBALS['_phpDocumentor_install_dir'].';'.ini_get('include_path'));
54
55 /**
56 * All command-line handling from previous version has moved to here
57 *
58 * Many settings also moved to phpDocumentor.ini
59 */
60 require_once "phpDocumentor/Setup.inc.php";
61
62 $phpdoc = new phpDocumentor_setup;
63 $phpdoc-> readCommandLineSettings();
64 $phpdoc-> setupConverters();
65 $phpdoc-> createDocs();
66 ?>
```

# File Source for DescHTML.inc

Documentation for this file is available at [DescHTML.inc](#)

```
1  <?php
2  /**
3  * All abstract representations of html tags in DocBlocks are handled by the
4  * classes in this file
5  *
6  * Before version 1.2, phpDocumentor simply passed html to converters, without
7  * much thought, except the {@link adv_htmlentities()} function was provided
8  * along with a list of allowed html. That list is no longer used, in favor
9  * of these classes.
10 *
11 * The PDF Converter output looked wretched in version 1.1.0 because line breaks
12 * in DocBlocks were honored. This meant that output often had just a few words
13 * on every other line! To fix this problem, DocBlock descriptions are now
14 * parsed using the {@link ParserDescParser}, and split into paragraphs. In
15 * addition, html in DocBlocks are parsed into these objects to allow for easy
16 * conversion in destination converters. This design also allows different
17 * conversion for different templates within a converter, which separates
18 * design from logic almost 100%
19 *
20 * phpDocumentor :: automatic documentation generator
21 *
22 * PHP versions 4 and 5
23 *
24 * Copyright (c) 2002-2007 Gregory Beaver
25 *
26 * LICENSE:
27 *
28 * This library is free software; you can redistribute it
29 * and/or modify it under the terms of the GNU Lesser General
30 * Public License as published by the Free Software Foundation;
31 * either version 2.1 of the License, or (at your option) any
32 * later version.
33 *
34 * This library is distributed in the hope that it will be useful,
35 * but WITHOUT ANY WARRANTY; without even the implied warranty of
36 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
37 * Lesser General Public License for more details.
38 *
39 * You should have received a copy of the GNU Lesser General Public
40 * License along with this library; if not, write to the Free Software
41 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
42 *
43 * @category ToolsAndUtilities
44 * @package phpDocumentor
45 * @subpackage DescHTML
46 * @author Greg Beaver <cellog@php.net>
47 * @copyright 2002-2007 Gregory Beaver
48 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
49 * @version CVS: $Id: DescHTML.inc 246329 2007-11-17 03:07:00Z ashnazg $
50 * @filesource
51 * @link http://www.phpdoc.org
52 * @link http://pear.php.net/PhpDocumentor
53 * @see parserDocBlock, parserInclude, parserPage, parserClass
54 * @see parserDefine, parserFunction, parserMethod, parserVar
55 * @since 1.2
56 * @todo CS cleanup - change package to PhpDocumentor
57 */
58 /**
59 * Used for <><> in a description
60 *
61 * @category ToolsAndUtilities
62 * @package phpDocumentor
63 * @subpackage DescHTML
64 * @author Greg Beaver <cellog@php.net>
65 * @copyright 2002-2007 Gregory Beaver
66 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
67 * @version Release: @VER@
```

```

68 * @link      http://www.phpdoc.org
69 * @link      http://pear.php.net/PhpDocumentor
70 * @since    1.2
71 * @todo     CS cleanup - change package to PhpDocumentor
72 * @todo     CS cleanup - rename class to ParserCode
73 */
74 class parserCode extends parserStringWithInlineTags
75 {
76     /**
77      * performs the conversion of code tags
78      *
79      * @param Converter &$c      the converter object
80      *
81      * @return string the converted code block
82      * @uses Converter::ProgramExample()
83      * @todo CS cleanup - rename method to convert()
84      */
85     function Convert(& $c)
86     {
87         if (!isset($this-> value[0])) {
88             return '';
89         }
90         if (is_string($this-> value[0]) && $this-> value[0] == "\n") {
91             $this-> value[0] = substr($this-> value[0], 1);
92         }
93         $linktags = array();
94         foreach ($this-> value as $val) {
95             if (phpDocumentor_get_class($val) == 'parserlinkinlinetag'
96                 || phpDocumentor_get_class($val) == 'parsertutorialinlinetag'
97             ) {
98                 $linktags[] = array(
99                     $c-> postProcess($val-> Convert($c, false, false)), $val);
100            }
101        }
102        $a = $c-> ProgramExample(rtrim(ltrim(parent::Convert($c,
103            false, false), "\n\r")));
104        foreach ($linktags as $tag) {
105            $a = str_replace($tag[0], $tag[1]-> Convert($c, false, false), $a);
106        }
107        return $a;
108    }
109 }
110 /**
111  * Used for <<pre>> in a description
112  *
113  * @category ToolsAndUtilities
114  * @package phpDocumentor
115  * @subpackage DescHTML
116  * @author Greg Beaver <cellog@php.net>
117  * @copyright 2002-2007 Gregory Beaver
118  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
119  * @version Release: @VER@
120  * @link http://www.phpdoc.org
121  * @link http://pear.php.net/PhpDocumentor
122  * @since 1.2
123  * @todo CS cleanup - change package to PhpDocumentor
124  * @todo CS cleanup - rename class to ParserPre
125  */
126 class parserPre extends parserStringWithInlineTags
127 {
128     /**
129      * performs the conversion of code tags
130      *
131      * @param Converter &$c      the converter object
132      *
133      * @return string the converted pre block
134      * @uses Converter::PreserveWhiteSpace()
135      * @todo CS cleanup - rename method to convert()
136      */
137     function Convert(& $c)
138     {
139         return $c-> PreserveWhiteSpace(rtrim(ltrim(parent::Convert($c,
140             false, false), "\n\r")));
141     }
142 }
143 /**
144  * Used for <<b>> in a description
145  *

```

```

148 * @category ToolsAndUtilities
149 * @package phpDocumentor
150 * @subpackage DescHTML
151 * @author Greg Beaver <cellog@php.net>
152 * @copyright 2002-2007 Gregory Beaver
153 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
154 * @version Release: @VER@
155 * @link http://www.phpdoc.org
156 * @link http://pear.php.net/PhpDocumentor
157 * @since 1.2
158 * @todo CS cleanup - change package to PhpDocumentor
159 * @todo CS cleanup - rename class to ParserB
160 */
161 class parserB extends parserStringWithInlineTags
162 {
163     /**
164      * performs the conversion of bold tags
165      *
166      * @param Converter &$c      the converter object
167      *
168      * @return string the converted pre block
169      * @uses Converter::Bolden()
170      * @todo CS cleanup - rename method to convert()
171      */
172     function Convert(& $c)
173     {
174         return $c-> Bolden(parent::Convert($c));
175     }
176 }
177 /**
178 * Used for <><i>> in a description
179 *
180 * @category ToolsAndUtilities
181 * @package phpDocumentor
182 * @subpackage DescHTML
183 * @author Greg Beaver <cellog@php.net>
184 * @copyright 2002-2007 Gregory Beaver
185 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
186 * @version Release: @VER@
187 * @link http://www.phpdoc.org
188 * @link http://pear.php.net/PhpDocumentor
189 * @since 1.2
190 * @todo CS cleanup - change package to PhpDocumentor
191 * @todo CS cleanup - rename class to ParserI
192 * @todo CS cleanup - rename class to ParserI
193 */
194 class parserI extends parserStringWithInlineTags
195 {
196     /**
197      * performs the conversion of italic tags
198      *
199      * @param Converter &$c      the converter object
200      *
201      * @return string the converted pre block
202      * @uses Converter::Italicize()
203      * @todo CS cleanup - rename method to convert()
204      */
205     function Convert(& $c)
206     {
207         return $c-> Italicize(parent::Convert($c));
208     }
209 }
210 /**
211 * Used for <><var>> in a description
212 *
213 * @category ToolsAndUtilities
214 * @package phpDocumentor
215 * @subpackage DescHTML
216 * @author Greg Beaver <cellog@php.net>
217 * @copyright 2002-2007 Gregory Beaver
218 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
219 * @version Release: @VER@
220 * @link http://www.phpdoc.org
221 * @link http://pear.php.net/PhpDocumentor
222 * @since 1.2
223 * @todo CS cleanup - change package to PhpDocumentor
224 * @todo CS cleanup - rename class to ParserDescVar
225 * /
226 class parserDescVar extends parserStringWithInlineTags

```

```

228 {
229     /**
230      * performs the conversion of variable tags
231      *
232      * @param Converter &$c      the converter object
233      *
234      * @return string the converted pre block
235      * @uses Converter::Varize()
236      * @todo CS cleanup - rename method to convert()
237      */
238     function Convert(&    $c)
239     {
240         return $c->  Varize(parent::Convert($c));
241     }
242 }
243
244 /**
245  * Used for <<samp>> in a description
246  *
247  * @category ToolsAndUtilities
248  * @package phpDocumentor
249  * @subpackage DescHTML
250  * @author Greg Beaver <cellog@php.net>
251  * @copyright 2002-2007 Gregory Beaver
252  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
253  * @version Release: @VER@
254  * @link http://www.phpdoc.org
255  * @link http://pear.php.net/PhpDocumentor
256  * @since 1.2
257  * @todo CS cleanup - change package to PhpDocumentor
258  * @todo CS cleanup - rename class to ParserSamp
259  */
260 class parserSamp extends parserStringWithInlineTags
261 {
262     /**
263      * performs the conversion of sample tags
264      *
265      * @param Converter &$c      the converter object
266      *
267      * @return string the converted pre block
268      * @uses Converter::Sampize()
269      * @todo CS cleanup - rename method to convert()
270      */
271     function Convert(&    $c)
272     {
273         return $c->  Sampize(parent::Convert($c));
274     }
275 }
276
277 /**
278  * Used for <<kbd>> in a description
279  *
280  * @category ToolsAndUtilities
281  * @package phpDocumentor
282  * @subpackage DescHTML
283  * @author Greg Beaver <cellog@php.net>
284  * @copyright 2002-2007 Gregory Beaver
285  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
286  * @version Release: @VER@
287  * @link http://www.phpdoc.org
288  * @link http://pear.php.net/PhpDocumentor
289  * @since 1.2
290  * @todo CS cleanup - change package to PhpDocumentor
291  * @todo CS cleanup - rename class to ParserKbd
292  */
293 class parserKbd extends parserStringWithInlineTags
294 {
295     /**
296      * performs the conversion of keyboard tags
297      *
298      * @param Converter &$c      the converter object
299      *
300      * @return string the converted pre block
301      * @uses Converter::Kbdize()
302      * @todo CS cleanup - rename method to convert()
303      */
304     function Convert(&    $c)
305     {
306         return $c->  Kbdize(parent::Convert($c));
307     }

```

```

308     }
309
310 /**
311 * Used for <>br></> in a description
312 *
313 * @category ToolsAndUtilities
314 * @package phpDocumentor
315 * @subpackage DescHTML
316 * @author Greg Beaver <cellog@php.net>
317 * @copyright 2002-2007 Gregory Beaver
318 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
319 * @version Release: @VER@
320 * @link http://www.phpdoc.org
321 * @link http://pear.php.net/PhpDocumentor
322 * @since 1.2
323 * @todo CS cleanup - change package to PhpDocumentor
324 * @todo CS cleanup - rename class to ParserBr
325 */
326 class parserBr extends parserStringWithInlineTags
327 {
328     /**
329      * performs the conversion of linebreak tags
330      *
331      * @param Converter &$c      the converter object
332      *
333      * @return string the converted pre block
334      * @uses Converter::Br()
335      * @todo CS cleanup - rename method to convert()
336      */
337     function Convert(& $c)
338     {
339         return $c-> Br($this-> getString());
340     }
341 }
342
343 /**
344 * Used for lists <>ol></> and <>ul></>
345 *
346 * @category ToolsAndUtilities
347 * @package phpDocumentor
348 * @subpackage DescHTML
349 * @author Greg Beaver <cellog@php.net>
350 * @copyright 2002-2007 Gregory Beaver
351 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
352 * @version Release: @VER@
353 * @link http://www.phpdoc.org
354 * @link http://pear.php.net/PhpDocumentor
355 * @since 1.2
356 * @todo CS cleanup - change package to PhpDocumentor
357 * @todo CS cleanup - rename class to ParserList
358 */
359 class parserList extends parserStringWithInlineTags
360 {
361     /**
362      * @var boolean
363      */
364     var $numbered;
365     /**
366      * @var integer
367      */
368     var $items = 0;
369     /**
370      * Constructor - create a new list
371      *
372      * @param integer $numbered a reference number for the new list
373      */
374     function parserList($numbered)
375     {
376         $this-> numbered = $numbered;
377     }
378
379     /**
380      * add an item to a list
381      *
382      * @param parserStringWithInlineTags $item the item to add
383      *
384      * @return void
385      */
386     function addItem($item)
387     {

```

```

388     $this-> value[$this-> items++] = $item;
389 }
390 /**
391 * add a list
392 *
393 * @param parserList $list the list to add
394 *
395 * @return void
396 */
397 function addList($list)
398 {
399     $this-> value[$this-> items++] = $list;
400 }
401
402 /**
403 * performs the conversion of list tags
404 *
405 * @param Converter &$c      the converter object
406 *
407 * @return string the converted pre block
408 * @uses Converter::ListItem() enclose each item of the list
409 * @uses Converter::EncloseList() enclose the list
410 * @todo CS cleanup - rename method to convert()
411 */
412 function Convert(& $c)
413 {
414     $list = '';
415     foreach ($this-> value as $item) {
416         $list .= $c-> ListItem(trim($item-> Convert($c)));
417     }
418     return $c-> EncloseList($list, $this-> numbered);
419 }
420 }
421 }
422 ?>

```

# File Source for DocBlockTags.inc

Documentation for this file is available at [DocBlockTags.inc](#)

```
1  <?php
2  /**
3  * All abstract representations of DocBlock tags are defined
4  * by the classes in this file
5  *
6  * phpDocumentor :: automatic documentation generator
7  *
8  * PHP versions 4 and 5
9  *
10 * Copyright (c) 2002-2008 Gregory Beaver
11 *
12 * LICENSE:
13 *
14 * This library is free software; you can redistribute it
15 * and/or modify it under the terms of the GNU Lesser General
16 * Public License as published by the Free Software Foundation;
17 * either version 2.1 of the License, or (at your option) any
18 * later version.
19 *
20 * This library is distributed in the hope that it will be useful,
21 * but WITHOUT ANY WARRANTY; without even the implied warranty of
22 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
23 * Lesser General Public License for more details.
24 *
25 * You should have received a copy of the GNU Lesser General Public
26 * License along with this library; if not, write to the Free Software
27 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
28 *
29 * @category ToolsAndUtilities
30 * @package phpDocumentor
31 * @subpackage DocBlockTags
32 * @author Greg Beaver <cellog@php.net>
33 * @copyright 2002-2008 Gregory Beaver
34 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
35 * @version CVS: $Id: DocBlockTags.inc 287889 2009-08-30 07:27:39Z ashnazg $
36 * @filesource
37 * @link http://www.phpdoc.org
38 * @link http://pear.php.net/PhpDocumentor
39 * @see parserDocBlock, parserInclude, parserPage, parserClass
40 * @see parserDefine, parserFunction, parserMethod, parserVar
41 * @since separate file since version 1.2
42 * @todo CS cleanup - change package to PhpDocumentor
43 */
44 /**
45 * used to represent standard tags like @access, etc.
46 * This class is aware of inline tags, and will automatically handle them
47 * using inherited functions
48 *
49 * @category ToolsAndUtilities
50 * @package phpDocumentor
51 * @subpackage DocBlockTags
52 * @author Greg Beaver <cellog@php.net>
53 * @copyright 2002-2008 Gregory Beaver
54 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
55 * @version Release: @VER@
56 * @link http://www.phpdoc.org
57 * @link http://pear.php.net/PhpDocumentor
58 * @since 1.0rc1
59 * @todo CS cleanup - change package to PhpDocumentor
60 * @todo CS cleanup - change classname to PhpDocumentor_*
61 */
62 class parserTag extends parserStringWithInlineTags
63 {
64     /**
65      * Type is used by many functions to skip the hassle of
66      * if phpDocumentor_get_class($blah) == 'parserBlah' always '_tag'
67      * @var string
```

```

68      */
69  var $type = '_tag';
70 /**
71 * tag name (see, access, etc.)
72 * @var string
73 */
74 var $keyword = '';
75
76 /**
77 * Set up the tag
78 *
79 * {@source}
80 *
81 * @param string $keyword tag name
82 * @param parserStringWithInlineTags $value tag value
83 * @param boolean $noparse whether to parse the $value
84 * for html tags
85 */
86 function parserTag($keyword, $value, $noparse = false)
87 {
88     $this-> keyword = $keyword;
89     if (!$noparse) {
90         $parser = new parserDescParser;
91         $parser-> subscribe('* ', $this);
92         $parser-> parse($value-> value, true, 'parserstringwithinlinetags');
93     } else {
94         $this-> value = $value;
95     }
96 }
97
98 /**
99 * Perform the output conversion on this {@link parserTag}
100 * using the {@link Converter output converter} that is passed in
101 *
102 * @param Converter &$converter the converter object
103 *
104 * @return string
105 * @see Converter
106 * @todo CS cleanup - rename to convert for camelCase rule
107 */
108 function Convert(& $converter)
109 {
110     if (is_array($this-> value)) {
111         if (count($this-> value) == 1) {
112             reset($this-> value);
113             list($val) = each($this-> value);
114             $a = $val-> Convert($converter);
115             return $a;
116         }
117         $result = '';
118         foreach ($this-> value as $val) {
119             // this is only true if we processed the description
120             // in the constructor
121             if (phpDocumentor_get_class($val)
122                 == 'parserstringwithinlinetags') {
123                 $result .= $converter->
124                     EncloseParagraph($val-> Convert($converter));
125             } else {
126                 $result .= $val-> Convert($converter);
127             }
128         }
129         return $result;
130     } else {
131         $a = $this-> value-> Convert($converter);
132         return $a;
133     }
134 }
135
136 /**
137 * Gets a count of the number of paragraphs in this
138 * tag's description.
139 *
140 * Useful in determining whether to enclose the
141 * tag in a paragraph or not.
142 *
143 * @return integer (actually, body is empty, so it doesn't return at all)
144 * @access private
145 * @todo does this need to be implemented? its body is empty
146 */
147 function _valueParagraphCount()

```

```

148 {
149 }
150
151 /**
152 * Called by the {@link parserDescParser} when processing a description.
153 *
154 * @param integer $a    not used
155 * @param array   $desc array of {@link parserStringWithInlineTags}
156 *                      representing paragraphs in the tag description
157 *
158 * @return void
159 * @see parserTag::parserTag()
160 * @todo CS cleanup - rename to handleEvent for camelCase rule
161 */
162 function HandleEvent($a,$desc)
163 {
164     $this-> value = $desc;
165 }
166
167 /**
168 * Returns the text minus any inline tags
169 *
170 * @return string the text minus any inline tags
171 * @see parserStringWithInlineTags::getString()
172 */
173 function getString()
174 {
175     if (is_array($this-> value)) {
176         $result = '';
177         foreach ($this-> value as $val) {
178             $result .= $val-> getString();
179         }
180         return $result;
181     } else {
182         return $this-> value-> getString();
183     }
184 }
185
186 /**
187 * This class represents the @name tag
188 *
189 * @category ToolsAndUtilities
190 * @package phpDocumentor
191 * @subpackage DocBlockTags
192 * @author Greg Beaver <cellog@php.net>
193 * @copyright 2002-2008 Gregory Beaver
194 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
195 * @version Release: @VER@
196 * @link http://www.phpdoc.org
197 * @link http://pear.php.net/PhpDocumentor
198 * @tutorial tags.name.pkg
199 * @todo CS cleanup - change package to PhpDocumentor
200 * @todo CS cleanup - change classname to PhpDocumentor_*
201 */
202
203 class parserNameTag extends parserTag
204 {
205     /**
206      * tag name
207      * @var string
208      */
209     var $keyword = 'name';
210
211     /**
212      * set up the name tag
213      *
214      * @param string $name tag name (not used)
215      * @param string $value tag value
216      */
217     function parserNameTag($name, $value)
218     {
219         $this-> value = $value;
220     }
221
222     /**
223      * process this tag through the given output converter
224      *
225      * @param Converter &$c      output converter
226      *
227      * @return string converted value of the tag

```

```

228     * @see parserStringWithInlineTags::Convert()
229     * @todo CS cleanup - rename to convert for camelCase rule
230     */
231     function Convert(& $c)
232     {
233         return $this-> value;
234     }
235 }
236
237 /**
238 * This class represents the @access tag
239 *
240 * @category ToolsAndUtilities
241 * @package phpDocumentor
242 * @subpackage DocBlockTags
243 * @author Greg Beaver <cellog@php.net>
244 * @copyright 2002-2008 Gregory Beaver
245 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
246 * @version Release: @VER@
247 * @link http://www.phpdoc.org
248 * @link http://pear.php.net/PhpDocumentor
249 * @tutorial tags.access.pkg
250 * @todo CS cleanup - change package to PhpDocumentor
251 * @todo CS cleanup - change classname to PhpDocumentor_*
252 */
253 class parserAccessTag extends parserTag
254 {
255     /**
256      * tag name
257      * @var string
258     */
259     var $keyword = 'access';
260
261     /**
262      * set to true if the returned tag has a value type of private, protected
263      * or public, false otherwise
264      * @var boolean
265     */
266     var $isValid = false;
267
268     /**
269      * checks $value to make sure it is private, protected or public, otherwise
270      * it's not a valid @access tag
271      *
272      * @param parserStringWithInlineTags $value the tag value
273      * @see $isValid
274     */
275     function parserAccessTag($value)
276     {
277         if (!is_string($value)) {
278             if (is_object($value)) {
279                 if (method_exists($value, 'getstring')) {
280                     $value = $value-> getString();
281                 }
282             }
283         }
284         switch(trim($value)) {
285             case 'private' :
286             case 'public' :
287             case 'protected' :
288                 $this-> value = $value;
289                 $this-> isValid = true;
290                 break;
291             default :
292                 addError(PDERROR_ACCESS_WRONG_PARAM, $value);
293                 $this-> value = 'public';
294                 break;
295         }
296     }
297 }
298
299 /**
300  * process this tag through the given output converter
301  *
302  * @param Converter &$converter the output converter
303  *
304  * @return string converted value of the tag
305  * @see parserStringWithInlineTags::Convert()
306  * @todo CS cleanup - rename to convert for camelCase rule
307 */

```

```

308     function Convert(& $converter)
309     {
310         return $this-> value;
311     }
312
313     /**
314      * No inline tags are possible, returns 'public', 'protected' or 'private'
315      *
316      * @return string returns the text minus any inline tags
317      */
318     function getString()
319     {
320         return $this-> value;
321     }
322 }
323
324 /**
325  * represents the "@return" tag
326  *
327  * @category ToolsAndUtilities
328  * @package phpDocumentor
329  * @subpackage DocBlockTags
330  * @author Greg Beaver <cellog@php.net>
331  * @copyright 2002-2008 Gregory Beaver
332  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
333  * @version Release: @VER@
334  * @link http://www.phpdoc.org
335  * @link http://pear.php.net/PhpDocumentor
336  * @tutorial tags.return.pkg
337  * @since 1.0rc1
338  * @todo CS cleanup - change package to PhpDocumentor
339  * @todo CS cleanup - change classname to PhpDocumentor_*
340 */
341 class parserReturnTag extends parserTag
342 {
343     /**
344      * always 'return'
345      * @var string
346      */
347     var $keyword = 'return';
348     /**
349      * the type a function returns
350      */
351     var $returnType = 'void';
352
353     /**
354      * contains a link to the documentation for a class
355      * passed as a type in @return, @var or @param
356      *
357      * Example:
358      *
359      * <code>
360      * class myclass
361      * {
362      * ...
363      * }
364      * /** @return myclass blahblahblah
365      * ...
366      * </code>
367      *
368      * In this case, $converted_returnType will contain a link to myclass
369      * instead of the string 'myclass'
370      *
371      * @var mixed either the same as $returnType or a link to the docs for a class
372      * @see $returnType
373      */
374     var $convertedReturnType = false;
375
376     /**
377      * set up the tag
378      *
379      * @param string $returnType returned datatype
380      * @param parserStringWithInlineTags $value tag value
381      */
382     function parserReturnTag($returnType, $value)
383     {
384         $this-> returnType = $returnType;
385         parent::parserTag('return', $value);
386     }
387 }
```

```

388 /**
389  * process this tag through the given output converter
390  * (sets up the $converted_returnType)
391  *
392  * @param Converter &$converter      the output converter
393  *
394  * @return string converted value of the tag
395  * @see parserStringWithInlineTags::Convert(), $converted_returnType
396  * @todo CS cleanup - rename to convert for camelCase rule
397  */
398 function Convert(&    $converter)
399 {
400     $my_types = '';
401     if (strpos($this-> returnType, '|')) {
402         $types = explode('|', $this-> returnType);
403         foreach ($types as $returntype) {
404             $a = $converter-> getLink($returntype);
405             if (is_object($a) && phpDocumentor_get_class($a) == 'classlink') {
406                 if (!empty($my_types)) {
407                     $my_types .= '|';
408                 }
409                 $my_types .= $converter->
410                             returnSee($a, $converter-> type_adjust($returntype));
411             } else {
412                 if (!empty($my_types)) {
413                     $my_types .= '|';
414                 }
415                 $my_types .= $converter-> type_adjust($returntype);
416             }
417         }
418         $this-> converted_returnType = $my_types;
419     } else {
420         $a = $converter-> getLink($this-> returnType);
421         if (is_object($a) && phpDocumentor_get_class($a) == 'classlink') {
422             $this-> converted_returnType = $converter->
423                             returnSee($a, $converter-> type_adjust($this-> returnType));
424         } else {
425             $this-> converted_returnType = $converter->
426                             type_adjust($this-> returnType);
427         }
428     }
429     return parserTag::Convert($converter);
430 }
431 }
432 /**
433  * represents the "@property" tag
434  *
435  * @category ToolsAndUtilities
436  * @package phpDocumentor
437  * @subpackage DocBlockTags
438  * @author Greg Beaver <cellog@php.net>
439  * @copyright 2002-2008 Gregory Beaver
440  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
441  * @version Release: @VER@
442  * @link http://www.phpdoc.org
443  * @link http://pear.php.net/PhpDocumentor
444  * @tutorial tags.property.pkg
445  * @since 1.4.0a1
446  * @todo CS cleanup - change package to PhpDocumentor
447  * @todo CS cleanup - change classname to PhpDocumentor_*
448  */
449 class parserPropertyTag extends parserReturnTag
450 {
451     /**
452      * always 'property'
453      * @var string
454      */
455     var $keyword = 'property';
456     /**
457      * the type a property has
458      * @var string
459      */
460     var $returnType = 'mixed';
461     /**
462      * set up the property tag
463      *
464      * @param string          $returnType the tag value's datatype
465      * @param parserStringWithInlineTags $value   the tag value
466

```

```

468      */
469      function parserPropertyTag($returnType, $value)
470      {
471          $this-> returnType = $returnType;
472          parent::parserTag($this-> keyword, $value);
473      }
474
475
476 /**
477 * represents the "@property-read" tag
478 *
479 * @category ToolsAndUtilities
480 * @package phpDocumentor
481 * @subpackage DocBlockTags
482 * @author Greg Beaver <cellog@php.net>
483 * @copyright 2002-2008 Gregory Beaver
484 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
485 * @version Release: @VER@
486 * @link http://www.phpdoc.org
487 * @link http://pear.php.net/PhpDocumentor
488 * @tutorial tags.property.pkg
489 * @since 1.4.0a1
490 * @todo CS cleanup - change package to PhpDocumentor
491 * @todo CS cleanup - change classname to PhpDocumentor_*
492 */
493 class parserPropertyReadTag extends parserPropertyTag
494 {
495     /**
496     * always 'property-read'
497     * @var string
498     */
499     var $keyword = 'property-read';
500
501
502 /**
503 * represents the "@property-write" tag
504 *
505 * @category ToolsAndUtilities
506 * @package phpDocumentor
507 * @subpackage DocBlockTags
508 * @author Greg Beaver <cellog@php.net>
509 * @copyright 2002-2008 Gregory Beaver
510 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
511 * @version Release: @VER@
512 * @link http://www.phpdoc.org
513 * @link http://pear.php.net/PhpDocumentor
514 * @tutorial tags.property.pkg
515 * @since 1.4.0a1
516 * @todo CS cleanup - change package to PhpDocumentor
517 * @todo CS cleanup - change classname to PhpDocumentor_*
518 */
519 class parserPropertyWriteTag extends parserPropertyTag
520 {
521     /**
522     * always 'property-write'
523     * @var string
524     */
525     var $keyword = 'property-write';
526
527
528 /**
529 * represents the "@method" tag
530 *
531 * @category ToolsAndUtilities
532 * @package phpDocumentor
533 * @subpackage DocBlockTags
534 * @author Greg Beaver <cellog@php.net>
535 * @copyright 2002-2008 Gregory Beaver
536 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
537 * @version Release: @VER@
538 * @link http://www.phpdoc.org
539 * @link http://pear.php.net/PhpDocumentor
540 * @tutorial tags.method.pkg
541 * @since 1.4.0a1
542 * @todo CS cleanup - change package to PhpDocumentor
543 * @todo CS cleanup - change classname to PhpDocumentor_*
544 */
545 class parserMethodTag extends parserPropertyTag
546 {
547     /**

```

```

548     * always 'method'
549     * @var string
550     */
551     var $keyword = 'method';
552     /**
553     * the return type a method has
554     * @var string
555     */
556     var $returnType = 'void';
557 }
558 /**
559 * represents the "@var" tag
560 *
561 *
562 * @category ToolsAndUtilities
563 * @package phpDocumentor
564 * @subpackage DocBlockTags
565 * @author Greg Beaver <cellog@php.net>
566 * @copyright 2002-2008 Gregory Beaver
567 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
568 * @version Release: @VER@
569 * @link http://www.phpdoc.org
570 * @link http://pear.php.net/PhpDocumentor
571 * @tutorial tags.var.pkg
572 * @since 1.0rc1
573 * @todo CS cleanup - change package to PhpDocumentor
574 * @todo CS cleanup - change classname to PhpDocumentor_*
575 */
576 class parserVarTag extends parserReturnTag
577 {
578     /**
579     * always 'var'
580     * @var string
581     */
582     var $keyword = 'var';
583     /**
584     * the type a var has
585     * @var string
586     */
587     var $returnType = 'mixed';
588 }
589 /**
590 * represents the "@param" tag
591 *
592 *
593 * @category ToolsAndUtilities
594 * @package phpDocumentor
595 * @subpackage DocBlockTags
596 * @author Greg Beaver <cellog@php.net>
597 * @copyright 2002-2008 Gregory Beaver
598 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
599 * @version Release: @VER@
600 * @link http://www.phpdoc.org
601 * @link http://pear.php.net/PhpDocumentor
602 * @tutorial tags.param.pkg
603 * @todo CS cleanup - change package to PhpDocumentor
604 * @todo CS cleanup - change classname to PhpDocumentor_*
605 */
606 class parserParamTag extends parserVarTag
607 {
608     /**
609     * always 'param'
610     * @var string
611     */
612     var $keyword = 'param';
613 }
614 /**
615 * represents the "@staticvar" tag
616 *
617 *
618 * @category ToolsAndUtilities
619 * @package phpDocumentor
620 * @subpackage DocBlockTags
621 * @author Greg Beaver <cellog@php.net>
622 * @copyright 2002-2008 Gregory Beaver
623 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
624 * @version Release: @VER@
625 * @link http://www.phpdoc.org
626 * @link http://pear.php.net/PhpDocumentor
627 * @tutorial tags.staticvar.pkg

```

```

628 * @todo      CS cleanup - change package to PhpDocumentor
629 * @todo      CS cleanup - change classname to PhpDocumentor_*
630 */
631 class parserStaticvarTag extends parserParamTag
632 {
633     /**
634      * always 'staticvar'
635      * @var string
636      */
637     var $keyword = 'staticvar';
638 }
639
640 /**
641  * represents the "@link" tag
642  *
643  * @category ToolsAndUtilities
644  * @package phpDocumentor
645  * @subpackage DocBlockTags
646  * @author Greg Beaver <cellog@php.net>
647  * @copyright 2002-2008 Gregory Beaver
648  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
649  * @version Release: @VER@
650  * @link http://www.phpdoc.org
651  * @link http://pear.php.net/PhpDocumentor
652  * @since 1.0rc1
653  * @tutorial tags.link.pkg
654  * @todo CS cleanup - change package to PhpDocumentor
655  * @todo CS cleanup - change classname to PhpDocumentor_*
656 */
657 class parserLinkTag extends parserTag
658 {
659     /**
660      * always 'link'
661      * @var string
662      */
663     var $keyword = 'link';
664
665     /**
666      * sets up the link tag
667      *
668      * @param string $link URL to link to
669      *                      (might also contain the URL's
670      *                      description text)
671      */
672     function parserLinkTag($link)
673     {
674         $start = $val = $link-> getString();
675         if (strpos($val, ' ')) {
676             $val = explode(' ', $val);
677             $start = array_shift($val);
678             $val = join($val, ' ');
679         }
680         $a = new parserLinkInlineTag($start, $val);
681         $b = new parserStringWithInlineTags;
682         $b-> add($a);
683         $this-> value = $b;
684     }
685 }
686
687 /**
688  * represents the "@see" tag
689  *
690  * @category ToolsAndUtilities
691  * @package phpDocumentor
692  * @subpackage DocBlockTags
693  * @author Greg Beaver <cellog@php.net>
694  * @copyright 2002-2008 Gregory Beaver
695  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
696  * @version Release: @VER@
697  * @link http://www.phpdoc.org
698  * @link http://pear.php.net/PhpDocumentor
699  * @since 1.0rc1
700  * @tutorial tags.see.pkg
701  * @todo CS cleanup - change package to PhpDocumentor
702  * @todo CS cleanup - change classname to PhpDocumentor_*
703  */
704 class parserSeeTag extends parserLinkTag
705 {
706     /**
707      * always 'see'

```

```

708     * @var string
709     */
710     var $keyword = 'see';
711
712     /**
713      * sets up the see tag
714      *
715      * @param string $name element to link to
716      */
717     function parserSeeTag($name)
718     {
719         parserTag::parserTag($this-> keyword, $name, true);
720     }
721
722     /**
723      * process this tag through the given output converter
724      *
725      * @param Converter &$converter      the output converter
726      *
727      * @return string converted value of the tag
728      * @see parserStringWithInlineTags::Convert()
729      * @todo CS cleanup - rename to convert for camelCase rule
730      */
731     function Convert(& $converter)
732     {
733         if ($this-> value-> hasInlineTag()) {
734             addErrorDie(PDERROR_INLINETAG_IN_SEE);
735         }
736         $a = $converter-> getLink(trim($this-> value-> Convert($converter)));
737         if (is_string($a)) {
738             // feature 564991
739             if (strpos($a, '::://')) {
740                 // php function
741                 return $converter-> returnLink($a, str_replace('PHP MANUAL#', '', $this-> value-> Convert($converter)));
742             }
743             return $a;
744         }
745         if (is_object($a)) {
746             return $converter-> returnSee($a);
747         }
748         // getLink parsed a comma-delimited list of linked things,
749         // add the commas back in
750         if (is_array($a)) {
751             $b = '';
752             foreach ($a as $i => $bub) {
753                 if (!empty($b)) {
754                     $b .= ', ';
755                 }
756                 if (is_string($a[$i])) {
757                     $b .= $a[$i];
758                 }
759                 if (is_object($a[$i])) {
760                     $b .= $converter-> returnSee($a[$i]);
761                 }
762             }
763             return $b;
764         }
765     }
766     return false;
767 }
768 }
769
770 /**
771  * represents the "@see" tag
772  *
773  * Link to a license, instead of including lines and lines of license information
774  * in every file
775  *
776  * @category ToolsAndUtilities
777  * @package phpDocumentor
778  * @subpackage DocBlockTags
779  * @author Greg Beaver <cellog@php.net>
780  * @copyright 2002-2008 Gregory Beaver
781  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
782  * @version Release: @VER@
783  * @link http://www.phpdoc.org
784  * @link http://pear.php.net/PhpDocumentor
785  * @tutorial tags.license.pkg
786  * @todo CS cleanup - change package to PhpDocumentor
787  * @todo CS cleanup - change classname to PhpDocumentor_*

```

```

788 */
789 class parserLicenseTag extends parserLinkTag
790 {
791     /**
792      * always 'license'
793      * @var string
794      */
795     var $keyword = 'license';
796
797     /**
798      * set up the license tag
799      *
800      * @param string $name unused?
801      * @param string $link URL to link to
802      */
803     function parserLicenseTag($name, $link)
804     {
805         $a      = explode(' ', $link-> getString());
806         $url   = array_shift($a);
807         $license = join($a, ' ');
808         if (empty($license)) {
809             $license = $url;
810         }
811         $a = new parserLinkInlineTag($url, $license);
812         $b = new parserStringWithInlineTags;
813         $b-> add($a);
814         $this-> value = $b;
815     }
816
817 }
818 /**
819  * represents the "@uses" tag
820  *
821  * This is exactly like @see except that the element used
822  * has a @useby link to this element added to its docblock
823  *
824  * @category ToolsAndUtilities
825  * @package phpDocumentor
826  * @subpackage DocBlockTags
827  * @author Greg Beaver <ccellog@php.net>
828  * @copyright 2002-2008 Gregory Beaver
829  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
830  * @version Release: @VER@
831  * @link http://www.phpdoc.org
832  * @link http://pear.php.net/PhpDocumentor
833  * @since 1.2
834  * @tutorial tags.uses.pkg
835  * @todo CS cleanup - change package to PhpDocumentor
836  * @todo CS cleanup - change classname to PhpDocumentor_*
837  */
838 class parserUsesTag extends parserSeeTag
839 {
840     /**
841      * Always "uses"
842      * @var string
843      */
844     var $keyword = 'uses';
845     /**
846      * @access private
847      */
848     var $_description;
849
850     /**
851      * set up the uses tag
852      *
853      * @param string $seeeel element to link to
854      * @param parserStringWithInlineTags $description description of how
855      *          the element is used
856      */
857     function parserUsesTag($seeeel, $description)
858     {
859         if ($seeeel-> hasInlineTag()) {
860             addErrorDie(PDERROR_DUMBUSES);
861         }
862         parent::parserSeeTag($seeeel);
863         $this-> _description = $description;
864     }
865
866     /**
867      * Return a link to documentation for other element,

```

```

868 * and description of how it is used
869 *
870 * Works exactly like {@link parent::Convert()}
871 * except that it also includes a description of
872 * how the element used is used.
873 *
874 * @param Converter &$c      the output converter
875 *
876 * @return string link to the uses target
877 * @todo CS cleanup - rename to convert for camelCase rule
878 */
879 function Convert(&    $c)
880 {
881     $val      = $this->  value;
882     $see      = parent::Convert($c);
883     $this->  value = $this->  _description;
884     $desc_val = parserTag::Convert($c);
885     if (!empty($desc_val)) {
886         $see .= ' - ' . $desc_val;
887     }
888     $this->  value = $val;
889     return $see;
890 }
891 /**
892 * Get the text of the link to the element that is being used
893 *
894 * @return string
895 * @access private
896 */
897 function getSeeElement()
898 {
899     return $this->  value->  getString();
900 }
901 /**
902 * Get the description of how the element used is being used.
903 *
904 * @return parserStringWithInlineTags
905 */
906 function getDescription()
907 {
908     return $this->  _description;
909 }
910 /**
911 * This is a virtual tag, it is created by @uses to cross-reference the used element
912 *
913 * This is exactly like @uses.
914 *
915 * @category ToolsAndUtilities
916 * @package phpDocumentor
917 * @subpackage DocBlockTags
918 * @author Greg Beaver <celflog@php.net>
919 * @copyright 2002-2008 Gregory Beaver
920 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
921 * @version Release: @VER@
922 * @link http://www.phpdoc.org
923 * @link http://pear.php.net/PhpDocumentor
924 * @since 1.2
925 * @todo CS cleanup - change package to PhpDocumentor
926 * @todo CS cleanup - change classname to PhpDocumentor_*
927 */
928 class parserUsedByTag extends parserUsesTag
929 {
930     /**
931      * Always "usedby"
932      * @var string
933      */
934     var $keyword = 'usedby';
935     /**
936      * @access private
937      */
938     var $_link;
939     /**
940      * set up the usedby tag
941      */
942     /**
943      * @param abstractLink $link      link of element that uses this element
944      */
945 
```

```

948     * @param string      $description description of how the element is used
949     */
950     function parserUsedByTag($link, $description)
951     {
952         $this-> _value = $description;
953         $this-> _link = $link;
954     }
955
956 /**
957 * process this tag through the given output converter
958 *
959 * @param Converter &$c      the output converter
960 *
961 * @return string
962 * @todo CS cleanup - rename to convert for camelCase rule
963 */
964     function Convert(&    $c)
965     {
966         $see      = $c-> returnSee($this-> _link);
967         $desc_val = parserTag::Convert($c);
968         if (!empty($desc_val)) {
969             $see .= ' - ' . $desc_val;
970         }
971         return $see;
972     }
973 }
974
975 /**
976 * represents "@tutorial"
977 *
978 * This is exactly like @see except that it only links to tutorials
979 *
980 * @category ToolsAndUtilities
981 * @package phpDocumentor
982 * @subpackage DocBlockTags
983 * @author Greg Beaver <cellog@php.net>
984 * @copyright 2002-2008 Gregory Beaver
985 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
986 * @version Release: @VER@
987 * @link http://www.phpdoc.org
988 * @link http://pear.php.net/PhpDocumentor
989 * @since 1.2
990 * @tutorial phpDocumentor/tutorials.pkg
991 * @tutorial tags tutorial.pkg
992 * @todo CS cleanup - change package to PhpDocumentor
993 * @todo CS cleanup - change classname to PhpDocumentor_*
994 */
995 class parserTutorialTag extends parserSeeTag
996 {
997     /**
998     * Always "tutorial"
999     * @var string
1000    */
1001    var $keyword = 'tutorial';

1003 /**
1004 * process this tag through the given output converter
1005 *
1006 * @param Converter &$converter      the output converter
1007 *
1008 * @return string/bool
1009 * @see parserStringWithInlineTags::Convert()
1010 * @todo CS cleanup - rename to convert for camelCase rule
1011 */
1012     function Convert(&    $converter)
1013     {
1014         $a = $converter-> getTutorialLink(trim($this-> _value-> Convert($converter)));
1015         if (is_string($a)) {
1016             return $a;
1017         }
1018         if (is_object($a)) {
1019             return $converter-> returnSee($a);
1020         }
1021         // getLink parsed a comma-delimited list of linked thingsie,
1022         // add the commas back in
1023         if (is_array($a)) {
1024             $b = '';
1025             foreach ($a as $i =>    $b) {
1026                 if (!empty($b)) {
1027                     $b .= ', ';
1028                 }
1029             }
1030             return $b;
1031         }
1032     }

```

```

1028         }
1029         if (is_string($a[$i])) {
1030             $b .= $a[$i];
1031         }
1032         if (is_object($a[$i])) {
1033             $b .= $converter-> returnSee($a[$i]);
1034         }
1035     }
1036     return $b;
1037 }
1038 }
1039 }
1040 }
1041 /**
1042 * represents "@filesource"
1043 *
1044 * Use this to create a link to a highlighted phpxref-style source file listing
1045 *
1046 * @category ToolsAndUtilities
1047 * @package phpDocumentor
1048 * @subpackage DocBlockTags
1049 * @author Greg Beaver <cellog@php.net>
1050 * @copyright 2002-2008 Gregory Beaver
1051 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
1052 * @version Release: @VER@
1053 * @link http://www.phpdoc.org
1054 * @link http://pear.php.net/PhpDocumentor
1055 * @since 1.2
1056 * @tutorial tags.filesource.pkg
1057 * @todo CS cleanup - change package to PhpDocumentor
1058 * @todo CS cleanup - change classname to PhpDocumentor_*
1059 */
1060
1061 class parserFileSourceTag extends parserTag
1062 {
1063     /**
1064      * Always "filesource"
1065      * @var string
1066      */
1067     var $keyword = 'filesource';
1068     /**
1069      * @var array
1070      */
1071     var $source;
1072     /**
1073      * @var string
1074      */
1075     var $path;
1076     /**
1077      * Flag variable, controls double writes of file for each converter
1078      * @var array
1079      * @access private
1080      */
1081     var $_converted = array();
1082     /**
1083      * Set {@link $source} to $value, and set up path
1084      *
1085      * @param string $filepath the file's path
1086      * @param array $value output from
1087      *          {@link phpDocumentorTWordParser::getFileSize()}
1088      */
1089     function parserFileSourceTag($filepath, $value)
1090     {
1091         parent::parserTag($this-> keyword, '');
1092         $this-> path = $filepath;
1093         $this-> source = $value;
1094     }
1095
1096     /**
1097      * Return a link to the highlighted source and generate the source
1098      *
1099      * @param Converter &$c the output converter
1100      *
1101      * @return string output from {@link getSourceLink()}
1102      * @uses ConvertSource() generate source code and write it out
1103      * @todo CS cleanup - rename to convert for camelCase rule
1104      */
1105     function Convert(& $c)
1106     {

```

```

1108     $this-> ConvertSource($c);
1109     return $this-> getSourceLink($c);
1110 }
1111 /**
1112 * convert the source code
1113 *
1114 * @param Converter &$c      the output converter
1115 *
1116 * @return void
1117 * @uses phpDocumentor_HighlightParser highlights source code
1118 * @uses writeSource()
1119 * @todo CS cleanup - rename to convertSource for camelCase rule
1120 * @todo what's up with all the "return" statements?
1121 *       can they all be removed?
1122 */
1123 function ConvertSource(&    $c)
1124 {
1125     $this-> writeSource($c, $c->
1126         ProgramExample($this-> source, true, false, false, false, $this-> path));
1127     return;
1128     $parser = new phpDocumentor_HighlightParser;
1129     $return = '';
1130     $return = $parser->
1131         parse($this-> source, $c, false, false, false, $this-> path);
1132     $this-> writeSource($c, $return);
1133 }
1134
1135 /**
1136 * have the output converter write the source code
1137 *
1138 * @param Converter &$c      the output converter
1139 * @param string    $source highlighted source code
1140 *
1141 * @return void
1142 * @uses Converter::writeSource() export highlighted file source
1143 */
1144 function writeSource(&    $c, $source)
1145 {
1146     $c-> writeSource($this-> path, $source);
1147 }
1148
1149 /**
1150 * gets path to the sourcecode file
1151 *
1152 * @param Converter &$c      the output converter
1153 *
1154 * @return output from getSourceLink()
1155 * @uses Converter::getSourceLink()
1156 */
1157 function getSourceLink(&    $c)
1158 {
1159     return $c-> getSourceLink($this-> path);
1160 }
1161 }
1162
1163 /**
1164 * represents "@example"
1165 *
1166 * @category ToolsAndUtilities
1167 * @package phpDocumentor
1168 * @subpackage DocBlockTags
1169 * @author Greg Beaver <cellog@php.net>
1170 * @copyright 2002-2008 Gregory Beaver
1171 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
1172 * @version Release: @VER@
1173 * @link http://www.phpdoc.org
1174 * @link http://pear.php.net/PhpDocumentor
1175 * @tutorial tags.example.pkg
1176 * @todo CS cleanup - change package to PhpDocumentor
1177 * @todo CS cleanup - change classname to PhpDocumentor_*
1178 */
1179
1180 class parserExampleTag extends parserFileSourceTag
1181 {
1182     /**
1183     * always "example"
1184     * @var string
1185     */
1186     var $keyword = 'example';
1187

```

```

1188 /**
1189 * Reads and parses the example file indicated
1190 *
1191 * The example tag takes one parameter: the full path to a php file that
1192 * should be parsed and included as an example.
1193 *
1194 * @param parserStringWithInlineTags $value      tag value
1195 * @param string                   $current_path path of file containing
1196 *                                     this @example tag
1197 *
1198 * @uses phpDocumentorTWordParser::getFileSource() uses to parse an example
1199 *       and retrieve all tokens by line number
1200 * @todo does this "x = y = z = false" still work as expected in PHP5?
1201 * @todo CS cleanup - rename constant to TOKENIZER_EXT
1202 */
1203 function parserExampleTag($value, $current_path)
1204 {
1205     global $phpDocumentor_setting;
1206     parent::parserTag('example', $value);
1207     $path = false;
1208     // code thanks to Sam Blum, modified by Greg Beaver
1209     $tagValue = $value->getString();
1210
1211     $path = $isAbsPath
1212         = $pathOnly
1213         = $fileName
1214         = $fileExt
1215         = $original_path
1216         = $title
1217         = false;
1218     do {
1219         // make sure the format is stuff.ext description
1220         if (!preg_match('`(.*)\.(\\w*)\\s(.*)`', $tagValue, $match)) {
1221             // or format is stuff.ext
1222             if (!preg_match('`(.*)\\.(\\w*)\\s*$`', $tagValue, $match)) {
1223                 // Murphy: Some funny path was given
1224                 $original_path = $tagValue; // used for error output
1225                 break; // try-block
1226             }
1227         }
1228         if (strlen($match[1]) === 0) {
1229             // Murphy: Some funny path was given
1230             $original_path = $tagValue; // used for error output
1231             break; // try-block
1232         }
1233         $fileExt = $match[2];
1234         $title = 'example';
1235         if (isset($match[3])) {
1236             $title = trim($match[3]);
1237         }
1238         // Replace windows '\' the path.
1239         $pathTmp = str_replace('\\', '/', $match[1]);
1240
1241         // Is there a path and a file or is it just a file?
1242         if (strpos($pathTmp, '/') === false) {
1243             // No path part
1244             $pathOnly = '';
1245             $fileName = $pathTmp . '.' . $fileExt;
1246         } else {
1247             // split the path on the last directory, find the filename
1248             $splitPos = strrpos($pathTmp, '/');
1249             $pathOnly = substr($match[1], 0, $splitPos+1);
1250             $fileName = substr($match[1], $splitPos+1) . '.' . $fileExt;
1251             // Is the path absolute? (i.e. does it start like an absolute path?)
1252             if ('/' === $pathTmp[0]) || preg_match('`^\\w*:`i', $pathTmp)) {
1253                 // works for both windows 'C:' and URLs like 'http://'
1254                 $isAbsPath = true; // Yes
1255             }
1256         }
1257
1258         $original_path = $pathOnly . $fileName;
1259
1260         // Now look for the file starting with abs. path.
1261         if ($isAbsPath) {
1262             // remove any weirdities like ../../file.ext
1263             $tmp = realpath($original_path);
1264             if ($tmp && is_file($tmp)) {
1265                 $path = $tmp;
1266             }
1267             // Always break if abs. path was detected,

```

```

1268 // even if file was not found.
1269 break; // try-block
1270 }
1271
1272 // Search for the example file some standard places
1273 // 1) Look if the ini-var examplesdir is set and look there ...
1274 if (isset($_phpDocumentor_setting['examplesdir'])) {
1275     $tmp = realpath($_phpDocumentor_setting['examplesdir'])
1276         . PATH_DELIMITER . $original_path);
1277     if ($tmp && is_file($tmp)) {
1278         $path = $tmp; // Yo! found it :)
1279         break; // try-block
1280     }
1281 }
1282
1283 // 2) Then try to look for an 'example/-dir
1284 // below the *currently* parsed file ...
1285 if (!empty($current_path)) {
1286     $tmp = realpath(dirname($current_path) . PATH_DELIMITER
1287         . 'examples' . PATH_DELIMITER . $fileName);
1288     if ($tmp && is_file($tmp)) {
1289         $path = $tmp; // Yo! found it :)
1290         break; // try-block
1291     }
1292 }
1293
1294 // 3) Then try to look for the example file
1295 // below the subdir PHPDOCUMENTOR_BASE/examples/ ...
1296 if (is_dir(PHPDOCUMENTOR_BASE . PATH_DELIMITER . 'examples')) {
1297     $tmp = realpath(PHPDOCUMENTOR_BASE . PATH_DELIMITER
1298         . 'examples' . PATH_DELIMITER . $original_path);
1299     if ($tmp && is_file($tmp)) {
1300         $path = $tmp; // Yo! found it :)
1301         break; // try-block
1302     }
1303 }
1304
1305 $tmp = realpath(PHPDOCUMENTOR_BASE . PATH_DELIMITER . $original_path);
1306 if ($tmp && is_file($tmp)) {
1307     $path = $tmp; // Yo! found it :)
1308     break; // try-block
1309 }
1310 // If we reach this point, nothing was found and $path is false.
1311 } while (false);
1312
1313 if (!$path) {
1314     addWarning(PDERROR_EXAMPLE_NOT_FOUND, $original_path);
1315     $this-> _title = 'example not found';
1316     $this-> path = false;
1317 } else {
1318     $this-> _title = ($title) ? $title : 'example';
1319     // make a unique html-filename but avoid it to get too long.
1320     $uniqueFileName = str_replace(array(':', DIRECTORY_SEPARATOR, '/'), '_', $path);
1321     $uniqueFileName = substr($uniqueFileName, -50) . '_' . md5($path);
1322     $this-> path = $uniqueFileName;
1323
1324     $f = @fopen($path, 'r');
1325     if ($f) {
1326         $example = fread($f, filesize($path));
1327         if ($tokenizer_ext) {
1328             $obj = new phpDocumentorTWordParser;
1329             $obj-> setup($example);
1330             $this-> source = $obj-> getFileSource();
1331             $this-> origsource = $example;
1332             unset($obj);
1333         } else {
1334             $this-> source = $example;
1335         }
1336     }
1337 }
1338 }
1339 }
1340
1341 /**
1342 * convert the source code
1343 *
1344 * @param Converter &$c      the output converter
1345 *
1346 * @return void
1347 * @uses phpDocumentor_HighlightParser highlights source code

```

```

1348 * @uses writeSource()
1349 * @todo CS cleanup - rename to convertSource for camelCase rule
1350 * @todo what's up with all the "return" statements?
1351 *       can they all be removed?
1352 */
1353 function ConvertSource(& $c)
1354 {
1355     $this-> writeSource($c, $c-> ProgramExample($this-> source, true, null,
1356                                         null, null, null, $this-> origsource));
1357     return;
1358     $parser = new phpDocumentor_HighlightParser;
1359     $return = '';
1360     $return = $parser-> parse($this-> source, $c);
1361     $this-> writeSource($c, $return);
1362 }
1363 /**
1364 * have the output converter write the source code
1365 *
1366 * @param Converter &$c      the output converter
1367 * @param string    $source highlighted source code
1368 *
1369 * @return void
1370 * @access private
1371 * @uses Converter::writeExample() writes final example out
1372 */
1373 function writeSource(& $c, $source)
1374 {
1375     if ($this-> path) {
1376         $c-> writeExample($this-> _title, $this-> path, $source);
1377     }
1378 }
1379 /**
1380 * Retrieve a converter-specific link to the example
1381 *
1382 * @param Converter &$c      the output converter
1383 *
1384 * @return string
1385 * @uses Converter::getExampleLink() retrieve the link to the example
1386 */
1387 function getSourceLink(& $c)
1388 {
1389     if (!$this-> path) return $this-> _title;
1390     return $c-> getExampleLink($this-> path, $this-> _title);
1391 }
1392 }
1393 */
1394 ?>
1395
1396 ?>

```

# File Source for Errors.inc

Documentation for this file is available at [Errors.inc](#)

```
1  <?php
2  /**
3   * Error handling for phpDocumentor
4   *
5   * phpDocumentor :: automatic documentation generator
6   *
7   * PHP versions 4 and 5
8   *
9   * Copyright (c) 2001-2008 Gregory Beaver
10  *
11  * LICENSE:
12  *
13  * This library is free software; you can redistribute it
14  * and/or modify it under the terms of the GNU Lesser General
15  * Public License as published by the Free Software Foundation;
16  * either version 2.1 of the License, or (at your option) any
17  * later version.
18  *
19  * This library is distributed in the hope that it will be useful,
20  * but WITHOUT ANY WARRANTY; without even the implied warranty of
21  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22  * Lesser General Public License for more details.
23  *
24  * You should have received a copy of the GNU Lesser General Public
25  * License along with this library; if not, write to the Free Software
26  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27  *
28  * @category ToolsAndUtilities
29  * @package phpDocumentor
30  * @subpackage Errors
31  * @author Greg Beaver <cellog@php.net>
32  * @copyright 2001-2008 Gregory Beaver
33  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34  * @version CVS: $Id: Errors.inc 253641 2008-02-24 02:35:44Z ashnazg $
35  * @filesource
36  * @link http://www.phpdoc.org
37  * @link http://pear.php.net/PhpDocumentor
38  * @see parserDocBlock, parserInclude, parserPage, parserClass
39  * @see parserDefine, parserFunction, parserMethod, parserVar
40  * @since 0.4
41  * @todo CS cleanup - change package to PhpDocumentor
42  */
43 /**
44  * warning triggered when inheritance could be from more than one class
45  */
46 define("PDERROR_MULTIPLE_PARENT" , 1);
47 /**
48  * warning triggered when parent class doesn't exist
49  */
50 define("PDERROR_PARENT_NOT_FOUND" , 2);
51 /**
52  * warning triggered when an {@inline tag} is not terminated
53  * (no } before the * / ending the comment)
54  */
55 define("PDERROR_UNTERMINATED_INLINE_TAG" , 3);
56 /**
57  * warning triggered when inheritance could be from more than one class
58  */
59 define("PDERROR_CLASS_EXISTS" , 4);
60 /**
61  * warning triggered when inheritance could be from more than one class
62  */
63 define("PDERROR_INHERITANCE_CONFLICT" , 5);
64 /**
65  * warning triggered when a converter is passed to
66  * {@link phpDocumentor_IntermediateParser::addConverter()} that is not a class
67 */
```

```

68 define( "PDERROR_CONVERTER_NOT_FOUND" , 6);
69 /**
70 * warning triggered when a converter is passed to
71 * {@link phpDocumentor_IntermediateParser::addConverter()} that is not a class
72 */
73 define( "PDERROR_NO_CONVERTERS" , 7);
74 /**
75 * warning triggered when the arguments to @access are neither public nor private
76 */
77 define( "PDERROR_ACCESS_WRONG_PARAM" , 8);
78 /**
79 * warning triggered when there are multiple @access tags in a docblock
80 */
81 define( "PDERROR_MULTIPLE_ACCESS_TAGS" , 9);
82 /**
83 * warning triggered when there are multiple @return tags in a docblock
84 */
85 define( "PDERROR_MULTIPLE_RETURN_TAGS" , 10);
86 /**
87 * warning triggered when there are multiple @var tags in a docblock
88 */
89 define( "PDERROR_MULTIPLE_VAR_TAGS" , 11);
90 /**
91 * warning triggered when there are multiple @package tags in a docblock
92 */
93 define( "PDERROR_MULTIPLE_PACKAGE_TAGS" , 12);
94 /**
95 * warning triggered when there are multiple @subpackage tags in a docblock
96 */
97 define( "PDERROR_MULTIPLE_SUBPACKAGE_TAGS" , 13);
98 /**
99 * warning triggered when the package or subpackage name is illegal
100 */
101 define( "PDERROR_ILLEGAL_PACKAGENAME" , 14);
102 /**
103 * warning triggered when there a @package tag is used in a function,
104 * define, method, var or include
105 */
106 define( "PDERROR_OVERRIDDEN_PACKAGE_TAGS" , 15);
107 /**
108 * warning triggered when there a @subpackage tag is used in a function,
109 * define, method, var or include
110 */
111 define( "PDERROR_OVERRIDDEN_SUBPACKAGE_TAGS" , 16);
112 /**
113 * warning triggered when classes in the same package have the same name
114 */
115 define( "PDERROR_CLASS_CONFLICT" , 17);
116 /**
117 * warning triggered when classes in the same package have the same name
118 */
119 define( "PDERROR_UNKNOWN_TAG" , 18);
120 /**
121 * warning triggered when there are multiple @name tags in a docblock
122 */
123 define( "PDERROR_MULTIPLE_NAME_TAGS" , 19);
124 /**
125 * warning triggered when there are multiple @name tags in a docblock
126 * @todo I think this description is a copy/paste that was never updated
127 */
128 define( "PDERROR_PACKAGEOUTPUT_DELETES_PARENT_FILE" , 20);
129 /**
130 * warning triggered when there are multiple @name tags in a docblock
131 * @todo I think this description is a copy/paste that was never updated
132 */
133 define( "PDERROR_GLOBAL_NOT_FOUND" , 21);
134 /**
135 * warning triggered when there are multiple @name tags in a docblock
136 * @todo I think this description is a copy/paste that was never updated
137 */
138 define( "PDERROR_MULTIPLE_GLOBAL_TAGS" , 22);
139 /**
140 * warning triggered when there are multiple @name tags in a docblock
141 * @todo I think this description is a copy/paste that was never updated
142 */
143 define( "PDERROR_MALFORMED_GLOBAL_TAG" , 23);
144 /**
145 * warning triggered when an @ignore tag is used in a DocBlock preceding
146 * a method, variable, include, or global variable
147 */

```

```

148 define("PDERROR_IGNORE_TAG_IGNORED" , 24);
149 /**
150 * warning triggered when a duplicate element is encountered that will be
151 * ignored by the documentor
152 */
153 define("PDERROR_ELEMENT_IGNORED" , 25);
154 /**
155 * warning triggered when an entire page is ignored because of @access private
156 */
157 define("PDERROR_PARSEPRIVATE" , 26);
158 /**
159 * warning triggered when an entire page is ignored because of @access private
160 * @todo I think this description is a copy/paste that was never updated
161 */
162 define("PDERROR_UNKNOWN_COMMANDLINE" , 27);
163 /**
164 * warning triggered when an entire page is ignored because of @access private
165 * @todo I think this description is a copy/paste that was never updated
166 */
167 define("PDERROR_NEED_WHITESPACE" , 28);
168 /**
169 * warning triggered when an entire page is ignored because of @access private
170 * @todo I think this description is a copy/paste that was never updated
171 */
172 define("PDERROR_CLASS_PARENT_NOT_FOUND" , 29);
173 /**
174 * warning triggered when a getClassByPackage is called and can't find the class
175 */
176 define("PDERROR_CLASS_NOT_IN_PACKAGE" , 30);
177 /**
178 * warning triggered when a { @source } inline tag is used in a docblock not
179 * preceding a function
180 */
181 define("PDERROR_SOURCE_TAG_FUNCTION_NOT_FOUND" , 31);
182 /**
183 * warning triggered when a docblock template is never turned off
184 * with /**#@-*/ (no space)
185 */
186 define("PDERROR_DB_TEMPLATE_UNTERMINATED" , 32);
187 /**
188 * warning triggered when a docblock has an unmatched <ol> or <ul>;
189 */
190 define("PDERROR_UNMATCHED_LIST_TAG" , 33);
191 /**
192 * warning triggered when another tag is nested in <b>;
193 * (not allowed in phpDocumentor)
194 */
195 define("PDERROR_CANT_NEST_IN_B" , 34);
196 /**
197 * warning triggered when a docbook tag is not properly matched
198 */
199 define("PDERROR_UNMATCHED_TUTORIAL_TAG" , 35);
200 /**
201 * warning triggered when an inline tag is found inside an xml tag name
202 * in a package page
203 */
204 define("PDERROR_CANT_HAVE_INLINE_IN_TAGNAME" , 36);
205 /**
206 * warning triggered when a tutorial is referenced
207 * via @tutorial/{ @tutorial} and is not found
208 */
209 define("PDERROR_TUTORIAL_NOT_FOUND" , 37);
210 /**
211 * warning triggered when a tutorial lists itself as a child tutorial
212 */
213 define("PDERROR_TUTORIAL_IS_OWN_CHILD" , 38);
214 /**
215 * warning triggered when a tutorial's child lists the parent tutorial
216 * as a child tutorial
217 */
218 define("PDERROR_TUTORIAL_IS_OWN_GRANDPA" , 39);
219 /**
220 * warning triggered when a tutorial's child in the .ini file doesn't exist in the
221 * package and subpackage of the parent
222 */
223 define("PDERROR_CHILD_TUTORIAL_NOT_FOUND" , 40);
224 /**
225 * warning triggered when a <pdffunction:funcname /> tag is used in the PDF
226 * Converter and no funcname is present (<pdffunction: />
227 */

```

```

228 define("PDERROR_PDFFUNCTION_NO_FUNC" , 41);
229 /**
230 * warning triggered when a <pdffunction:funcname /> tag is used in the PDF
231 * Converter and funcname is not a {@link Cezpdf} method
232 */
233 define("PDERROR_PDF_METHOD_DOESNT_EXIST" , 42);
234 /**
235 * warning triggered when a <pdffunction:funcname arg=$tempvar/> tag
236 * is used in the PDF
237 * Converter and "tempvar" is not set from the return of a previous pdffunction tag
238 */
239 define("PDERROR_PDF_TEMPVAR_DOESNT_EXIST" , 43);
240 /**
241 * warning triggered when a subsection's title is asked for, but the subsection
242 * is not found
243 */
244 define("PDERROR_TUTORIAL_SUBSECTION_NOT_FOUND" , 44);
245 /**
246 * warning triggered when a subsection's title is asked for, but the subsection
247 * is not found
248 */
249 define("PDERROR_UNTERMINATED_ATTRIB" , 45);
250 /**
251 * warning triggered when no @package tag is used in a page-level
252 * or class-level DocBlock
253 */
254 define("PDERROR_NO_PACKAGE_TAG" , 46);
255 /**
256 * warning triggered when no @access private tag is used in a
257 * global variable/method/var with _ as first char in name
258 * and --pear was specified
259 */
260 define("PDERROR_PRIVATE_ASSUMED" , 47);
261 /**
262 * warning triggered when an example's path from @example /path/to/example.php
263 * is not found
264 */
265 define("PDERROR_EXAMPLE_NOT_FOUND" , 48);
266 /**
267 * warning triggered when an example's path from @example /path/to/example.php
268 * is not found
269 */
270 define("PDERROR_NO_CONVERTER_HANDLER" , 49);
271 /**
272 * warning triggered when an example's path from @example /path/to/example.php
273 * is not found
274 */
275 define("PDERROR_INLINETAG_IN_SEE" , 50);
276 /**
277 * warning triggered when an id attribute in a tutorial docbook tag is not
278 * an {@id} inline tag
279 */
280 define("PDERROR_ID_MUST_BE_INLINE" , 51);
281 /**
282 * warning triggered when an {@internal}} tag is not closed
283 */
284 define("PDERROR_INTERNAL_NOT_CLOSED" , 52);
285 /**
286 * warning triggered when an {@source} tag is found in a short description
287 */
288 define("PDERROR_SOURCE_TAG_IGNORED" , 53);
289 /**
290 * warning triggered when a child converter doesn't override
291 * getFormattedClassTrees()
292 */
293 define("PDERROR_CONVERTER_OVR_GFCT" , 54);
294 /**
295 * warning triggered when a package is already associated with a category, and
296 * a new association is found
297 */
298 define("PDERROR_PACKAGECAT_SET" , 55);
299 /**
300 * warning triggered when text in a docblock list is not contained in
301 * an <> opening tag
302 */
303 define("PDERROR_TEXT_OUTSIDE_LI" , 56);
304 /**
305 * warning triggered when a DocBlock html tag is unclosed
306 */
307 define("PDERROR_UNCLOSED_TAG" , 57);

```

```

308 /**
309  * warning triggered by @filesource, if PHP < 4.3.0
310 */
311 define("PDERROR_TAG_NOT_HANDLED" , 58);
312 /**
313  * warning triggered by sourcecode="on", if PHP < 4.3.0
314 */
315 define("PDERROR_SOURCECODE_IGNORED" , 59);
316 /**
317  * warning triggered by an empty tag
318 */
319 define("PDERROR_MALFORMED_TAG" , 60);
320 /**
321  * warning triggered by more than 1 @category tag
322 */
323 define("PDERROR_MULTIPLE_CATEGORY_TAGS" , 61);
324 /**
325  * warning triggered by {@inheritDoc} in a non-inheritable situation
326 */
327 define("PDERROR_INHERITDOC_DONT_WORK_HERE" , 62);
328 /**
329  * warning triggered by @example path/to/example with no title
330 */
331 define("PDERROR_EMPTY_EXAMPLE_TITLE" , 63);
332 /**
333  * warning triggered by non-existent template directory
334 */
335 define("PDERROR_TEMPLATEDIR_DOESNT_EXIST" , 64);
336 /**
337  * warning triggered by an unterminated entity in a tutorial
338 */
339 define("PDERROR_UNTERMINATED_ENTITY" , 65);
340 /**
341  * warning triggered by an unterminated entity in a tutorial
342 */
343 define("PDERROR_BEAUTIFYING_FAILED" , 66);
344 /**
345  * warning triggered by a function with no name
346 */
347 /**
348  * function ($params)
349  * {
350  * }
351  */
352 /**
353  * triggers this error
354 */
355 define("PDERROR_FUNCTION_HAS_NONAME" , 67);
356 /**
357  * warning triggered by a page-level docblock preceding a source element
358 */
359 /**
360  * <code>
361  *   * Page-level DocBlock
362  *   * @package pagepackage
363  *   * {@*}
364  *   * include 'file.php';
365  * </code>
366 */
367 define("PDERROR_DOCBLOCK_CONFLICT" , 68);
368 /**
369  * warning triggered when a file does not contain a page-level docblock
370 */
371 define("PDERROR_NO_PAGE_LEVELDOCBLOCK" , 69);
372 /**
373  * warning triggered when the first docblock in a file with a @package tag
374  * precedes a class. In this case, the class gets the docblock.
375 */
376 define("PDERROR_DOCBLOCK_Goes_CLASS" , 70);
377 /**
378  * warning triggered in tutorial parsing if there is a missing {@id} inline tag
379 */
380 define("PDERROR_NO_DOCBOOK_ID" , 71);
381 /**
382  * warning triggered if someone brilliant tries "class X extends X {"
383 */
384 define("PDERROR_CANNOT_EXTEND_SELF" , 72);
385 /**
386  * warning triggered by improper "@uses      {@link blah}"
387 */

```

```

388 define("PDERROR_DUMBUSES" , 73);
389 /**
390 * warning triggered if <>ul> is nested inside <>ul> and not
<>li>
391 */
392 define("PDERROR_UL_IN_UL" , 74);
393 /**
394 * warning triggered if a command line option does not have a valid value passed in
395 */
396 define("PDERROR_INVALID_VALUES" , 75);
397 /**
398 * warning triggered when {@internal} is nested inside another {@internal}
399 */
400 define("PDERROR_NESTED_INTERNAL" , 76);
401 /**
402 * warning triggered when @todo is used on an include element
403 */
404 define("PDERROR_NOTODO_INCLUDE" , 77);
405 /**
406 * warning triggered when a class or method hasn't got docblock
407 */
408 define("PDERROR_UNDOCUMENTED_ELEMENT" , 78);
409 /**
410 * warning triggered when any of {@property}, {@property-read},
411 * {@property-write}), or {@method}) tag does not have name
412 */
413 define("PDERROR_MISSING_PROPERTY_TAG_NAME" , 79);
414 /**
415 * warning triggered when the PHP version being used has dangerous bug/behavior
416 */
417 define("PDERROR_DANGEROUS_PHP_BUG_EXISTS" , 80);
418 /**
419 * warning triggered when the alias value in an page-level docblock's @name tag
420 * is the same value as the target filename is it supposed to alias
421 */
422 define("PDERROR_NAME_ALIAS_SAME_AS_TARGET" , 81);
423 /**
424 * warning triggered when the a loop recursion tripwire has been tripped
425 */
426 define("PDERROR_LOOP_RECURSION_LIMIT_REACHED" , 82);
427 /**
428 * Error messages for phpDocumentor parser warnings
429 * @global array $GLOBALS['phpDocumentor_warning_descrip']
430 * @name $phpDocumentor_warning_descrip
431 */
432 $GLOBALS['phpDocumentor_warning_descrip'] =
433 array(
434     PDERROR_MULTIPLE_PARENT =>
435         'Class %s has multiple possible parents, package inheritance aborted'
436     ,
437     PDERROR_PARENT_NOT_FOUND =>
438         'Class %s parent %s not found'
439     ,
440     PDERROR_INHERITANCE_CONFLICT =>
441         'Class %s in file %s has multiple possible parents named %s. .
442         'Cannot resolve name conflict,' . "\n"
443         ' try ignoring a file that contains the conflicting parent class'
444     ,
445     PDERROR_UNKNOWN_TAG =>
446         'Unknown tag "@%s" used'
447     ,
448     PDERROR_IGNORE_TAG_IGNORED =>
449         '@ignore tag used for %s element "%s" will be ignored'
450     ,
451     PDERROR_ELEMENT_IGNORED =>
452         "\n" . 'duplicate %s element "%s" in file %s will be ignored.'
453         .
454         'Use an @ignore tag on the original '
455         'if you want this case to be documented.'
456     ,
457     PDERROR_PARSEPRIVATE =>
458         "entire page %s ignored because of @access private." . "\n"
459         "Choose -pp to enable parsing of private elements"
460     ,
461     PDERROR_CLASS_PARENT_NOT_FOUND =>
462         "class %s in package %s parent not found in @see parent::%s"
463     ,
464     PDERROR_CLASS_NOT_IN_PACKAGE =>
465         "class %s was not found in package %s"

```

```

466
467     PDERROR_DB_TEMPLATE_UNTERMINATED =>
468         'docblock template never terminated with /**#@-*/'
469
470     PDERROR_PDF_METHOD_DOESNT_EXIST =>
471         '<pdffunction:%s /> called, but pdf method "%s" doesn\'t exist'
472
473     PDERROR_TUTORIAL_NOT_FOUND =>
474         "tutorial \"%s\" not found, does it exist?"
475
476     PDERROR_CHILD_TUTORIAL_NOT_FOUND =>
477         'child tutorial "%s" listed in %s not found '
478         'in parent package "%s" subpackage "%s"'
479
480     PDERROR_TUTORIAL_SUBSECTION_NOT_FOUND =>
481         'tutorial %s subsection "%s" doesn\'t exist, '
482         'but its title was asked for'
483
484     PDERROR_NO_PACKAGE_TAG =>
485         'no @package tag was used in a DocBlock for %s %s'
486
487     PDERROR_PRIVATE_ASSUMED =>
488         '%s "%s" is assumed to be @access private because its name '
489         'starts with _, but has no @access tag'
490
491     PDERROR_EXAMPLE_NOT_FOUND =>
492         'example file "%s" does not exist'
493
494     PDERROR_SOURCE_TAG_IGNORED =>
495         '{@source} can only be used in the long description, '
496         'not in the short description: "%s"'
497
498     PDERROR_PACKAGECAT_SET =>
499         'package %s is already in category %s, '
500         'will now replace with category %s'
501
502     PDERROR_SOURCECODE_IGNORED =>
503         'sourcecode command-line option is ignored '
504         'when your PHP build has no tokenizer support'
505
506     PDERROR_INHERITDOC_DONT_WORK_HERE =>
507         '{@inheritdoc} can only be used in the docblock of a child class'
508
509     PDERROR_EMPTY_EXAMPLE_TITLE =>
510         'Example file found at "%s" has no title, using "%s"'
511
512     PDERROR_DOCBLOCK_CONFLICT =>
513         'Page-level DocBlock precedes "%s %s", '
514         'use another DocBlock to document the source element'
515
516     PDERROR_NO_PAGE_LEVELDOCBLOCK =>
517         'File "%s" has no page-level DocBlock, '
518         'use @package in the first DocBlock to create one'
519
520     PDERROR_DOCBLOCK_Goes_CLASS =>
521         'DocBlock would be page-level, but precedes class "%s", '
522         'use another DocBlock to document the file'
523
524     PDERROR_NO_DOCBOOK_ID =>
525         'Tutorial section %s "%s" has no id="@id subsection" tag '
526         '( {@id} for refentry)'
527
528     PDERROR_BEAUTIFYING_FAILED =>
529         'Beautifying failed: %s'
530
531     PDERROR_NOTODO_INCLUDE =>
532         '@todo on an include element is ignored (line %s, file %s)'
533
534     PDERROR_UNDOCUMENTED_ELEMENT =>
535         '%s "%s" has no %s-level DocBlock.'
536
537     PDERROR_MISSING_PROPERTY_TAG_NAME =>
538         '@%s magic tag does not have name, illegal. Ignoring tag "@%s %s %s"'
539
540     PDERROR_NAME_ALIAS_SAME_AS_TARGET =>
541         '@name value is the same as the filename it is supposed to alias'
542
543
544
545 );

```

```

546 //*****
547
548
549
550 /**
551 * Error messages for phpDocumentor parser errors
552 * @global array $GLOBALS['phpDocumentor_error_descrip']
553 * @name $phpDocumentor_error_descrip
554 */
555 $GLOBALS['phpDocumentor_error_descrip'] =
556     array(
557         PDERROR_UNTERMINATED_INLINE_TAG =>
558             'Inline tag {@$s} in tag %s is unterminated, "%s"'
559         ,
560         PDERROR_CLASS_EXISTS =>
561             'Class %s already exists in package "%s"'
562         ,
563         PDERROR_CONVERTER_NOT_FOUND =>
564             'Converter %s specified by --output command-line option is not a class'
565         ,
566         PDERROR_NO_CONVERTERS =>
567             'No Converters have been specified by --output command-line option'
568         ,
569         PDERROR_ACCESS_WRONG_PARAM =>
570             '@access was passed neither "public" nor "private." Was
passed: "%s"'
571         ,
572         PDERROR_MULTIPLE_ACCESS_TAGS =>
573             'DocBlock has multiple @access tags, illegal.
574             ignoring additional tag "@access %s"'
575         ,
576         PDERROR_MULTIPLE_RETURN_TAGS =>
577             'DocBlock has multiple @return tags, illegal.
578             ignoring additional tag "@return %s %s"'
579         ,
580         PDERROR_MULTIPLE_VAR_TAGS =>
581             'DocBlock has multiple @var tags, illegal.
582             ignoring additional tag "@var %s %s"'
583         ,
584         PDERROR_MULTIPLE_PACKAGE_TAGS =>
585             'DocBlock has multiple @package tags, illegal.
586             ignoring additional tag "@package %s"'
587         ,
588         PDERROR_MULTIPLE_SUBPACKAGE_TAGS =>
589             'DocBlock has multiple @subpackage tags, illegal.
590             ignoring additional tag "@subpackage %s"'
591         ,
592         PDERROR_ILLEGAL_PACKAGENAME =>
593             '@%s tag has illegal %s name "%s"'
594         ,
595         PDERROR_OVERRIDDEN_PACKAGE_TAGS =>
596             '%s %s\'s DocBlock has @package tag, illegal.
597             ignoring tag "@package %s"'
598         ,
599         PDERROR_OVERRIDDEN_SUBPACKAGE_TAGS =>
600             '"%s" %s\'s DocBlock has @subpackage tags, illegal.
601             ignoring tag "@subpackage %s"'
602         ,
603         PDERROR_CLASS_CONFLICT =>
604             'class "%s" has multiple declarations in package %s,
605             in file %s and file %s, documentation will have output errors!'
606         ,
607         PDERROR_MULTIPLE_NAME_TAGS =>
608             'DocBlock has multiple @name tags, illegal.
609             ignoring additional tag "@name %s"'
610         ,
611         PDERROR_PACKAGEOUTPUT Deletes_PARENT_FILE =>
612             '-po (packageoutput) option deletes parent file "%s" containing class
613             '%s.''
614             '\n' . Try using --defaultpackagename (-dn) %s to
615             include the parent file in the same package as the class'
616         ,
617         PDERROR_GLOBAL_NOT_FOUND =>
618             'global variable %s specified in @global tag was never found'
619         ,
620         PDERROR_MULTIPLE_GLOBAL_TAGS =>
621             '@global define tag already used for global variable "%s",
622             ignoring @global %s'
623         ,
624         PDERROR_MALFORMED_GLOBAL_TAG =>

```

```

624           'incorrect @global syntax. '
625           'Should be @global vartype $varname or @global vartype description'
626
627 PDERROR_UNKNOWN_COMMANDLINE =>
628           'Unknown command-line option "%s" encountered, use phpdoc -h for help'
629
630 PDERROR_NEED_WHITESPACE =>
631           'phpDocumentor programmer error - wordparser whitespace set to false '
632           'in handleDocBlock, notify developers. You should never see this error'
633
634 PDERROR_SOURCE_TAG_FUNCTION_NOT_FOUND =>
635           '{@source} tag used in a docblock that isn\'t preceding a function'
636
637 PDERROR_UNMATCHED_LIST_TAG =>
638           'unmatched ol or ul tag in DocBlock, parsing will be incorrect'
639
640 PDERROR_CANT_NEST_IN_B =>
641           'Can\'t nest a code, pre, ul, or ol tag in a b tag in '
642           'phpDocumentor DocBlock (%s tag nested)'
643
644 PDERROR_UNMATCHED_TUTORIAL_TAG =>
645           'While parsing extended documentation, "%s" tag was matched '
646           'with "%s" endtag, missing endtag'          ."\n\ttag"
contents:\ "%s\" "
647
648 PDERROR_CANT_HAVE_INLINE_IN_TAGNAME =>
649           'Can\'t have an inline tag inside a package page XML tag!'
650
651 PDERROR_TUTORIAL_IS_OWN_CHILD =>
652           'Tutorial %s lists itself as its own child in %s, illegal'
653
654 PDERROR_TUTORIAL_IS_OWN_GRANDPA =>
655           'Tutorial %s\'s child %s lists %s as its child in %s, illegal'
656
657 PDERROR_PDFFUNCTION_NO_FUNC =>
658           'Invalid pdffunction syntax: "<pdffunction: />", '
659           'should be "<pdffunction:functionname [arg="value"]/>"'
660
661 PDERROR_PDF_TEMPVAR_DOESNT_EXIST =>
662           '<pdffunction:%s arg=%s /> called '
663           'but temporary variable "%s" doesn\'t exist'
664
665 PDERROR_UNTERMINATED_ATTRIB =>
666           'Tutorial tag %s attribute %s is unterminated, current value "%s" '
667
668 PDERROR_NO_CONVERTER_HANDLER =>
669           'Handler for element of type "%s" called, but %s is not a method of %s'
670
671 PDERROR_INLINETAG_IN_SEE =>
672           'Inline tags are not allowed in a @see tag'
673
674 PDERROR_ID_MUST_BE_INLINE =>
675           '<%s id="%s"> must be <%s id="@id %s">'
676
677 PDERROR_INTERNAL_NOT_CLOSED =>
678           '{@internal was never terminated with }}'
679
680 PDERROR_CONVERTER_OVR_GFCT =>
681           'Converter "%s" must override getFormattedClassTrees() but doesn\'t'
682
683 PDERROR_TEXT_OUTSIDE_LI =>
684           'Text cannot be outside of li tag in a DocBlock list, '
685           'parsing will be incorrect'
686
687 PDERROR_UNCLOSED_TAG =>
688           'Unclosed %s tag in DocBlock, parsing will be incorrect'
689
690 PDERROR_TAG_NOT_HANDLED =>
691           '"%s" tag is not available in PHP built without tokenizer support, tag
ignored'
692
693 PDERROR_MALFORMED_TAG =>
694           '"%s" tag was used without any parameters, illegal'
695
696 PDERROR_MULTIPLE_CATEGORY_TAGS =>
697           'package has multiple @category tags, ignoring "@category %s"'
698
699 PDERROR_TEMPLATEDIR_DOESNT_EXIST =>
700           'template directory "%s" does not exist'
701

```

```

702     PDERROR_UNTERMINATED_ENTITY =>
703         'entity &s is unterminated'
704     ,
705     PDERROR_FUNCTION_HAS_NONAME =>
706         'function has no name (PHP error - test your file before parsing!)'
707     ,
708     PDERROR_CANNOT_EXTEND_SELF =>
709         'class %s cannot extend itself - TEST YOUR CODE BEFORE PARSING'
710     ,
711     PDERROR_DUMBUSES =>
712         '@uses can only link to string data'
713     ,
714     PDERROR_UL_IN_UL =>
715         'ul/ol tags cannot be directly nested inside ul/ol, nest inside li'
716     ,
717     PDERROR_INVALID_VALUES =>
718         'command %s was passed "%s" but must be one of %s'
719     ,
720     PDERROR_NESTED_INTERNAL =>
721         '{@internal} cannot be nested inside {@internal}}'
722     ,
723     PDERROR_DANGEROUS_PHP_BUG_EXISTS =>
724         'Dangerous PHP Bug exists in PHP version %s that can be triggered '
725         'by this parse (see PHP Bug #%s and PEAR Bug #%s)'
726     ,
727     PDERROR_LOOP_RECURSION_LIMIT_REACHED =>
728         'An internal loop in PhpDocumentor has reached its preset '
729         'recursion limit, preventing a possible infinite loop condition.'
730     );
731
732 /**
733 * encapsulates warning information
734 *
735 * @category ToolsAndUtilities
736 * @package phpDocumentor
737 * @subpackage Errors
738 * @author Greg Beaver <cellog@php.net>
739 * @copyright 2001-2008 Gregory Beaver
740 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
741 * @version Release: @VER@
742 * @link http://www.phpdoc.org
743 * @link http://pear.php.net/PhpDocumentor
744 * @todo CS cleanup - change package to PhpDocumentor
745 */
746 class RecordWarning
747 {
748     /**
749      * name of global variable that descriptors for this warning/error is kept
750      * @var string
751      */
752     var $type = 'phpDocumentor_warning_descrip';
753     /**
754      * file this error occurred in
755      * @var string
756      */
757     var $file = false;
758     /**
759      * line number of the file this error occurred in
760      * @var integer
761      */
762     var $linenum;
763     /**
764      * error string
765      * @var string
766      */
767     var $data;
768     /**
769      * error number
770      * @see Errors.inc
771      * @var string
772      */
773     var $num;
774     /**
775      * Constructor
776      *
777      * @param string $file filename this error occurred in {@link $file}
778      * @param integer $linenum line number this error occurred on {@link $linenum}
779      * @param integer $num Error number defined in {@link Errors.inc}
780      * @param string $data... variable number of strings, up to 4,
781      */

```

```

782 * @todo CS Cleanup - do I need to add $data to the method signature?
783 *                      to sprintf based on the error number
784 */
785 function RecordWarning($file, $linenum, $num)
786 {
787     $this-> file = $file;
788     $this-> linenum = $linenum;
789     $a = array(' ', ' ', ' ', ' ');
790     if (func_num_args() > 3) {
791         for ($i=3;$i< func_num_args();$i++) {
792             $a[$i - 3] = func_get_arg($i);
793         }
794     }
795     $this-> num = $num;
796     $this-> data =
797         sprintf($GLOBALS[$this-> type][$this-> num], $a[0], $a[1], $a[2], $a[3]);
798     $this-> output();
799 }
800 }
801 /**
802 * prints the warning
803 *
804 * @param string $string the warning to print
805 *
806 * @return void
807 */
808 function output($string = false)
809 {
810     if ($string) {
811         if ($this-> file) {
812             return
813                 "WARNING in $this-> file on line $this-> linenum: $this->
814 > data\n";
815         } else {
816             return "WARNING: $this-> data\n";
817         }
818     }
819     if ($this-> file) {
820         phpDocumentor_out("WARNING in $this-> file "
821                         "on line $this-> linenum: $this-> data\n");
822     } else {
823         phpDocumentor_out("WARNING: $this-> data\n");
824     }
825     flush();
826 }
827 }
828 /**
829 * encapsulates error information
830 *
831 * @category ToolsAndUtilities
832 * @package phpDocumentor
833 * @subpackage Errors
834 * @author Greg Beaver <celog@php.net>
835 * @copyright 2001-2008 Gregory Beaver
836 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
837 * @version Release: @VER@
838 * @link http://www.phpdoc.org
839 * @link http://pear.php.net/PhpDocumentor
840 * @todo CS cleanup - change package to PhpDocumentor
841 */
842 class RecordError extends RecordWarning
843 {
844     /**
845      * name of global variable that descriptors for this warning/error is kept
846      * @var string
847      */
848     var $type = 'phpDocumentor_error_descrip';
849
850     /**
851      * prints the error
852      *
853      * @param string $string the error to print
854      *
855      * @return string
856      */
857     function output($string = false)
858     {
859         if ($string) {

```

```

861         if ($this-> file) {
862             return "\n\tERROR in $this-> file on line $this-> linenum: $this-
863 > data\n";
864         }
865     } else {
866         return "\n\tERROR: $this-> data\n";
867     }
868 }
869 if ($this-> file) {
870     phpDocumentor_out(" \n\tERROR in $this-> file "
871 " on line $this-> linenum: $this-> data\n" );
872 } else {
873     phpDocumentor_out(" \n\tERROR: $this-> data\n" );
874 }
875 flush();
876 }
877 }
878 /**
879 * contains all the errors/warnings
880 *
881 * @category ToolsAndUtilities
882 * @package phpDocumentor
883 * @subpackage Errors
884 * @author Greg Beaver <cellog@php.net>
885 * @copyright 2001-2008 Gregory Beaver
886 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
887 * @version Release: @VER@
888 * @link http://www.phpdoc.org
889 * @link http://pear.php.net/PhpDocumentor
890 * @todo CS cleanup - change package to PhpDocumentor
891 * @see $errors, $warnings
892 */
893
894 class ErrorTracker
895 {
896     /**
897      * array of {@link RecordError}s
898      * @var array
899      */
899     var $errors = array();
900     /**
901      * array of {@link RecordWarning}s
902      * @var array
903      */
903     var $warnings = array();
904     /**
905      * @var string
906      */
906     var $curfile = '';
907     /**
908      * @var integer
909      */
909     var $linenum = 0;
910     /**
911      * index in {@link $errors} of last error triggered
912      * @var integer/false
913      */
913     var $lasterror = false;
914     /**
915      * index in {@link $warnings} of last warning triggered
916      * @var integer/false
917      */
917     var $lastwarning = false;
918     /**
919      * This function subscribes to two events in the Parser
920      * in order to keep track of line number information and file name.
921      *
922      * @param integer $num parser-passed event
923      * (see {@link PHPDOCUMENTOR_EVENT_NEWLINENUM,}
924      * PHPDOCUMENTOR_EVENT_NEWFILE})
925      * @param mixed $data either a line number if $num is
926      * PHPDOCUMENTOR_EVENT_NEWLINENUM or a file name
927      * if $num is PHPDOCUMENTOR_EVENT_NEWFILE
928      *
929      * @return void
930      */

```

```

940     function handleEvent($num,$data)
941     {
942         switch($num) {
943             case PHPDOCUMENTER_EVENT_NEWINENUM :
944                 $this-> linenum = $data;
945                 break;
946
947             case PHPDOCUMENTER_EVENT_NEWFILE :
948                 $this-> linenum = 0;
949                 $this-> curfile = $data;
950                 break;
951
952             case 1000000635 : // debugging
953                 phpDocumentor_out($this-> curfile .
954                     ' has ' . $this-> linenum . ' lines' . "\n" );
955                 flush();
956                 break;
957         }
958     }
959
960 /**
961 * add a new warning to the {@link $warnings} array
962 *
963 * @param integer $num      error number from {@link Errors.inc}
964 * @param string  $data...  up to 4 string parameters to sprintf()
965 *                         into the error string for error number $num
966 *
967 * @return void
968 * @todo CS Cleanup - do I need to add $data to the method signature?
969 */
970 function addWarning($num)
971 {
972     $a = array(' ', ' ', ' ', ' ');
973     if (func_num_args() > 1) {
974         for ($i=1;$i< func_num_args();$i++) {
975             $a[$i - 1] = func_get_arg($i);
976         }
977     }
978     $this-> warnings[] = new RecordWarning($this-> curfile,
979                                         $this-> linenum, $num, $a[0], $a[1], $a[2], $a[3]);
980     $this-> lastwarning = count($this-> warnings) - 1;
981 }
982
983 /**
984 * add a new error to the {@link $errors} array
985 *
986 * @param integer $num      error number from {@link Errors.inc}
987 * @param string  $data...  up to 4 string parameters to sprintf()
988 *                         into the error string for error number $num
989 *
990 * @return void
991 * @todo CS Cleanup - do I need to add $data to the method signature?
992 */
993 function addError($num)
994 {
995     $a = array(' ', ' ', ' ', ' ');
996     if (func_num_args() > 1) {
997         for ($i=1;$i< func_num_args();$i++) {
998             $a[$i - 1] = func_get_arg($i);
999         }
1000    }
1001    $this-> errors[] = new RecordError($this-> curfile,
1002                                         $this-> linenum, $num, $a[0], $a[1], $a[2], $a[3]);
1003    $this-> lasterror = count($this-> errors) - 1;
1004 }
1005
1006 /**
1007 * add a new error to the {@link $errors} array and returns the error string
1008 *
1009 * @param integer $num      error number from {@link Errors.inc}
1010 * @param string  $data...  up to 4 string parameters to sprintf()
1011 *                         into the error string for error number $num
1012 *
1013 * @return void
1014 * @todo CS Cleanup - do I need to add $data to the method signature?
1015 */
1016 function addErrorReturn($num)
1017 {
1018     $a = array(' ', ' ', ' ', ' ');
1019     if (func_num_args() > 1) {

```

```

1020         for ($i=1;$i< func_num_args();$i++) {
1021             $a[$i - 1] = func_get_arg($i);
1022         }
1023     }
1024     $this-> errors[] = new RecordError($this-> curfile,
1025                                         $this-> linenum, $num, $a[0], $a[1], $a[2], $a[3], false);
1026     $this-> lasterror = count($this-> errors) - 1;
1027 }
1028 /**
1029 * Get sorted array of all warnings in parsing/conversion
1030 *
1031 * @return array
1032 */
1033 function & returnWarnings()
1034 {
1035     usort($this-> warnings, array($this, "errorsort"));
1036     return $this-> warnings;
1037 }
1038
1039 /**
1040 * Get sorted array of all non-fatal errors in parsing/conversion
1041 *
1042 * @return array
1043 */
1044 function & returnErrors()
1045 {
1046     usort($this-> errors, array($this, "errorsort"));
1047     return $this-> errors;
1048 }
1049
1050 /**
1051 * sort two errors
1052 *
1053 * @param RecordError|RecordWarning $a the first error/warning
1054 * @param RecordError|RecordWarning $b the second error/warning
1055 *
1056 * @return int
1057 * @access private
1058 */
1059 function errorsort($a, $b)
1060 {
1061     if (!$a-> file) return -1;
1062     if (!$b-> file) return 1;
1063     if ($a-> file == $b-> file) {
1064         if ($a-> linenum == $b-> linenum) return 0;
1065         if ($a-> linenum < $b-> linenum) return -1;
1066         return 1;
1067     }
1068     return strnatcasecmp($a-> file, $b-> file);
1069 }
1070
1071 /**
1072 * Get the error message of the last error
1073 *
1074 * @return string
1075 */
1076 function returnLastError()
1077 {
1078     return $this-> errors[$this-> lasterror]-> output(true);
1079 }
1080
1081 /**
1082 * Get the warning message of the last warning
1083 *
1084 * @return string
1085 */
1086 function returnLastWarning()
1087 {
1088     return $this-> warnings[$this-> lastwarning]-> output(true);
1089 }
1090
1091 }
1092
1093 /**
1094 * @global ErrorTracker $GLOBALS['phpDocumentor_errors']
1095 * @name $phpDocumentor_errors
1096 */
1097 $GLOBALS['phpDocumentor_errors'] = new ErrorTracker;
1098
1099 /**

```

```

1100 * add an Error
1101 *
1102 * @param integer $num      error number from {@link Errors.inc}
1103 * @param string  $data...   up to 4 string parameters to sprintf()
1104 *                           into the error string for error number $num
1105 *
1106 * @return void
1107 * @see ErrorTracker::addError()
1108 * @todo CS Cleanup - do I need to add $data to the method signature?
1109 */
1110 function addError($num)
1111 {
1112     global $phpDocumentor_errors;
1113     $a = array(' ', ' ', ' ', ' ');
1114     if ($func_num_args() > 1) {
1115         for ($i=1;$i< $func_num_args();$i++) {
1116             $a[$i - 1] = func_get_arg($i);
1117         }
1118     }
1119     $phpDocumentor_errors-> addError($num, $a[0], $a[1], $a[2], $a[3]);
1120 }
1121 /**
1122 * like {@link addError()} but exits parsing
1123 *
1124 * @param integer $num      error number from {@link Errors.inc}
1125 * @param string  $data...   up to 4 string parameters to sprintf()
1126 *                           into the error string for error number $num
1127 *
1128 * @return void
1129 * @global ErrorTracker repository for all errors generated by phpDocumentor
1130 * @see ErrorTracker::addError()
1131 * @todo CS Cleanup - do I need to add $data to the method signature?
1132 */
1133 function addErrorDie($num)
1134 {
1135     global $phpDocumentor_errors;
1136     $a = array(' ', ' ', ' ', ' ');
1137     if ($func_num_args() > 1) {
1138         for ($i=1;$i< $func_num_args();$i++) {
1139             $a[$i - 1] = func_get_arg($i);
1140         }
1141     }
1142     $phpDocumentor_errors-> addErrorReturn($num, $a[0], $a[1], $a[2], $a[3]);
1143     echo $phpDocumentor_errors-> returnLastError();
1144     die(1);
1145 }
1146 /**
1147 * add a Warning
1148 *
1149 * @param integer $num      warning number from {@link Errors.inc}
1150 * @param string  $data...   up to 4 string parameters to sprintf()
1151 *                           into the error string for error number $num
1152 *
1153 * @return void
1154 * @global ErrorTracker repository for all errors generated by phpDocumentor
1155 * @see ErrorTracker::addWarning()
1156 * @todo CS Cleanup - do I need to add $data to the method signature?
1157 */
1158 function addWarning($num)
1159 {
1160     global $phpDocumentor_errors;
1161     $a = array(' ', ' ', ' ', ' ');
1162     if ($func_num_args() > 1) {
1163         for ($i=1;$i< $func_num_args();$i++) {
1164             $a[$i - 1] = func_get_arg($i);
1165         }
1166     }
1167     $phpDocumentor_errors-> addWarning($num, $a[0], $a[1], $a[2], $a[3]);
1168 }
1169 */
1170 ?>
```

# File Source for InlineTags.inc

Documentation for this file is available at [InlineTags.inc](#)

```
1  <?php
2  /**
3  * All abstract representations of inline tags are in this file
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2002-2008 Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @category ToolsAndUtilities
29 * @package phpDocumentor
30 * @subpackage InlineTags
31 * @author Gregory Beaver <cellog@php.net>
32 * @copyright 2002-2008 Gregory Beaver
33 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34 * @version CVS: $Id: InlineTags.inc 286921 2009-08-08 05:01:24Z ashnazg $
35 * @filesource
36 * @link http://www.phpdoc.org
37 * @link http://pear.php.net/PhpDocumentor
38 * @since separate file since 1.2
39 * @todo CS cleanup - change package to PhpDocumentor
40 */
41 /**
42 * Use this element to represent an {@}inline tag} like {@}link}
43 *
44 * @category ToolsAndUtilities
45 * @package phpDocumentor
46 * @subpackage InlineTags
47 * @author Gregory Beaver <cellog@php.net>
48 * @copyright 2002-2008 Gregory Beaver
49 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
50 * @version Release: @VER@
51 * @filesource
52 * @link http://www.phpdoc.org
53 * @link http://pear.php.net/PhpDocumentor
54 * @see parserStringWithInlineTags
55 * @since 1.0rc1
56 * @tutorial inlinetags.pkg
57 * @todo CS cleanup - change package to PhpDocumentor
58 * @todo CS cleanup - change classname to PhpDocumentor_*
59 */
60 class parserInlineTag extends parserBase
61 {
62 /**
63 * Element type
64 *
65 * Type is used by many functions to skip the hassle of
66 */
67
```

```

68     * <code>
69     * if phpDocumentor_get_class($blah) == 'parserBlah'
70     * </code>
71     * always "inlinetag"
72     * @var string
73     */
74 var $type = 'inlinetag';
75 /**
76     * the name of the inline tag (like link)
77     * @var string
78     */
79 var $inlinetype = '';
80
81 /**
82     * sets up the tag
83     *
84     * @param string $type tag type (example: link)
85     * @param string $value tag value (example: what to link to)
86     */
87 function parserInlineTag($type, $value)
88 {
89     $this-> inlinetype = $type;
90     $this-> value = trim($value);
91 }
92
93 /**
94     * get length of the tag
95     *
96     * @return integer length of the tag
97     * @todo CS cleanup - rename to strlen for camelCase rule
98     */
99 function Strlen()
100 {
101     // fix 1203451
102     if (is_array($this-> value)) {
103         return array_reduce(create_function('$a, $b', 'return $a + strlen($b);'),
104                             + count($this-> value));
105     }
106     return strlen($this-> value);
107 }
108
109 /**
110     * always gets an empty string
111     *
112     * @return string always '', used by {@link Parser::handleDocBlock()} to
113     * calculate the short description of a DocBlock
114     * @see parserStringWithInlineTags::getString()
115     * @see parserStringWithInlineTags::trimmedStrlen()
116     */
117 function getString()
118 {
119     return '';
120 }
121
122 /**
123     * represents inline links
124     *
125     * @category ToolsAndUtilities
126     * @package phpDocumentor
127     * @subpackage InlineTags
128     * @author Gregory Beaver <cellog@php.net>
129     * @copyright 2002-2008 Gregory Beaver
130     * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
131     * @version Release: @VER@
132     * @filesource
133     * @link http://www.phpdoc.org
134     * @link http://pear.php.net/PhpDocumentor
135     * @see parserStringWithInlineTags
136     * @since 1.0rc1
137     * @tutorial tags.inlinelink.pkg
138     * @todo CS cleanup - change package to PhpDocumentor
139     * @todo CS cleanup - change classname to PhpDocumentor_*
140     */
141 class parserLinkInlineTag extends parserInlineTag
142 {
143     /**
144     * text to display in the link, can be different from the link for standard
145     * links like websites
146     * @var string

```

```

148 */
149 var $linktext = '';
150
151 /**
152 * sets up the tag
153 *
154 * @param string $link stored in $value, see {@link parserBase::$value}
155 * @param string $text see {@link $linktext}
156 */
157 function parserLinkInlineTag($link, $text)
158 {
159     if (strpos($link, ',')) {
160         $link = explode(',', $link);
161         parserInlineTag::parserInlineTag('link', '');
162         $this-> value = $link;
163     } else {
164         parserInlineTag::parserInlineTag('link', $link);
165     }
166     $this-> linktext = trim($text);
167 }
168
169 /**
170 * calls the output conversion
171 *
172 * @param Converter &$c converter used to change the abstract link
173 *                      into text for display
174 *
175 * @return false/string returns the converted link or false
176 *                     if not converted successfully
177 * @todo CS cleanup - rename to convert for camelCase rule
178 */
179 function Convert(& $c)
180 {
181     if (is_array($this-> value)) {
182         $ret = '';
183         foreach ($this-> value as $text) {
184             if (!empty($ret)) {
185                 $ret .= ', ';
186             }
187             $ret .= $this-> ConvertPart($c, trim($text));
188         }
189         return $ret;
190     } else {
191         return $this-> ConvertPart($c, $this-> value);
192     }
193 }
194
195 /**
196 * convert part of the tag
197 *
198 * @param Converter &$c the output converter
199 * @param string $value the tag value
200 *
201 * @return string
202 * @todo CS cleanup - rename to convertPart for camelCase rule
203 */
204 function ConvertPart(& $c, $value)
205 {
206     if (strpos($value, '://')) || (strpos($value, 'mailto:') === 0)) {
207         if (strpos($value, ',')) {
208             $value = explode(',', $value);
209             $link = array_shift($value);
210             $text = join(' ', $value);
211         } else {
212             $link = $value;
213             $text = $this-> linktext;
214         }
215         return $c-> returnLink($link, htmlspecialchars($text));
216     } else {
217         $savevalue = $value;
218         $descrip = false;
219         if (strpos(trim($value), ' ')) {
220             $v = preg_split('/\s/', trim($value));
221             if (in_array(strtolower($v[0]), array('object', 'function'))) {
222                 if (!isset($v[1])
223                     || (isset($v[1]) && strlen($v[1])
224                         && ! in_array($v[1]{0}, array('$', '&'))
225                         && $v[1] != '#commanana####'
226                     )
227             ) {

```

```

228                     $vsave = $v[0];
229                     array_shift($v);
230                     $v[0] = $vsave . ' ' . $v[0];
231                 }
232             }
233             $value = $c->    getLink($v[0]);
234             array_shift($v);
235             $descrip = join($v, ' ');
236             $descrip = str_replace('###commanana####', ' ', $descrip);
237         } else {
238             $value = $c->    getLink($value);
239         }
240         if (is_string($value)) {
241             // feature 564991
242             if (strpos($value, '://')) {
243                 // php function
244                 return $c->    returnLink($value, $descrip ? $descrip :
245                                         str_replace('PHP MANUAL#', ' ', $value));
246             }
247             return $value;
248         }
249         if (!$descrip) {
250             $descrip = $c->    type_adjust($savevalue);
251         }
252         if (is_object($value)) {
253             return $c->    returnSee($value, $descrip);
254         }
255         return $savevalue;
256     }
257 }
258 }
259 /**
260 * Represents inline links to external tutorial documentation
261 *
262 * @category ToolsAndUtilities
263 * @package phpDocumentor
264 * @subpackage InlineTags
265 * @author Gregory Beaver <cellog@php.net>
266 * @copyright 2002-2008 Gregory Beaver
267 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
268 * @version Release: @VER@
269 * @filesource
270 * @link http://www.phpdoc.org
271 * @link http://pear.php.net/PhpDocumentor
272 * @see parserStringWithInlineTags
273 * @tutorial tags.inlinetutorial.pkg
274 * @todo CS cleanup - change package to PhpDocumentor
275 * @todo CS cleanup - change classname to PhpDocumentor_*
276 */
277
278 class parserTutorialInlineTag extends parserLinkInlineTag
279 {
280     /**
281      * constructor
282      *
283      * @param string $link stored in $value, see {@link parserBase::$value}
284      * @param string $text see {@link $linktext}
285      */
286     function parserTutorialInlineTag($link,$text)
287     {
288         parserInlineTag::parserInlineTag('tutorial', $link);
289         $this-> linktext = trim($text);
290     }
291
292     /**
293      * convert part of the tag
294      *
295      * @param Converter &$c      converter used to change the abstract link
296                               into text for display
297      *
298      * @return mixed returns the converted link
299                               or false if not converted successfully
300      * @todo CS cleanup - rename to convert for camelCase rule
301      */
302     function Convert(& $c)
303     {
304         $descrip = false;
305         if (strpos($this-> value, ',') === false) {
306             if (strpos(trim($this-> value), ' ') != false) {
307                 $v      = explode(' ', trim($this-> value));

```

```

308             $value = $c-> getTutorialLink($v[0]);
309             array_shift($v);
310             $descrip = join($v, ' ');
311         } else {
312             $value = $c-> getTutorialLink($this-> value);
313         }
314     } else {
315         $vals = explode(',', $this-> value);
316         $descrip = array();
317         foreach ($vals as $val) {
318             $val = trim($val);
319             if (strpos($val, ',')) {
320                 $v = explode(',', $val);
321                 $value[] = $c-> getTutorialLink($v[0]);
322                 array_shift($v);
323                 $descrip[] = join($v, ' ');
324             } else {
325                 $value[] = $c-> getTutorialLink($val);
326                 $descrip[] = false;
327             }
328         }
329     }
330     if (is_string($value)) {
331         return $value;
332     }
333     if (is_object($value)) {
334         return $c-> returnSee($value, $descrip);
335     }
336     /*
337      * getLink parsed a comma-delimited list of linked thingies,
338      * add the commas back in
339      */
340     if (is_array($value)) {
341         $a = '';
342         foreach ($value as $i => $bub) {
343             if (!empty($a)) {
344                 $a .= ', ';
345             }
346             if (is_string($value[$i])) {
347                 $a .= $value[$i];
348             }
349             if (is_object($value[$i])) {
350                 $a .= $c-> returnSee($value[$i], $descrip[$i]);
351             }
352         }
353         return $a;
354     }
355     return false;
356 }
357 }
358 /**
359  * represents inline source tag, used for function/method source
360  *
361  * @category ToolsAndUtilities
362  * @package phpDocumentor
363  * @subpackage InlineTags
364  * @author Gregory Beaver <cellog@php.net>
365  * @copyright 2002-2008 Gregory Beaver
366  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
367  * @version Release: @VER@
368  * @filesource
369  * @link http://www.phpdoc.org
370  * @link http://pear.php.net/PhpDocumentor
371  * @see parserStringWithInlineTags
372  * @tutorial tags.inlinesource.pkg
373  * @todo CS cleanup - change package to PhpDocumentor
374  * @todo CS cleanup - change classname to PhpDocumentor_*
375  */
376 class parserSourceInlineTag extends parserInlineTag
377 {
378     /**
379      * always 'source'
380      * @var string
381      */
382     var $inlinetype = 'source';
383     /**
384      * First line of source code to display
385      * @var integer
386      * @see $end

```

```

388 */
389 var $start = 1;
390 /**
391 * Last line to display
392 * @var '*'|integerIf '*' then the whole source will be used, otherwise
393 * the {@link $start} to $end line numbers will be displayed
394 */
395 var $end = '*';
396 /**
397 * tokenized source organized by line numbers for php 4.3.0+, the old
398 * {@source} tag used a string
399 * @var string|array
400 */
401 var $source = false;
402 /**#@+ @access private */
403 /** @var string|false*/
404 var $_class;
405 /**#@-*/
406
407 /**
408 * constructor
409 *
410 * @param string $value format "start [end]",
411 * where start and end are line numbers
412 * with the end line number optional
413 */
414 function parserSourceInlineTag($value)
415 {
416     parserInlineTag::parserInlineTag('source', '');
417     preg_match('/^([0-9]+)\W([0-9]*)$/i', trim($value), $match);
418     if (!count($match)) {
419         preg_match('/^([0-9]+)$i', trim($value), $match);
420         if (count($match)) {
421             $this-> start = (int) $match[1];
422         }
423     } else {
424         $this-> start = (int) $match[1];
425         $this-> end = (int) $match[2];
426     }
427 }
428 /**
429 * only used to determine blank lines. {@source} will not be blank, probably
430 *
431 * @return int
432 */
433 function Strlen()
434 {
435     return 1;
436 }
437
438 /**
439 * gets the source string
440 *
441 * @return string
442 */
443 function getString()
444 {
445     return '{@source}';
446 }
447
448 /**
449 * sets the source tag's value
450 *
451 * @param string|array $source source code
452 * @param string|bool $class class name if this is a method,
453 * boolean in php 4.3.0,
454 * if this is a method this will be true
455 *
456 * @return void
457 */
458 function setSource($source, $class = false)
459 {
460     if (is_array($source)) {
461         $this-> _class = $class;
462         $this-> source = $source;
463     } else {
464         $source = strstr($source, 'function');
465         $pos = strrpos($source, '}');
466         $this-> source = substr($source, 0, $pos + 1);

```

```

468     }
469 }
470 /**
471 * convert the tag
472 *
473 * @param Converter &$c      the output converter object
474 *
475 * @return string
476 * @uses stringConvert() in PHP 4.2.3-, this method is used to convert
477 * @uses arrayConvert() in PHP 4.3.0+, this method is used to convert
478 * @todo CS cleanup - rename to convert for camelCase rule
479 */
480 function Convert(&    $c)
481 {
482     if (is_string($this->  source)) {
483         return $this->  stringConvert($c);
484     }
485     return $this->  arrayConvert($c);
486 }
487 }
488 /**
489 * converter helper used in PHP 4.3.0+
490 *
491 * @param Converter &$c      the output converter object
492 *
493 * @return string
494 * @uses phpDocumentor_HighlightParser Parses the tokenized source
495 */
496 function arrayConvert(&    $c)
497 {
498     $source = $this->  source;
499     if ($this->  end != '*') {
500         $source = array_slice($this->  source, 0, $this->  end + $this->  start - 1);
501     }
502     $start = $this->  start - 1;
503     if ($start <  0) {
504         $start = 0;
505     }
506     return $c->  ProgramExample($source, true, true, $this->  _class, $start);
507 }
508 }
509 /**
510 * converter helper used in PHP 4.2.3-
511 *
512 * @param Converter &$c      the output converter object
513 *
514 * @return string
515 * @uses Converter::unmangle() remove the extraneous stuff from
516 * @link highlight_string()
517 * @deprecated in favor of PHP 4.3.0+ {@link arrayConvert()}
518 */
519 function stringConvert(&    $c)
520 {
521     $source = highlight_string('<?php ' . $this->  source . ' ?>', true);
522     $source = '<code>' . substr($source, strlen('<code><font color="#000000">'));
523     $source = str_replace('<br />', ) - 1);
524     <font color="#0000CC">&lt;?php&nbsp;</font>' . $source . '</code><font color="#0000CC">?&gt;</font>';
525     $source = str_replace('}' . $source . '<code>' . $source . ', $source);
526     if ($this->  start || ($this->  end != '*')) {
527         $source = explode('<br />', $source);
528         $start = $this->  start;
529         if ($this->  end != '*') {
530             $source = array_slice($source, $start - 1, $this->  end - $start + 1);
531         } else {
532             $source = array_slice($source, $start - 1);
533         }
534         $source = implode($source, '<br />');
535         if ($start >  0) {
536             $source = "    <code>      $source" . ;
537         }
538         if ($this->  end != '*') {
539             $source = "      $source</code>" . ;
540         }
541     }
542     $source = $c->  unmangle($source, $this->  source);
543     return $source;
544 }
545 }

```

```

546     }
547
548 /**
549 * Represents the example inline tag, used to display an example file
550 * inside a docblock or tutorial
551 *
552 * @category ToolsAndUtilities
553 * @package phpDocumentor
554 * @subpackage InlineTags
555 * @author Gregory Beaver <cellog@php.net>
556 * @copyright 2002-2008 Gregory Beaver
557 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
558 * @version Release: @VER@
559 * @filesource
560 * @link http://www.phpdoc.org
561 * @link http://pear.php.net/PhpDocumentor
562 * @see parserStringWithInlineTags
563 * @tutorial tags.inlineexample.pkg
564 * @todo CS cleanup - change package to PhpDocumentor
565 * @todo CS cleanup - change classname to PhpDocumentor_*
566 */
567 class parserExampleInlineTag extends parserSourceInlineTag
568 {
569     /**
570      * constructor
571      *
572      * @param string $value      format "filepath[ start [end]]"
573      *                           where start and end are line numbers
574      *                           with the end line number optional
575      * @param string $current_path full path to the current file,
576      *                           used to check relative directory locations
577      * @param bool   $isTutorial if true, then this is in a tutorial
578      *
579      * @return mixed
580      * @todo replace tokenizer_ext constant with TOKENIZER_EXT for CS rule
581      */
582     function parserExampleInlineTag($value, $current_path, $isTutorial = false)
583     {
584         global $_phpDocumentor_setting;
585         parserInlineTag::parserInlineTag('example', '');
586         $path    = false;
587         $tagValue = trim($value);
588         $path    = $isAbsPath = $pathOnly = $fileName = $fileExt
589         = $original_path = $title = false;
590         do {
591             // make sure the format is stuff.ext startline[ endline]
592             if (!preg_match('`(.*)\.(\\w*)\\s(.*)``', $tagValue, $match)) {
593                 // or format is stuff.ext
594                 if (!preg_match('`^(.*).(\w*)$``', $tagValue, $match)) {
595                     // Murphy: Some funny path was given
596                     $original_path = $tagValue; // used for error output
597                     break; // try-block
598                 }
599             }
600             if (strlen($match[1]) === 0) {
601                 // Murphy: Some funny path was given
602                 $original_path = $tagValue; // used for error output
603                 break; // try-block
604             }
605             $fileExt = $match[2];
606             if (isset($match[3])) {
607                 $lines    = explode(' ', trim($match[3]));
608                 $this-> start = (int) $lines[0];
609                 if (isset($lines[1])) {
610                     $this-> end = (int) $lines[1];
611                 }
612             }
613             // Replace windows '\' the path.
614             $pathTmp = str_replace('\\', '/', $match[1]);
615
616             // Is there a path and a file or is it just a file?
617             if (strpos($pathTmp, '/') === false) {
618                 // No path part
619                 $pathOnly = '';
620                 $fileName = $pathTmp . '.' . $fileExt;
621             } else {
622                 // split the path on the last directory, find the filename
623                 $splitPos = strrpos($pathTmp, '/');
624                 $pathOnly = substr($match[1], 0, $splitPos+1);
625                 $fileName = substr($match[1], $splitPos+1) . '.' . $fileExt;
626             }
627         }
628     }
629 }

```

```

626 // Is the path absolute? (i.e. does it start like an absolute path?)
627 if ('/' === $pathTmp[0]) || preg_match('/^\\w*:\\i', $pathTmp)) {
628     // works for both windows 'C:' and URLs like 'http://'
629     $isAbsPath = true; // Yes
630 }
631 }
632
633 $original_path = $pathOnly . $fileName;
634
635 // Now look for the file starting with abs. path.
636 if ($isAbsPath) {
637     // remove any weirdities like ../file.ext
638     $tmp = realpath($original_path);
639     if ($tmp && is_file($tmp)) {
640         $path = $tmp;
641     }
642     /*
643      * Always break if abs. path was detected,
644      * even if file was not found.
645      */
646     break; // try-block
647 }
648
649 // Search for the example file some standard places
650 // 1) Look if the ini-var examplesdir is set and look there ...
651 if (isset($phpDocumentor_setting['examplesdir'])) {
652     $tmp = realpath($phpDocumentor_setting['examplesdir']
653     . PATH_DELIMITER . $original_path);
654     if ($tmp && is_file($tmp)) {
655         $path = $tmp; // Yo! found it :)
656         break; // try-block
657     }
658 }
659
660 // 2) Then try to look for an 'example/-dir
661 // below the *currently* parsed file ...
662 if (!empty($current_path)) {
663     $tmp = realpath(dirname($current_path) . PATH_DELIMITER . 'examples'
664     . PATH_DELIMITER . $fileName);
665     if ($tmp && is_file($tmp)) {
666         $path = $tmp; // Yo! found it :)
667         break; // try-block
668     }
669 }
670
671 // 3) Then try to look for the example file
672 // below the subdir PHPDOCUMENTOR_BASE/examples/ ...
673 if (is_dir(PHPDOCUMENTOR_BASE . PATH_DELIMITER . 'examples')) {
674     $tmp = realpath(PHPDOCUMENTOR_BASE . PATH_DELIMITER . 'examples'
675     . PATH_DELIMITER . $original_path);
676     if ($tmp && is_file($tmp)) {
677         $path = $tmp; // Yo! found it :)
678         break; // try-block
679     }
680 }
681
682 $tmp = realpath(PHPDOCUMENTOR_BASE . PATH_DELIMITER . $original_path);
683 if ($tmp && is_file($tmp)) {
684     $path = $tmp; // Yo! found it :)
685     break; // try-block
686 }
687 // If we reach this point, nothing was found and $path is false.
688 } while (false);

689 if (!$path) {
690     addWarning(PDERROR_EXAMPLE_NOT_FOUND, $original_path);
691     $this->path = false;
692 } else {
693     $f = @fopen($path, 'r');
694     if ($f) {
695         $example = fread($f, filesize($path));
696         if (tokenizer_ext && ! $isTutorial) {
697             $obj = new phpDocumentorTWordParser;
698             $obj->setup($example);
699             $this->setSource($obj->getFileSource());
700             unset($obj);
701         } else {
702             $this->setSource($example);
703         }
704     }
705 }

```

```

706     }
707 }
708 /**
709 * sets the source
710 *
711 * @param string/array $source source code
712 * @param string/bool $class class name if this is a method,
713 *                             boolean in php 4.3.0,
714 *                             if this is a method this will be true
715 *
716 * @return void
717 */
718
719 function setSource($source, $class = false)
720 {
721     $this-> _class = $class;
722     $this-> source = $source;
723 }
724
725 /**
726 * converter helper for PHP 4.3.0+
727 *
728 * @param Converter &$c      output converter
729 *
730 * @return string
731 * @uses phpDocumentor_HighlightParser Parses the tokenized source
732 */
733 function arrayConvert(&    $c)
734 {
735     $source = $this-> source;
736     if ($this-> end != '*') {
737         $source = array_slice($this-> source, 0, $this-> end + $this-> start - 1);
738     }
739     $start = $this-> start - 1;
740     if ($start < 0) {
741         $start = 0;
742     }
743     return $c-> exampleProgramExample($source, true, true, $this-> _class, $start);
744 }
745
746 /**
747 * Return the source for the example file, enclosed in
748 * a <programlisting> tag to use in a tutorial
749 *
750 * @return string
751 */
752 function getProgramListing()
753 {
754     $source = explode("\n" , $this-> source);
755     $start = $this-> start;
756     if ($this-> end != '*') {
757         $source = array_slice($source, $start - 1, $this-> end - $start + 1);
758     } else {
759         $source = array_slice($source, $start - 1);
760     }
761     $source = join("\n" , $source);
762     return
763     "<programlisting role=\"php\">
764     <![CDATA[<\n"
765     $source .
766     "\n]>\n</programlisting>" ;
767 }
768
769 /**
770 * Represents the inheritdoc inline tag, used by classes/methods/vars to inherit
771 * documentation from the parent class if possible
772 *
773 * @category ToolsAndUtilities
774 * @package phpDocumentor
775 * @subpackage InlineTags
776 * @author Gregory Beaver <cellog@php.net>
777 * @copyright 2002-2008 Gregory Beaver
778 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
779 * @version Release: @VER@
780 * @filesource
781 * @link http://www.phpdoc.org
782 * @link http://pear.php.net/PhpDocumentor
783 * @see parserStringWithInlineTags
784 * @tutorial tags.inlineinheritdoc.pkg
785

```

```

786 * @todo      CS cleanup - change package to PhpDocumentor
787 * @todo      CS cleanup - change classname to PhpDocumentor_*
788 */
789 class parserInheritdocInlineTag extends parserInlineTag
790 {
791     /**
792      * always 'inheritdoc'
793      * @var string
794      */
795     var $inlinetype = 'inheritdoc';
796
797     /**
798      * Does nothing, overrides parent constructor
799      */
800     function parserInheritdocInlineTag()
801     {
802     }
803
804     /**
805      * only sets a warning and returns empty
806      *
807      * @return string
808      * @todo CS cleanup - rename to convert for camelCase rule
809      */
810     function Convert()
811     {
812         addWarning(PDERROR_INHERITDOC_DONT_WORK_HERE);
813         return '';
814     }
815 }
816
817 /**
818  * Represents the inline {@}id tag for tutorials
819  *
820  * @category ToolsAndUtilities
821  * @package phpDocumentor
822  * @subpackage InlineTags
823  * @author Gregory Beaver <cellog@php.net>
824  * @copyright 2002-2008 Gregory Beaver
825  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
826  * @version Release: @VER@
827  * @filesource
828  * @link http://www.phpdoc.org
829  * @link http://pear.php.net/PhpDocumentor
830  * @see parserStringWithInlineTags
831  * @tutorial tags.inlineid.pkg
832  * @todo CS cleanup - change package to PhpDocumentor
833  * @todo CS cleanup - change classname to PhpDocumentor_*
834  */
835 class parserIdInlineTag extends parserInlineTag
836 {
837     /**
838      * always 'id'
839      * @var string
840      */
841     var $inlinetype = 'id';
842
843     /**
844      * package of the {@}id
845      * @var string
846      */
847     var $package = 'default';
848
849     /**
850      * category of the {@}id
851      * @var string
852      */
853     var $category = 'default';
854
855     /**
856      * subpackage of the {@}id
857      * @var string
858      */
859     var $subpackage = '';
860
861     /**
862      * full name of the tutorial
863      * @var string
864      */
865     var $tutorial;
866
867     /**
868      * section/subsection name
869      * @var string
870      */

```

```

866     var $id;
867
868     /**
869      * constructor
870      *
871      * @param string $category    category name
872      * @param string $package     package name
873      * @param string $subpackage  subpackage name
874      * @param string $tutorial   tutorial name
875      * @param string $id         section/subsection name
876      */
877     function parserIdInlineTag($category,$package,$subpackage,$tutorial,$id = false)
878     {
879         $this-> package = $package;
880         $this-> subpackage = $subpackage;
881         $this-> tutorial = $tutorial;
882         $this-> id = $id;
883         $this-> category = $category;
884     }
885
886     /**
887      * converter
888      *
889      * @param Converter &$c      output converter
890      *
891      * @return string
892      * @uses Converter::getTutorialId() retrieve converter-specific ID
893      * @todo CS cleanup - rename to convert for camelCase rule
894      */
895     function Convert(& $c)
896     {
897         if (!$this-> id) {
898             return '';
899         }
900         return $c-> getTutorialId($this-> package, $this-> subpackage,
901             $this-> tutorial, $this-> id, $this-> category);
902     }
903 }
904
905 /**
906  * Represents {@toc} for table of contents generation in tutorials
907  *
908  * @category ToolsAndUtilities
909  * @package phpDocumentor
910  * @subpackage InlineTags
911  * @author Gregory Beaver <cellog@php.net>
912  * @copyright 2002-2008 Gregory Beaver
913  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
914  * @version Release: @VER@
915  * @filesource
916  * @link http://www.phpdoc.org
917  * @link http://pear.php.net/PhpDocumentor
918  * @see parserStringWithInlineTags
919  * @tutorial tags.inlinetoc.pkg
920  * @todo CS cleanup - change package to PhpDocumentor
921  * @todo CS cleanup - change classname to PhpDocumentor_*
922  */
923 class parserTocInlineTag extends parserInlineTag
924 {
925     /**
926      * always 'toc'
927      * @var string
928      */
929     var $inlinetype = 'toc';
930
931     /**
932      * @var array format:
933      * <pre>
934      * array(
935      *     array(
936      *         'tagname' => section,
937      *         'link'    => returnsee link,
938      *         'id'      => anchor name,
939      *         'title'   => from title tag
940      *     ),
941      *     ...
942      * )
943      * </pre>
944      * @access private
945      */
946     var $_toc = false;

```

```

946 /**
947 * full path to tutorial, used in conversion
948 * @var string
949 * @access private
950 */
951 var $_path = false;
952
953 /**
954 * constructor
955 */
956 function parserTocInlineTag()
957 {
958     parent::parserInlineTag('toc', '');
959 }
960
961 /**
962 * set the TOC
963 *
964 * @param array $toc format:
965 * <pre>
966 * array(
967 *     array(
968 *         'tag'    =>  {@link parserXMLDocBookTag},
969 *         'id'     =>  {@link parserIdInlineTag},
970 *         'title'  =>  {@link parserXMLDocBookTag title}
971 *     ),
972 *     ...
973 * )
974 </pre>
975 *
976 * @return void
977 */
978 function setTOC($toc)
979 {
980     $this-> toc = $toc;
981 }
982
983 /**
984 * set the path
985 *
986 * @param string $path the path
987 *
988 * @return void
989 */
990 function setPath($path)
991 {
992     $this-> _path = $path;
993 }
994
995 /**
996 * converter method
997 *
998 * <pre>
999 * array(
1000 *     'tagname' => string name of tag,
1001 *     'link'     => {@link tutorialLink} to the tutorial,
1002 *     'id'       => converter specific tutorial ID from
1003 *                   {@link Converter::getTutorialId()}
1004 *     'title'   => title of the tutorial)
1005 </pre>
1006 * and returns the results as the table of contents
1007 *
1008 * @param Converter &$c      converter object
1009 *
1010 * @return mixed
1011 * @uses Converter::getTutorialId() retrieve the tutorial ID for
1012 * @uses Converter::formatTutorialTOC() passes an array of format:
1013 * @todo CS cleanup - rename to convert for camelCase rule
1014 */
1015 function Convert(& $c)
1016 {
1017     $newtoc = array();
1018     if (isset($this-> toc) && is_array($this-> toc)) {
1019         foreach ($this-> toc as $i => $toc) {
1020             if (isset($toc['title'])) {
1021                 $toc['tag']-> setTitle($toc['title']);
1022             } else {
1023                 $toc['tag']-> setTitle(new parserStringWithInlineTags);
1024             }
1025             $newtoc[$i]['tagname'] = $toc['tag']-> name;

```

```

1026             $l = new tutorialLink;
1027             if (!isset($toc['title'])) {
1028                 $title = 'section ' . $toc['id']-> id;
1029             } else {
1030                 $title = $toc['title']-> Convert($c);
1031             }
1032             $l-> addLink($toc['id']-> id, $this-> _path, basename($this-> _path),
1033                         $toc['id']-> package, $toc['id']-> subpackage, strip_tags($title));
1034             $newtoc[$i]['link'] = $c-> returnSee($l);
1035             $newtoc[$i]['id'] = $c-> getTutorialId($toc['id']-> package,
1036                                         $toc['id']-> subpackage, basename($this-> _path),
1037                                         $toc['id']-> id, $toc['id']-> category);
1038             $newtoc[$i]['title'] = $title;
1039         }
1040     }
1041     return $c-> formatTutorialTOC($newtoc);
1042 }
1043 ?>
1044 ?>

```

# File Source for LinkClasses.inc

Documentation for this file is available at [LinkClasses.inc](#)

```
1  <?php
2  /**
3   * Linking to element documentation is performed by the classes in this file.
4   *
5   * abstractLink descendants contain enough information to differentiate every
6   * documentable element, and so can be converted to a link string by
7   * {@link Converter::returnSee()}
8   *
9   * phpDocumentor :: automatic documentation generator
10  *
11  * PHP versions 4 and 5
12  *
13  * Copyright (c) 2002-2008 Gregory Beaver
14  *
15  * LICENSE:
16  *
17  * This library is free software; you can redistribute it
18  * and/or modify it under the terms of the GNU Lesser General
19  * Public License as published by the Free Software Foundation;
20  * either version 2.1 of the License, or (at your option) any
21  * later version.
22  *
23  * This library is distributed in the hope that it will be useful,
24  * but WITHOUT ANY WARRANTY; without even the implied warranty of
25  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
26  * Lesser General Public License for more details.
27  *
28  * You should have received a copy of the GNU Lesser General Public
29  * License along with this library; if not, write to the Free Software
30  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
31  *
32  * @category ToolsAndUtilities
33  * @package phpDocumentor
34  * @subpackage Links
35  * @author Gregory Beaver <cellog@php.net>
36  * @copyright 2002-2008 Gregory Beaver
37  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
38  * @version CVS: $Id: LinkClasses.inc 253641 2008-02-24 02:35:44Z ashnazg $
39  * @filesource
40  * @link http://www.phpdoc.org
41  * @link http://pear.php.net/PhpDocumentor
42  * @since 1.2.0
43  * @todo CS cleanup - change package to PhpDocumentor
44  */
45
46 /**
47  * linking classes parent
48  *
49  * @category ToolsAndUtilities
50  * @package phpDocumentor
51  * @subpackage Links
52  * @author Gregory Beaver <cellog@php.net>
53  * @copyright 2002-2008 Gregory Beaver
54  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
55  * @version Release: @VER@
56  * @link http://www.phpdoc.org
57  * @link http://pear.php.net/PhpDocumentor
58  * @todo CS cleanup - change package to PhpDocumentor
59  * @todo CS cleanup - change classname to PhpDocumentor_*
60  */
61 class abstractLink
62 {
63     /**#@+
64      * @var string
65      */
66     var $path;
67     /***
```

```

68     * phpdoc alias _phpdoc_inc for phpdoc.inc
69     */
70     var $fileAlias = '';
71     /**
72      * element type linked to.
73      * can only be a documentable element
74     */
75     var $type = '';
76     var $name = '';
77     var $category = '';
78     var $package = '';
79     var $subpackage = '';
80     /**#@-*/
81
82     /**
83      * sets up the link
84     *
85      * @param string $path      full path to file containing element
86      * @param string $fileAlias page name, as configured by {@link Parser::parse}
87      * @param string $name      element name
88      * @param string $package   package element is in
89      * @param string $subpackage subpackage element is in
90      * @param string $category  optional category that documentation is in
91     *
92     * @return void
93     */
94     function addLink($path, $fileAlias, $name, $package, $subpackage,
95                      $category = false)
96     {
97         $this-> path = $path;
98         $this-> fileAlias = $fileAlias;
99         $this-> name = $name;
100        $this-> category = $category;
101        $this-> package = $package;
102        $this-> subpackage = $subpackage;
103    }
104
105
106    /**
107     * procedural page link
108     *
109     * @category ToolsAndUtilities
110     * @package phpDocumentor
111     * @subpackage Links
112     * @author Gregory Beaver <cellog@php.net>
113     * @copyright 2002-2008 Gregory Beaver
114     * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
115     * @version Release: @VER@
116     * @link http://www.phpdoc.org
117     * @link http://pear.php.net/PhpDocumentor
118     * @todo CS cleanup - change package to PhpDocumentor
119     * @todo CS cleanup - change classname to PhpDocumentor_*
120     */
121     class pageLink extends abstractLink
122     {
123         /**
124          * @var string
125         */
126         var $type = 'page';
127     }
128
129
130     /**
131      * function link
132      *
133      * @category ToolsAndUtilities
134      * @package phpDocumentor
135      * @subpackage Links
136      * @author Gregory Beaver <cellog@php.net>
137      * @copyright 2002-2008 Gregory Beaver
138      * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
139      * @version Release: @VER@
140      * @link http://www.phpdoc.org
141      * @link http://pear.php.net/PhpDocumentor
142      * @todo CS cleanup - change package to PhpDocumentor
143      * @todo CS cleanup - change classname to PhpDocumentor_*
144     */
145     class functionLink extends abstractLink
146     {
147         /**
148          * @var string

```

```

148      */
149      var $type = 'function';
150 }
151 /**
152 * define link
153 *
154 * @category ToolsAndUtilities
155 * @package phpDocumentor
156 * @subpackage Links
157 * @author Gregory Beaver <cellog@php.net>
158 * @copyright 2002-2008 Gregory Beaver
159 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
160 * @version Release: @VER@
161 * @link http://www.phpdoc.org
162 * @link http://pear.php.net/PhpDocumentor
163 * @todo CS cleanup - change package to PhpDocumentor
164 * @todo CS cleanup - change classname to PhpDocumentor_*
165 */
166 class defineLink extends abstractLink
167 {
168     /**
169      * @var string
170     */
171     var $type = 'define';
172 }
173 /**
174 * global variable link
175 *
176 * @category ToolsAndUtilities
177 * @package phpDocumentor
178 * @subpackage Links
179 * @author Gregory Beaver <cellog@php.net>
180 * @copyright 2002-2008 Gregory Beaver
181 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
182 * @version Release: @VER@
183 * @link http://www.phpdoc.org
184 * @link http://pear.php.net/PhpDocumentor
185 * @todo CS cleanup - change package to PhpDocumentor
186 * @todo CS cleanup - change classname to PhpDocumentor_*
187 */
188 class globalLink extends abstractLink
189 {
190     /**
191      * @var string
192     */
193     var $type = 'global';
194 }
195 /**
196 * class link
197 *
198 * @category ToolsAndUtilities
199 * @package phpDocumentor
200 * @subpackage Links
201 * @author Gregory Beaver <cellog@php.net>
202 * @copyright 2002-2008 Gregory Beaver
203 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
204 * @version Release: @VER@
205 * @link http://www.phpdoc.org
206 * @link http://pear.php.net/PhpDocumentor
207 * @todo CS cleanup - change package to PhpDocumentor
208 * @todo CS cleanup - change classname to PhpDocumentor_*
209 */
210 class classLink extends abstractLink
211 {
212     /**
213      * @var string
214     */
215     var $type = 'class';
216 }
217 /**
218 * method link
219 *
220 * @category ToolsAndUtilities
221 * @package phpDocumentor
222 * @subpackage Links
223 * @author Gregory Beaver <cellog@php.net>
224

```

```

228 * @copyright 2002-2008 Gregory Beaver
229 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
230 * @version Release: @VER@
231 * @link http://www.phpdoc.org
232 * @link http://pear.php.net/PhpDocumentor
233 * @todo CS cleanup - change package to PhpDocumentor
234 * @todo CS cleanup - change classname to PhpDocumentor_*
235 */
236 class methodLink extends abstractLink
237 {
238     /**
239      * @var string
240     */
241     var $type = 'method';
242     /**
243      * @var string
244     */
245     var $class = '';
246
247     /**
248      * sets up the link
249      *
250      * @param string $class      class name
251      * @param string $path       full path to file containing element
252      * @param string $fileAlias page name, as configured by {@link Parser::parse}
253      * @param string $name       element name
254      * @param string $package    package element is in
255      * @param string $subpackage subpackage element is in
256      * @param string $category   optional category that documentation is in
257      *
258      * @return void
259     */
260     function addLink($class, $path, $fileAlias, $name, $package, $subpackage,
261                     $category = false)
262     {
263         $this-> class = $class;
264         abstractLink::addLink($path, $fileAlias, $name, $package, $subpackage,
265                               $category);
266     }
267 }
268 /**
269  * class variable link
270  *
271  * @category ToolsAndUtilities
272  * @package phpDocumentor
273  * @subpackage Links
274  * @author Gregory Beaver <cellog@php.net>
275  * @copyright 2002-2008 Gregory Beaver
276  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
277  * @version Release: @VER@
278  * @link http://www.phpdoc.org
279  * @link http://pear.php.net/PhpDocumentor
280  * @todo CS cleanup - change package to PhpDocumentor
281  * @todo CS cleanup - change classname to PhpDocumentor_*
282  */
283 class varLink extends methodLink
284 {
285     /**
286      * @var string
287     */
288     var $type = 'var';
289 }
290 /**
291  * class constant link
292  *
293  * @category ToolsAndUtilities
294  * @package phpDocumentor
295  * @subpackage Links
296  * @author Gregory Beaver <cellog@php.net>
297  * @copyright 2002-2008 Gregory Beaver
298  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
299  * @version Release: @VER@
300  * @link http://www.phpdoc.org
301  * @link http://pear.php.net/PhpDocumentor
302  * @todo CS cleanup - change package to PhpDocumentor
303  * @todo CS cleanup - change classname to PhpDocumentor_*
304  */
305 class constLink extends methodLink

```

```

308  {
309      /**
310      * @var string
311      */
312      var $type = 'const';
313  }
314
315 /**
316 * tutorial link
317 *
318 * @category ToolsAndUtilities
319 * @package phpDocumentor
320 * @subpackage Links
321 * @author Gregory Beaver <cellog@php.net>
322 * @copyright 2002-2008 Gregory Beaver
323 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
324 * @version Release: @VER@
325 * @link http://www.phpdoc.org
326 * @link http://pear.php.net/PhpDocumentor
327 * @todo CS cleanup - change package to PhpDocumentor
328 * @todo CS cleanup - change classname to PhpDocumentor_*
329 */
330 class tutorialLink extends abstractLink
331 {
332     /**#@+
333     * @var string
334     */
335     var $type = 'tutorial';
336     var $section = '';
337     var $title = false;
338     /**#@-*/
339
340     /**
341     * sets up the link
342     *
343     * @param string $section    section/subsection name
344     * @param string $path       full path to file containing element
345     * @param string $name       page name, as configured by {@link Parser::parse}
346     * @param string $package    package element is in
347     * @param string $subpackage subpackage element is in
348     * @param string $title      title of tutorial
349     * @param string $category   optional category that documentation is in
350     *
351     * @return void
352     */
353     function addLink($section, $path, $name, $package, $subpackage, $title = false,
354                     $category = false)
355     {
356         $this-> section = $section;
357         $this-> title = $title;
358         parent::addLink($path, '', $name, $package, $subpackage, $category);
359     }
360 }
361 ?>
```

# File Source for Beautifier.php

Documentation for this file is available at [Beautifier.php](#)

```
1  <?PHP
2  /**
3  * XML/Beautifier.php
4  *
5  * Format XML files containing unknown entities (like all of peardoc)
6  *
7  * phpDocumentor :: automatic documentation generator
8  *
9  * PHP versions 4 and 5
10 *
11 * Copyright (c) 2004-2006 Gregory Beaver
12 *
13 * LICENSE:
14 *
15 * This library is free software; you can redistribute it
16 * and/or modify it under the terms of the GNU Lesser General
17 * Public License as published by the Free Software Foundation;
18 * either version 2.1 of the License, or (at your option) any
19 * later version.
20 *
21 * This library is distributed in the hope that it will be useful,
22 * but WITHOUT ANY WARRANTY; without even the implied warranty of
23 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
24 * Lesser General Public License for more details.
25 *
26 * You should have received a copy of the GNU Lesser General Public
27 * License along with this library; if not, write to the Free Software
28 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
29 *
30 * @package    phpDocumentor
31 * @subpackage Parsers
32 * @author     Greg Beaver <cellog@php.net>
33 * @copyright  2004-2006 Gregory Beaver
34 * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
35 * @version    CVS: $Id: Beautifier.php 212211 2006-04-30 22:18:14Z cellog $
36 * @filesource
37 * @link       http://www.phpdoc.org
38 * @link       http://pear.php.net/PhpDocumentor
39 * @since      1.3.0
40 */
41
42 /**
43 * This is just like XML_Beautifier, but uses {@link phpDocumentor_XML_Beautifier_Tokenizer}
44 * @package phpDocumentor
45 * @subpackage Parsers
46 * @since 1.3.0
47 */
48 class phpDocumentor_peardoc2_XML_Beautifier extends XML_Beautifier {
49
50 /**
51 * format a file or URL
52 *
53 * @access public
54 * @param string $file      filename
55 * @param mixed  $newFile   filename for beautified XML file (if none is given, the XML
56 * string will be returned.)
57 *                                         if you want overwrite the original file, use
58 * XML_BEAUTIFIER_OVERWRITE
59 * @param string $renderer  Renderer to use, default is the plain xml renderer
60 * @return mixed           XML string of no file should be written, true if file could
61 * be written
62 * @throws PEAR_Error
63 * @uses _loadRenderer() to load the desired renderer
64 */
65 function formatFile($file, $newFile = null, $renderer = "Plain" ) {
66 }
```

```

65     if ($this-> apiVersion() != '1.0') {
66         return $this-> raiseError('API version must be 1.0');
67     }
68     /**
69      * Split the document into tokens
70      * using the XML_Tokenizer
71      */
72     require_once dirname(__FILE__) . '/Tokenizer.php';
73     $tokenizer = new phpDocumentor_XML_Beautifier_Tokenizer();
74
75     $tokens = $tokenizer-> tokenize( $file, true );
76
77     if (PEAR::isError($tokens)) {
78         return $tokens;
79     }
80
81     include_once dirname(__FILE__) . '/Plain.php';
82     $renderer = new PHPDoc_XML_Beautifier_Renderer_Plain($this-> _options);
83
84     $xml = $renderer-> serialize($tokens);
85
86     if ($newFile == null) {
87         return $xml;
88     }
89
90     $fp = @fopen($newFile, "w");
91     if (!$fp) {
92         return PEAR::raiseError("Could not write to output file",
93             XML_BEAUTIFIER_ERROR_NO_OUTPUT_FILE);
94     }
95
96     flock($fp, LOCK_EX);
97     fwrite($fp, $xml);
98     flock($fp, LOCK_UN);
99     fclose($fp);
100    return true;
101}
102 /**
103  * format an XML string
104  *
105  * @access public
106  * @param string $string XML
107  * @return string formatted XML string
108  * @throws PEAR_Error
109  */
110 function formatString($string, $renderer = "Plain")
111 {
112     if ($this-> apiVersion() != '1.0') {
113         return $this-> raiseError('API version must be 1.0');
114     }
115     /**
116      * Split the document into tokens
117      * using the XML_Tokenizer
118      */
119     require_once dirname(__FILE__) . '/Tokenizer.php';
120     $tokenizer = new phpDocumentor_XML_Beautifier_Tokenizer();
121
122     $tokens = $tokenizer-> tokenize( $string, false );
123
124     if (PEAR::isError($tokens)) {
125         return $tokens;
126     }
127
128     include_once dirname(__FILE__) . '/Plain.php';
129     $renderer = new PHPDoc_XML_Beautifier_Renderer_Plain($this-> _options);
130
131     $xml = $renderer-> serialize($tokens);
132
133     return $xml;
134 }
135 ?>

```

# File Source for Tokenizer.php

Documentation for this file is available at [Tokenizer.php](#)

```
1  <?php
2  /**
3  * XML/Beautifier.php
4  *
5  * Format XML files containing unknown entities (like all of peardoc)
6  *
7  * phpDocumentor :: automatic documentation generator
8  *
9  * PHP versions 4 and 5
10 *
11 * Copyright (c) 2004-2006 Gregory Beaver
12 *
13 * LICENSE:
14 *
15 * This library is free software; you can redistribute it
16 * and/or modify it under the terms of the GNU Lesser General
17 * Public License as published by the Free Software Foundation;
18 * either version 2.1 of the License, or (at your option) any
19 * later version.
20 *
21 * This library is distributed in the hope that it will be useful,
22 * but WITHOUT ANY WARRANTY; without even the implied warranty of
23 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
24 * Lesser General Public License for more details.
25 *
26 * You should have received a copy of the GNU Lesser General Public
27 * License along with this library; if not, write to the Free Software
28 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
29 *
30 * @package    phpDocumentor
31 * @subpackage Parsers
32 * @author     Greg Beaver <cellog@php.net>
33 * @copyright  2004-2006 Gregory Beaver
34 * @license    http://www.opensource.org/licenses/lgpl-license.php LGPL
35 * @version    CVS: $Id: Tokenizer.php 238276 2007-06-22 14:58:30Z ashnazg $
36 * @filesource
37 * @link       http://www.phpdoc.org
38 * @link       http://pear.php.net/PhpDocumentor
39 * @since      1.3.0
40 */
41 /**
42 * From the XML_Beautifier package
43 */
44 require_once 'XML/Beautifier/Tokenizer.php';
45 /**
46 * Highlights source code using {@link parse()}
47 * @package phpDocumentor
48 * @subpackage Parsers
49 */
50 class phpDocumentor_XML_Beautifier_Tokenizer extends XML_Beautifier_Tokenizer
{
51     /**
52     * @access private
53     */
54     var $_curthing;
55     var $_tag;
56     var $_attrs;
57     var $_attr;
58
59     /**
60     * @var array
61     */
62     var $eventHandlers = array(
63         PHPDOC_XMLTOKEN_EVENT_NOEVENTS => 'normalHandler',
64         PHPDOC_XMLTOKEN_EVENT_XML => 'parseXMLHandler',
65         PHPDOC_XMLTOKEN_EVENT_PI => 'parsePiHandler',
66
67     );
68 }
```

```

68             PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE => 'attrHandler',
69             PHPDOC_XMLTOKEN_EVENT_OPENTAG => 'tagHandler',
70             PHPDOC_XMLTOKEN_EVENT_IN_CDATA => 'realcdataHandler',
71             PHPDOC_XMLTOKEN_EVENT_DEF => 'defHandler',
72             PHPDOC_XMLTOKEN_EVENT_CLOSETAG => 'closetagHandler',
73             PHPDOC_XMLTOKEN_EVENT_ENTITY => 'entityHandler',
74             PHPDOC_XMLTOKEN_EVENT_COMMENT => 'commentHandler',
75             PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE => 'stringHandler',
76             PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE => 'stringHandler',
77             PHPDOC_XMLTOKEN_EVENT_CDATA => 'parseCdataHandler',
78         );
79
80     /**
81      * Parse a new file
82      *
83      * The parse() method is a do...while() loop that retrieves tokens one by
84      * one from the {@link $_event_stack}, and uses the token event array set up
85      * by the class constructor to call event handlers.
86      *
87      * The event handlers each process the tokens passed to them, and use the
88      * {@link _addoutput()} method to append the processed tokens to the
89      * {@link $_line} variable. The word parser calls {@link newLineNum()}
90      * every time a line is reached.
91      *
92      * In addition, the event handlers use special linking functions
93      * {@link _link()} and its cousins (_classlink(), etc.) to create in-code
94      * hyperlinks to the documentation for source code elements that are in the
95      * source code.
96      *
97      * @uses setupStates() initialize parser state variables
98      * @uses configWordParser() pass $parse_data to prepare retrieval of tokens
99      * @param string
100     * @param Converter
101     * @param false/string full path to file with @filesource tag, if this
102     * is a @filesource parse
103     * @param false/integer starting line number from {@source linenum}
104     * @staticvar integer used for recursion limiting if a handler for
105     * an event is not found
106     * @return bool
107     */
108    function parseString ($parse_data)
109    {
110        static $endrecur = 0;
111        $parse_data = str_replace(array("\r\n", "\n"), "\t", $parse_data);
112        $this-> setupStates($parse_data);
113
114        $this-> configWordParser(PHPDOC_XMLTOKEN_EVENT_NOEVENTS);
115        // initialize variables so E_ALL error_reporting doesn't complain
116        $pevent = 0;
117        $word = 0;
118        $this-> _curthing = '';
119
120        do
121        {
122            $lpevent = $pevent;
123            $pevent = $this-> _event_stack-> getEvent();
124            if ($lpevent != $pevent)
125            {
126                $this-> _last_pevent = $lpevent;
127                $this-> configWordParser($pevent);
128            }
129            $this-> _wp-> setWhitespace(true);
130
131            $dbg_linenum = $this-> _wp-> linenum;
132            $dbg_pos = $this-> _wp-> getPos();
133            $this-> _pv_last_word = $word;
134            $this-> _pv_curline = $this-> _wp-> linenum;
135            $word = $this-> _wp-> getWord();
136
137            if (PHPDOCUMENTOR_DEBUG == true)
138            {
139                echo "LAST: " . $this-> _pv_last_word;
140                echo "| " . $this-> _wp-> getPos();
141                echo "| \n" . $this-> _wp-> getLine();
142                echo "PEVENT: " . $this-> getParserEventName($pevent) . "\n";
143                echo "LASTPEVENT: " . $this-> getParserEventName($this-> _last_pevent) . "\n";
144                echo "LINE: " . $this-> _line . "\n";
145            }
146        }
147    }

```

```

145 //           echo "OUTPUT: ".$this->_output."\n";
146 echo $dbg_lineno.'. '$dbg_pos . ":" ;
147 echo '|'.htmlspecialchars($word);
148 echo "|\\n" ;
149 echo "-----\n\n" ;
150 flush();
151 }
152 if (isset($this-> eventHandlers[$pevent]))
153 {
154     $handle = $this-> eventHandlers[$pevent];
155     $this-> $handle($word, $pevent);
156 } else
157 {
158     echo ('WARNING: possible error, no handler for event number '.$pevent);
159     if ($endrecur++ == 25)
160     {
161         return $this-> raiseError("FATAL ERROR, recursion limit
reached");
162     }
163 }
164 } while (!$word === false));
165 return true;
166 }
167 /**
168 * Event Handlers
169 *
170 * All Event Handlers use {@link checkEventPush()} and
171 * {@link checkEventPop()} to set up the event stack and parser state.
172 * @access private
173 * @param string/array token value
174 * @param integer parser event from {@link Parser.inc}
175 */
176 /**
177 * Most tokens only need highlighting, and this method handles them
178 */
179 function normalHandler($word, $pevent)
180 {
181     if ($this-> checkEventPush($word, $pevent)) {
182         $this-> _wp-> backupPos($word);
183         $this-> _addoutput($pevent);
184         $this-> _curthing = '';
185         return;
186     }
187     $this-> _curthing .= $word;
188
189     if ($this-> checkEventPop($word, $pevent)) {
190         $this-> _addoutput($pevent);
191         $this-> _curthing = '';
192     }
193 }
194 /**
195 * handle <!-- comments -->
196 */
197 function commentHandler($word, $pevent)
198 {
199     if ($this-> checkEventPush($word, $pevent)) {
200         $this-> _wp-> backupPos($word);
201         return;
202     }
203
204     $this-> _curthing .= $word;
205     if ($this-> checkEventPop($word, $pevent)) {
206         $this-> _addoutput($pevent);
207         $this-> _curthing = '';
208     }
209 }
210 }
211 /**
212 * handle <?Processor instructions?>
213 */
214 function parsePiHandler($word, $pevent)
215 {
216     if ($this-> checkEventPush($word, $pevent)) {
217         $this-> _wp-> backupPos($word);
218         return;
219     }
220     if ($this-> checkEventPop($word, $pevent)) {
221         $this-> _addoutput($pevent);
222     }

```

```

224     $this-> _curthing = '';
225     $this-> _attrs = null;
226     return;
227 }
228 if (!strlen($this-> _curthing)) {
229     $this-> _curthing .= str_replace('<?' , '' , $word);
230 } else {
231     if (!isset($this-> _attrs) || !is_string($this-> _attrs)) {
232         $this-> _attrs = '';
233     }
234     $this-> _attrs .= $word;
235 }
236 }
237 /**
238 * handle <?xml Processor instructions?
239 */
240 function parseXMLHandler($word, $pevent)
241 {
242     if ($this-> checkEventPush($word, $pevent)) {
243         $this-> _wp-> backupPos($word);
244         return;
245     }
246
247     $this-> _curthing .= $word;
248     if ($this-> checkEventPop($word, $pevent)) {
249         $this-> _addoutput($pevent);
250         $this-> _curthing = '';
251     }
252 }
253 }
254 /**
255 * handle <![CDATA[ unescaped text ]]>
256 */
257 function realcdataHandler($word, $pevent)
258 {
259     $this-> _curthing .= $word;
260     if ($this-> checkEventPop($word, $pevent)) {
261         $this-> _addoutput($pevent);
262         $this-> _curthing = '';
263     }
264 }
265 }
266 /**
267 * handle <tags>
268 */
269 function tagHandler($word, $pevent)
270 {
271     if ($this-> checkEventPush($word, $pevent)) {
272         $this-> _wp-> backupPos($word);
273         $this-> _curthing = '';
274         return;
275     }
276
277     if ($word{0} == '<' ) {
278         $this-> _tag = substr($word, 1);
279     }
280
281     if ($this-> checkEventPop($word, $pevent)) {
282         $this-> _addoutput($pevent);
283         $this-> _tag = null;
284         $this-> _attrs = null;
285         if ($word == '>' ) {
286             $this-> _event_stack-> pushEvent(PHPDOC_XMLTOKEN_EVENT_CDATA);
287             return;
288         }
289     }
290 }
291 }
292 /**
293 * handle </tags>
294 */
295 function closetagHandler($word, $pevent)
296 {
297     if ($this-> checkEventPush($word, $pevent)) {
298         $this-> _wp-> backupPos($word);
299         return;
300     }
301     if ($this-> checkEventPop($word, $pevent)) {
302         $this-> _addoutput($pevent);
303     }

```

```

304         $this-> _tag = '';
305     }
306     $this-> _tag = trim(str_replace('</', '', $word));
307 }
308 /**
309 * handle <!def>
310 */
311 function defHandler($word, $pevent)
312 {
313     if ($this-> checkEventPush($word, $pevent)) {
314         $this-> _wp-> backupPos($word);
315         return;
316     }
317
318     $this-> _curthing .= $word;
319     if ($this-> checkEventPop($word, $pevent)) {
320         $this-> _addoutput($pevent);
321         $this-> _curthing = '';
322     }
323 }
324 }
325 /**
326 * Most tokens only need highlighting, and this method handles them
327 */
328 function attrHandler($word, $pevent)
329 {
330     if ($e = $this-> checkEventPush($word, $pevent)) {
331         return;
332     }
333     if (!isset($this-> _attrs) || !is_array($this-> _attrs)) {
334         $this-> _attrs = array();
335     }
336     if (strpos($word, '=') != '') {
337         $this-> _attrs[$this-> _attr] = trim(str_replace('=', '', $word));
338     }
339     if ($this-> checkEventPop($word, $pevent)) {
340         $this-> _wp-> backupPos($word);
341         return;
342     }
343 }
344 }
345 /**
346 * handle attribute values
347 */
348 function stringHandler($word, $pevent)
349 {
350     if ($this-> checkEventPop($word, $pevent)) {
351         return;
352     }
353     $this-> _attrs[$this-> _attr] = $word;
354 }
355 /**
356 * handle &entities;
357 */
358 function entityHandler($word, $pevent)
359 {
360     if ($this-> checkEventPop($word, $pevent)) {
361         $this-> _addoutput($pevent);
362         $this-> _curthing = '';
363         return;
364     }
365     if (strlen($word) && $word[0] == '&') {
366         $word = substr($word, 1);
367     }
368     $this-> _curthing .= $word;
369 }
370 /**
371 * handle tag contents
372 */
373 function parseCdataHandler($word, $pevent)
374 {
375     if ($this-> checkEventPush($word, $pevent)) {
376         $this-> _wp-> backupPos($word);
377         if (strlen($this-> _curthing)) {
378             $this-> _addoutput($pevent);
379         }
380     }
381 }
382 
```

```

384         $this-> _curthing = '';
385         return;
386     }
387     if ($this-> checkEventPop($word, $event)) {
388         if (strlen($this-> _curthing)) {
389             $this-> _addoutput($event);
390         }
391         $this-> _curthing = '';
392         $this-> _event_stack-> pushEvent(PHPDOC_XMLTOKEN_EVENT_CLOSETAG);
393         return;
394     }
395     $this-> _curthing .= $word;
396 }
397 /**
398 */
400 /**
401 * Handler for real character data
402 *
403 * @access protected
404 * @param object XML parser object
405 * @param string CDATA
406 * @return void
407 */
408 function incdataHandler($parser, $cdata)
409 {
410     if ((string)$cdata === '') {
411         return true;
412     }
413     $struct = array(
414         "type"          => PHPDOC_BEAUTIFIER_CDATA,
415         "data"          => $cdata,
416         "depth"         => $this-> _depth
417     );
418
419     $this-> _appendToParent($struct);
420 }
421 /**
422 * Output Methods
423 * @access private
424 */
425 /**
426 * This method adds output to {@link $_line}
427 *
428 * If a string with variables like "$test this" is present, then special
429 * handling is used to allow processing of the variable in context.
430 * @see _flush_save()
431 */
432 function _addoutput($event)
433 {
434     $type =
435     array(
436         PHPDOC_XMLTOKEN_EVENT_NOEVENTS => '_handleXMLDefault',
437         PHPDOC_XMLTOKEN_EVENT_CLOSETAG => 'endHandler',
438         PHPDOC_XMLTOKEN_EVENT_ENTITY => 'entityrefHandler',
439         PHPDOC_XMLTOKEN_EVENT_DEF => '_handleXMLDefault',
440         PHPDOC_XMLTOKEN_EVENT_PI => 'parsePiHandler',
441         PHPDOC_XMLTOKEN_EVENT_XML => '_handleXMLDefault',
442         PHPDOC_XMLTOKEN_EVENT_OPENTAG => 'startHandler',
443         PHPDOC_XMLTOKEN_EVENT_COMMENT => '_handleXMLDefault',
444         PHPDOC_XMLTOKEN_EVENT_CDATA => 'cdataHandler',
445         PHPDOC_XMLTOKEN_EVENT_IN_CDATA => 'incdataHandler',
446     );
447     $method = $type[$event];
448     switch ($event) {
449         case PHPDOC_XMLTOKEN_EVENT_COMMENT :
450             echo "comment: $this->_curthing\n";
451             $this-> $method(false, $this-> _curthing);
452             break;
453         case PHPDOC_XMLTOKEN_EVENT_OPENTAG :
454             echo "open tag: $this->_tag\n";
455             var_dump($this-> _attrs);
456             $this-> $method(false, $this-> _tag, $this-> _attrs);
457             break;
458         case PHPDOC_XMLTOKEN_EVENT_CLOSETAG :
459             echo "close tag: $this->_tag\n";
460             $this-> $method(false, $this-> _curthing);
461             break;
462         case PHPDOC_XMLTOKEN_EVENT_NOEVENTS :

```

```

464             if (!strlen($this-> _curthing)) {
465                 return;
466             }
467         //     echo "default: $this->_curthing\n";
468         $this-> $method(false, $this-> _curthing);
469         break;
470     //     case PHPDOC_XMLTOKEN_EVENT_DEF :
471         //         echo "<!definition: $this->_curthing\n";
472         $this-> $method(false, $this-> _curthing);
473         break;
474     case PHPDOC_XMLTOKEN_EVENT_PI :
475         //         echo "<?pi: $this->_curthing\n";
476         //         echo "<?pi attrs: $this->_attrs\n";
477         $this-> $method(false, $this-> _curthing, $this-> _attrs);
478         break;
479     case PHPDOC_XMLTOKEN_EVENT_XML :
480         //         echo "<?xml: $this->_curthing\n";
481         $this-> $method(false, $this-> _curthing, $this-> _attrs);
482         break;
483     case PHPDOC_XMLTOKEN_EVENT_CDATA :
484     case PHPDOC_XMLTOKEN_EVENT_IN_CDATA :
485         //         echo "cdata: $this->_curthing\n";
486         $this-> $method(false, $this-> _curthing);
487         break;
488     case PHPDOC_XMLTOKEN_EVENT_ENTITY :
489         //         echo "entity: $this->_curthing\n";
490         $this-> $method(false, $this-> _curthing, false, false, false);
491         break;
492     }
493 }
494 /**
495 */
496 /**
497 * tell the parser's WordParser {@link $wp} to set up tokens to parse words by.
498 * tokens are word separators. In English, a space or punctuation are examples of tokens.
499 * In PHP, a token can be a ;, a parenthesis, or even the word "function"
500 * @param $value integer an event number
501 * @see WordParser
502 */
503
504 function configWordParser($e)
505 {
506     $this-> _wp-> setSeparator($this-> tokens[($e + 100)]);
507 }
508 /**
509 * this function checks whether parameter $word is a token for pushing a new event onto the
Event Stack.
510 * @return mixed returns false, or the event number
511 */
512
513 function checkEventPush($word,$pevent)
514 {
515     $e = false;
516     if (isset($this-> pushEvent[$pevent]))
517     {
518         if (isset($this-> pushEvent[$pevent][strtolower($word)]))
519             $e = $this-> pushEvent[$pevent][strtolower($word)];
520     }
521     if ($e)
522     {
523         $this-> _event_stack-> pushEvent($e);
524         return $e;
525     } else {
526         return false;
527     }
528 }
529
530 /**
531 * this function checks whether parameter $word is a token for popping the current event
off of the Event Stack.
532 * @return mixed returns false, or the event number popped off of the stack
533 */
534
535 function checkEventPop($word,$pevent)
536 {
537     if (!isset($this-> popEvent[$pevent])) return false;
538     if (in_array(strtolower($word), $this-> popEvent[$pevent]))
539     {
540         return $this-> _event_stack-> popEvent();
541     } else {

```

```

542         return false;
543     }
544 }
545 /**
546 * Initialize all parser state variables
547 * @param boolean true if we are highlighting an inline {@}source} tag's
548 * output
549 * @param false/stringname of class we are going to start from
550 * @uses $_wp sets to a new {@link phpDocumentor_HighlightWordParser}
551 */
552 function setupStates($parsedata)
553 {
554     $this-> _output = '';
555     $this-> _line = '';
556     unset($this-> _wp);
557     $this-> _wp = new WordParser;
558     $this-> _wp-> setup($parsedata);
559     $this-> _event_stack = @new EventStack;
560     $this-> _event_stack-> popEvent();
561     $this-> _event_stack-> pushEvent(PHPDOC_XMLTOKEN_EVENT_NOEVENTS);
562     $this-> _pv_linenumber = null;
563     $this-> _pv_next_word = false;
564 }
565 /**
566 * Initialize the {@link $tokenpushEvent, $wordpushEvent} arrays
567 */
568 function phpDocumentor_XML_Beautifier_Tokenizer()
569 {
570     $this-> tokens[STATE_XMLTOKEN_CDATA] =
571     $this-> tokens[STATE_XMLTOKEN_NOEVENTS] = array('<?xml' , '<!--' ,
572 '!-->', '<!' , '</' , '<?' , '<' , '>' , '&');
573     $this-> tokens[STATE_XMLTOKEN_OPENTAG] =
574     "\t" , " " , ">" , "/>";
575     $this-> tokens[STATE_XMLTOKEN_XML] =
576     $this-> tokens[STATE_XMLTOKEN_PI] = array("\n" , "\t" ,
577 " " , "?>");
578     $this-> tokens[STATE_XMLTOKEN_IN_CDATA] = array(']>' );
579     $this-> tokens[STATE_XMLTOKEN_CLOSETAG] = array("\n" ,
580 " " , ">" );
581     $this-> tokens[STATE_XMLTOKEN_COMMENT] = array("\n" ,
582 " " , '-->' );
583     $this-> tokens[STATE_XMLTOKEN_DEF] = array("\n" ,
584 " " , "[>' , ">' );
585     $this-> tokens[STATE_XMLTOKEN_ENTITY] = array("\n" ,
586 " " , ', ' );
587     $this-> tokens[STATE_XMLTOKEN_ATTRIBUTE] =
588     "\n" , " " , ">" , "/>";
589     $this-> tokens[STATE_XMLTOKEN_DOUBLEQUOTE] = array("\n" ,
590 " " , " " );
591     $this-> tokens[STATE_XMLTOKEN_SINGLEQUOTE] =
592     "\n" , " " );
593 /**
594 * Push events
595 */
596 $this-> pushEvent[PHPDOC_XMLTOKEN_EVENT_NOEVENTS] =
597 array(
598     '<' => PHPDOC_XMLTOKEN_EVENT_OPENTAG,
599     '<?' => PHPDOC_XMLTOKEN_EVENT_PI,
600     '<?xml' => PHPDOC_XMLTOKEN_EVENT_XML,
601     '</' => PHPDOC_XMLTOKEN_EVENT_CLOSETAG,
602     '&' => PHPDOC_XMLTOKEN_EVENT_ENTITY,
603     '<![' cdata[' => PHPDOC_XMLTOKEN_EVENT_IN_CDATA,
604     '<!--' => PHPDOC_XMLTOKEN_EVENT_COMMENT,
605     '<!' => PHPDOC_XMLTOKEN_EVENT_DEF,
606     );
607 /**
608 * Push events for attributes
609 */
610 $this-> pushEvent[PHPDOC_XMLTOKEN_EVENT_OPENTAG] =
611 array(
612     " " => PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE,
613     "\n" => PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE,
614     );
615 /**
616 * Pop event
617 */
618 $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_IN_CDATA] = array(']>' );
619 /**

```

```

617     $this-> pushEvent[PHPDOC_XMLTOKEN_EVENT_CDATA] =
618         array(
619             '<' => PHPDOC_XMLTOKEN_EVENT_OPENTAG,
620             '<?' => PHPDOC_XMLTOKEN_EVENT_PI,
621             '&' => PHPDOC_XMLTOKEN_EVENT_ENTITY,
622             '<!--' => PHPDOC_XMLTOKEN_EVENT_COMMENT,
623             '<!' => PHPDOC_XMLTOKEN_EVENT_DEF,
624             '<![CDATA[' => PHPDOC_XMLTOKEN_EVENT_IN_CDATA,
625         );
626     /*****
627
628     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_XML] =
629     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_PI] = array('?' );
630     *****/
631
632     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_ENTITY] = array(';');
633     *****/
634
635     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE] = array("'");
636     *****/
637
638     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE] = array("''");
639     *****/
640
641     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_OPENTAG] = array('>', '/>');
642     *****/
643
644     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_CLOSETAG] = array('>');
645     *****/
646
647     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_COMMENT] = array('--');
648     *****/
649
650     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_DEF] = array('>', ']>');
651     *****/
652
653     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE] = array('>', '/>');
654     *****/
655
656     $this-> popEvent[PHPDOC_XMLTOKEN_EVENT_CDATA] =
657         array('</');
658     *****/
659 }
660
661 function getParserEventName ($value)
662 {
663     $lookup = array(
664         PHPDOC_XMLTOKEN_EVENT_NOEVENTS =>
665         "PHPDOC_XMLTOKEN_EVENT_NOEVENTS"
666         PHPDOC_XMLTOKEN_EVENT_PI => "PHPDOC_XMLTOKEN_EVENT_PI"
667         PHPDOC_XMLTOKEN_EVENT_OPENTAG =>
668         "PHPDOC_XMLTOKEN_EVENT_OPENTAG"
669         PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE =>
670         "PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE"
671         PHPDOC_XMLTOKEN_EVENT_CLOSETAG =>
672         "PHPDOC_XMLTOKEN_EVENT_CLOSETAG"
673         PHPDOC_XMLTOKEN_EVENT_ENTITY => "PHPDOC_XMLTOKEN_EVENT_ENTITY"
674         PHPDOC_XMLTOKEN_EVENT_COMMENT =>
675         "PHPDOC_XMLTOKEN_EVENT_COMMENT"
676         PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE =>
677         "PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE"
678         PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE =>
679         "PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE"
680         PHPDOC_XMLTOKEN_EVENT_CDATA => 'PHPDOC_XMLTOKEN_EVENT_CDATA',
681         PHPDOC_XMLTOKEN_EVENT_DEF => 'PHPDOC_XMLTOKEN_EVENT_DEF',
682         PHPDOC_XMLTOKEN_EVENT_XML => 'PHPDOC_XMLTOKEN_EVENT_XML',
683         PHPDOC_XMLTOKEN_EVENT_IN_CDATA => 'PHPDOC_XMLTOKEN_EVENT_IN_CDATA',
684     );
685     if (isset($lookup[$value]))
686         return $lookup[$value];
687     else return $value;
688 }
689
690 /**
691  * starting state */
692 define("PHPDOC_XMLTOKEN_EVENT_NOEVENTS" , 1);
693 /**
694  * currently in starting state */
695 define("STATE_XMLTOKEN_NOEVENTS" , 101);
696

```

```

690  /** used when a processor instruction is found */
691  define("PHPDOC_XMLTOKEN_EVENT_PI" , 2);
692  /** currently in processor instruction */
693  define("STATE_XMLTOKEN_PI" , 102);
694
695  /** used when an open <tag> is found */
696  define("PHPDOC_XMLTOKEN_EVENT_OPENTAG" , 3);
697  /** currently parsing an open <tag> */
698  define("STATE_XMLTOKEN_OPENTAG" , 103);
699
700  /** used when a <tag attr="attribute"> is found */
701  define("PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE" , 4);
702  /** currently parsing an open <tag> */
703  define("STATE_XMLTOKEN_ATTRIBUTE" , 104);
704
705  /** used when a close </tag> is found */
706  define("PHPDOC_XMLTOKEN_EVENT_CLOSETAG" , 5);
707  /** currently parsing a close </tag> */
708  define("STATE_XMLTOKEN_CLOSETAG" , 105);
709
710  /** used when an &entity; is found */
711  define("PHPDOC_XMLTOKEN_EVENT_ENTITY" , 6);
712  /** currently parsing an &entity; */
713  define("STATE_XMLTOKEN_ENTITY" , 106);
714
715  /** used when a <!-- comment --> is found */
716  define("PHPDOC_XMLTOKEN_EVENT_COMMENT" , 7);
717  /** currently parsing a <!-- comment --> */
718  define("STATE_XMLTOKEN_COMMENT" , 107);
719
720  /** used when a <!-- comment --> is found */
721  define("PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE" , 8);
722  /** currently parsing a <!-- comment --> */
723  define("STATE_XMLTOKEN_SINGLEQUOTE" , 108);
724
725  /** used when a <!-- comment --> is found */
726  define("PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE" , 9);
727  /** currently parsing a <!-- comment --> */
728  define("STATE_XMLTOKEN_DOUBLEQUOTE" , 109);
729
730  /** used when a <! is found */
731  define("PHPDOC_XMLTOKEN_EVENT_DEF" , 10);
732  /** currently parsing a <! */
733  define("STATE_XMLTOKEN_DEF" , 110);
734
735  /** used when a <! is found */
736  define("PHPDOC_XMLTOKEN_EVENT_CDATA" , 11);
737  /** currently parsing a <! */
738  define("STATE_XMLTOKEN_CDATA" , 111);
739
740  /** used when a <?xml is found */
741  define("PHPDOC_XMLTOKEN_EVENT_XML" , 12);
742  /** currently parsing a <?xml */
743  define("STATE_XMLTOKEN_XML" , 112);
744
745  /** used when a <![CDATA[ section is found */
746  define('PHPDOC_XMLTOKEN_EVENT_IN_CDATA' , 13);
747  /** currently parsing a <![CDATA[ ]]> */
748  define('STATE_XMLTOKEN_IN_CDATA' , 113);
749
750  /** do not remove, needed in plain renderer */
751  define('PHPDOC_BEAUTIFIER_CDATA' , 100000);
752 ?>

```

# File Source for HighlightParser.inc

Documentation for this file is available at [HighlightParser.inc](#)

```
1  <?php
2  /**
3  * Source Code Highlighting
4  *
5  * The classes in this file are responsible for the dynamic @example, @filesource
6  * and {@source} tags output. Using the phpDocumentor_HighlightWordParser,
7  * the phpDocumentor_HighlightParser retrieves PHP tokens one by one from the
8  * array generated by {@link phpDocumentorTWordParser} source retrieval functions
9  * and then highlights them individually.
10 *
11 * It accomplishes this highlighting through the assistance of methods in
12 * the output Converter passed to its parse() method, and then returns the
13 * fully highlighted source as a string
14 *
15 * phpDocumentor :: automatic documentation generator
16 *
17 * PHP versions 4 and 5
18 *
19 * Copyright (c) 2002-2008 Gregory Beaver
20 *
21 * LICENSE:
22 *
23 * This library is free software; you can redistribute it
24 * and/or modify it under the terms of the GNU Lesser General
25 * Public License as published by the Free Software Foundation;
26 * either version 2.1 of the License, or (at your option) any
27 * later version.
28 *
29 * This library is distributed in the hope that it will be useful,
30 * but WITHOUT ANY WARRANTY; without even the implied warranty of
31 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
32 * Lesser General Public License for more details.
33 *
34 * You should have received a copy of the GNU Lesser General Public
35 * License along with this library; if not, write to the Free Software
36 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
37 *
38 * @category ToolsAndUtilities
39 * @package phpDocumentor
40 * @subpackage Parsers
41 * @author Gregory Beaver <cellog@php.net>
42 * @copyright 2002-2008 Gregory Beaver
43 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
44 * @version CVS: $Id: HighlightParser.inc 253641 2008-02-24 02:35:44Z ashnazg $
45 * @filesource
46 * @link http://www.phpdoc.org
47 * @link http://pear.php.net/PhpDocumentor
48 * @tutorial tags.example.pkg, tags.filesource.pkg, tags.inlinesource.pkg
49 * @since 1.2.0beta3
50 * @todo CS cleanup - change package to PhpDocumentor
51 */
52 /**
53 * Retrieve tokens from an array of tokens organized by line numbers
54 *
55 * @category ToolsAndUtilities
56 * @package phpDocumentor
57 * @subpackage Parsers
58 * @author Gregory Beaver <cellog@php.net>
59 * @copyright 2002-2008 Gregory Beaver
60 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
61 * @version Release: @VER@
62 * @link http://www.phpdoc.org
63 * @link http://pear.php.net/PhpDocumentor
64 * @since 1.2.0beta3
65 * @todo CS cleanup - change package to PhpDocumentor
66 * @todo CS cleanup - change class name to PhpDocumentor_*
```

```

68 */
69 class phpDocumentor_HighlightWordParser extends phpDocumentorTWordParser
70 {
71     /**
72      * Hash used to keep track of line numbers that have already been initialized
73      * @var array
74      * @access private
75     */
76     var $_listLineNums = array();
77     /**
78      * Initialize the parser object
79      *
80      * @param array &$input the input
81      * @param phpDocumentor_HighlightParser &$parser the parser
82      *
83      * @return void
84     */
85     function setup(& $input, & $parser)
86     {
87         $this-> parser = & $parser;
88         $this-> data = & $input;
89         $this-> _all = $input;
90         $this-> _sourceline = 0;
91         $this-> pos = 0;
92         $this-> linenum = 0;
93     }
94     /**
95      * debugging function
96      *
97      * @return void
98      * @access private
99      */
100    function printState()
101    {
102        $linenum = $this-> linenum;
103        $pos = $this-> pos;
104        if (!isset($this-> _all[$this-> linenum][$this-> pos])) {
105            $linenum++;
106            $pos = 0;
107        }
108        $details = '';
109        $token = $this-> _all[$linenum][$pos];
110        if (is_array($token)) {
111            $details = token_name($token[0]);
112            $token = htmlspecialchars($token[1]);
113        } else {
114            $token = htmlspecialchars($token);
115        }
116        debug('Next Token ' . $this-> linenum . '-' . $this-> pos . ':' . $details);
117        var_dump($token);
118    }
119    /**
120     * Retrieve the position of the next token that will be parsed
121     * in the internal token array
122     *
123     * @return array format: array(line number, position)
124     */
125    function nextToken()
126    {
127        $linenum = $this-> linenum;
128        $pos = $this-> pos;
129        if (!isset($this-> _all[$this-> linenum][$this-> pos])) {
130            $linenum++;
131            $pos = 0;
132        }
133        if (!isset($this-> _all[$linenum][$pos])) {
134            return false;
135        }
136        return array($linenum, $pos);
137    }
138    /**
139     * Retrieve the next token
140     *
141     * @return array|string either array(PHP token constant, token) or string
142     *                      non-specific separator
143     */
144    function getWord()
145    {
146

```

```

148     {
149         if (!isset($this-> _all[$this-> linenum][$this-> pos])) {
150             $this-> linenum++;
151             $this-> pos = 0;
152             if (!isset($this-> _all[$this-> linenum])) {
153                 return false;
154             }
155             $this-> _parser-> newLineNum();
156             return $this-> getWord();
157         }
158         $word = $this-> _all[$this-> linenum][$this-> pos++];
159         return str_replace("\t" , ' ', $word);
160     }
161
162     /**
163      * back the word parser to the previous token as defined by $last_token
164      *
165      * @param array/string $last_token token, or output from {@link nextToken()}
166      * @param bool $is_pos if true, backupPos interprets $last_token
167      *                      to be the position in the internal token
168      *                      array of the last token
169      *
170      * @return void
171     */
172     function backupPos($last_token, $is_pos = false)
173     {
174         if (!$last_token) {
175             return;
176         }
177         if ($is_pos) {
178             $this-> linenum = $last_token[0];
179             $this-> pos = $last_token[1];
180             return;
181         }
182         if ($last_token === false) {
183             return;
184         }
185
186         //fancy_debug('before', $this->linenum, $this->pos,
187         //token_name($this->_all[$this->linenum][$this->pos][0]),
188         //htmlentities($this->_all[$this->linenum][$this->pos][1]),
189         // $this->_all[$this->linenum]);
190
191         do {
192             $this-> pos--;
193             if ($this-> pos < 0) {
194                 $this-> linenum--;
195                 if ($this-> linenum < 0) {
196                     var_dump($last_token);
197                     break;
198                 }
199                 $this-> pos = count($this-> _all[$this-> linenum]) - 1;
200             }
201         } while (!$this-> tokenEquals($last_token, str_replace("\t" , ' ', $this-> _all[$this-> linenum][$this-> pos])));
202
203         //fancy_debug('after', $this->linenum, $this->pos,
204         //token_name($this->_all[$this->linenum][$this->pos][0]),
205         //htmlentities($this->_all[$this->linenum][$this->pos][1]));
206     }
207 }
208
209 /**
210  * Highlights source code using {@link parse()}
211  *
212  * @category ToolsAndUtilities
213  * @package phpDocumentor
214  * @subpackage Parsers
215  * @author Gregory Beaver <cellog@php.net>
216  * @copyright 2002-2008 Gregory Beaver
217  * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
218  * @version Release: @VER@
219  * @link http://www.phpdoc.org
220  * @link http://pear.php.net/PhpDocumentor
221  * @since 1.2.0beta3
222  * @todo CS cleanup - change package to PhpDocumentor
223  * @todo CS cleanup - change class name to PhpDocumentor_*
224  */
225 class phpDocumentor_HighlightParser extends phpDocumentorTParser
226 {

```

```

228     /**#@+
229      * @access private
230      */
231
232     /**
233      * Highlighted source is built up in this string
234      * @var string
235      */
236     var $_output;
237
238     /**
239      * contents of the current source code line as it is parsed
240      * @var string
241      */
242     var $_line;
243
244     /**
245      * Used to retrieve highlighted tokens
246      * @var Converter a descendant of Converter
247      */
248     var $_converter;
249
250     /**
251      * Path to file being highlighted, if this is from a @filesource tag
252      * @var false/string full path
253      */
254     var $_filesourcepath;
255
256     /**
257      * @var array
258      */
259     var $eventHandlers = array(
260         PARSER_EVENT_ARRAY => 'defaultHandler',
261         PARSER_EVENT_CLASS => 'handleClass',
262         PARSER_EVENT_COMMENT => 'handleComment',
263         PARSER_EVENT_DOCBLOCK_TEMPLATE => 'handleDocBlockTemplate',
264         PARSER_EVENT_END_DOCBLOCK_TEMPLATE => 'handleEndDocBlockTemplate',
265         PARSER_EVENT_LOGICBLOCK => 'handleLogicBlock',
266         PARSER_EVENT_METHOD_LOGICBLOCK => 'handleMethodLogicBlock',
267         PARSER_EVENT_NOEVENTS => 'defaultHandler',
268         PARSER_EVENT_OUTPHP => 'defaultHandler',
269         PARSER_EVENT_CLASS_MEMBER => 'handleClassMember',
270         PARSER_EVENT_DEFINE => 'defaultHandler',
271         PARSER_EVENT_DEFINE_PARAMS => 'defaultHandler',
272         PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS => 'defaultHandler',
273         PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS => 'defaultHandler',
274         PARSER_EVENT_DOCBLOCK => 'handleDocBlock',
275         PARSER_EVENT_TAGS => 'handleTags',
276         PARSER_EVENT_DESC => 'handleDesc',
277         PARSER_EVENT_DOCKEYWORD => 'handleTag',
278         PARSER_EVENT_DOCKEYWORD_EMAIL => 'handleDockeywordEmail',
279         PARSER_EVENT_EOFQUOTE => 'handleQuote',
280         PARSER_EVENT_FUNCTION => 'handleFunction',
281         PARSER_EVENT_METHOD => 'handleMethod',
282         PARSER_EVENT_FUNCTION_PARAMS => 'handleFunctionParams',
283         PARSER_EVENT_FUNC_GLOBAL => 'handleFuncGlobal',
284         PARSER_EVENT_INLINE_DOCKEYWORD => 'handleInlineDockeyword',
285         PARSER_EVENT_INCLUDE => 'defaultHandler',
286         PARSER_EVENT_INCLUDE_PARAMS => 'defaultHandler',
287         PARSER_EVENT_QUOTE => 'handleQuote',
288         PARSER_EVENT_QUOTE_VAR => 'handleQuoteVar',
289         PARSER_EVENT_PHPCODE => 'handlePhpCode',
290         PARSER_EVENT_SINGLEQUOTE => 'handleSingleQuote',
291         PARSER_EVENT_STATIC_VAR => 'defaultHandler',
292         PARSER_EVENT_STATIC_VAR_VALUE => 'defaultHandler',
293         PARSER_EVENT_VAR => 'handleVar',
294     );
295
296     /**
297      * event handlers for @tags
298      * @tutorial tags.pkg
299      */
300     var $tagHandlers = array(
301         '*' => 'defaultTagHandler',
302         'abstract' => 'coreTagHandler',
303         'access' => 'coreTagHandler',
304         'author' => 'coreTagHandler',
305         'category' => 'coreTagHandler',
306         'copyright' => 'coreTagHandler',
307         'deprecated' => 'coreTagHandler',

```

```

308     'example'      => 'coreTagHandler',
309     'filesource'   => 'coreTagHandler',
310     'final'        => 'coreTagHandler',
311     'global'       => 'globalTagHandler',
312     'ignore'       => 'coreTagHandler',
313     'license'      => 'coreTagHandler',
314     'link'         => 'coreTagHandler',
315     'name'         => 'coreTagHandler',
316     'package'      => 'coreTagHandler',
317     'param'        => 'paramTagHandler',
318     'parameter'    => 'paramTagHandler',
319     'see'          => 'coreTagHandler',
320     'since'        => 'coreTagHandler',
321     'subpackage'   => 'coreTagHandler',
322     'internal'    => 'coreTagHandler',
323     'return'       => 'returnTagHandler',
324     'static'       => 'coreTagHandler',
325     'staticvar'    => 'staticvarTagHandler',
326     'throws'       => 'coreTagHandler',
327     'todo'         => 'coreTagHandler',
328     'tutorial'     => 'coreTagHandler',
329     'uses'         => 'coreTagHandler',
330     'var'          => 'varTagHandler',
331     'version'      => 'coreTagHandler',
332     'property'     => 'propertyTagHandler',
333     'property-read'=> 'propertyTagHandler',
334     'property-write'=> 'propertyTagHandler',
335     'method'       => 'propertyTagHandler'
336   );
337 /**
338 */
339 /**
340 * wraps the current line (via the converter) and resets it to empty
341 *
342 * @return void
343 * @uses Converter::SourceLine() encloses {@link $_line} in a
344 *        converter-specific format
345 */
346 function newLineNum()
347 {
348     if ($this-> _pf_no_output_yet) {
349         return;
350     }
351     $this-> _flush_save();
352     $this-> _line .= $this-> _converter-> flushHighlightCache();
353     $this-> _output .= $this-> _converter-> SourceLine($this-> _wp-> linenum,
354             $this-> _line, $this-> _path);
355     $this-> _line = '';
356 }
357 /**
358 * Start the parsing at a certain line number
359 *
360 * @param int $num line number
361 *
362 * @return void
363 */
364 function setLineNum($num)
365 {
366     $this-> _wp-> linenum = $num;
367 }
368 /**
369 * Parse a new file
370 *
371 * The parse() method is a do...while() loop that retrieves tokens one by
372 * one from the {@link $_event_stack}, and uses the token event array set up
373 * by the class constructor to call event handlers.
374 *
375 * The event handlers each process the tokens passed to them, and use the
376 * {@link _addoutput()} method to append the processed tokens to the
377 * {@link $_line} variable. The word parser calls {@link newLineNum()}
378 * every time a line is reached.
379 *
380 * In addition, the event handlers use special linking functions
381 * {@link _link()} and its cousins (_classlink(), etc.) to create in-code
382 * hyperlinks to the documentation for source code elements that are in the
383 * source code.
384 *
385 * @param array      &$parse_data      the parse data

```

```

388 * @param Converter      &$converter          the converter object
389 * @param bool            $inlinesourceparse whether this data is from an
390 *                         inline {@source} tag
391 * @param string/false   $class               if a string, it is the name of the
392 *                         class whose method we are parsing
393 * @param false/integer  $linenum             containing a {@source} tag
394 * @param false/string    $filesourcepath     starting line number from
395 *                                         {@source linenum}
396 * @param false/string    $filesourcepath     full path to file with @filesource
397 *                                         tag, if this is a @filesource parse
398 *
399 * @staticvar int         used for recursion limiting if a handler for
400 *                         an event is not found
401 * @return bool
402 * @uses setupStates()   initialize parser state variables
403 * @uses configWordParser() pass $parse_data to prepare retrieval of tokens
404 * @todo CS cleanup - rename tokenizer_ext constant to uppercase
405 */
406 function parse (& $parse_data, & $converter, $inlinesourceparse = false,
407                 $class = false, $linenum = false, $filesourcepath = false)
408 {
409     if (!tokenizer_ext) {
410         if (is_array($parse_data)) {
411             $parse_data = join($parse_data, '');
412         }
413         $parse_data = explode("\n", $parse_data);
414         $this-> _output = '';
415         foreach ($parse_data as $linenum => $line) {
416             if ($linenum > 0) {
417                 $this-> _output .= $converter-> SourceLine($linenum,
418                                                 $line, $filesourcepath);
419             }
420         }
421         return $converter-> PreserveWhiteSpace($this-> _output);
422     }
423     static $endrecur = 0;
424     $this-> _converter = & $converter;
425     $converter-> startHighlight();
426     $this-> _path = $filesourcepath;
427     $this-> setupStates($inlinesourceparse, $class);
428
429     $this-> configWordParser($parse_data);
430     if ($linenum !== false) {
431         $this-> setLineNum($linenum);
432     }
433     // initialize variables so E_ALL error_reporting doesn't complain
434     $pevent = 0;
435     $word = 0;
436
437     do {
438         $lpevent = $pevent;
439         $pevent = $this-> _event_stack-> getEvent();
440         if ($lpevent != $pevent) {
441             $this-> _last_pevent = $lpevent;
442         }
443
444         if ($pevent == PARSER_EVENT_CLASS_MEMBER) {
445             $this-> _wp-> setWhitespace(true);
446         } else {
447             $this-> _wp-> setWhitespace(false);
448         }
449
450         if (!is_array($word)) {
451             $lw = $word;
452         }
453         if (is_array($word) && $word[0] != T_WHITESPACE) {
454             $lw = $word;
455         }
456         $dbg_linenum = $this-> _wp-> linenum;
457         $dbg_pos = $this-> _wp-> getPos();
458         $word = $this-> _wp-> getWord();
459         if (is_array($word) && ($word[0] == T_WHITESPACE ||
460             $word[0] == T_COMMENT) &&
461             $pevent != PARSER_EVENT_CLASS_MEMBER)
462         {
463             //debug("added " . $this-> _wp->linenum . '-' . $this-> _wp-
464             >pos);
465             $this-> _addoutput($word);
466             continue;
467         } else {

```

```

467     $this-> _pv_last_word = $lw;
468 }
469 if ($pevent != PARSER_EVENT_DOCBLOCK) {
470     $this-> _pv_last_next_word = $this-> _pv_next_word;
471     $this-> _pv_next_word = $this-> _wp-> nextToken();
472 }
473 // in wordparser, have to keep track of lines
474 // $this->publishEvent(PHPDOCUMENTOR_EVENT_NEWLINENUM,
475 //   $this->_wp->linenum);
476 if (PHPDOCUMENTOR_DEBUG == true) {
477     echo "LAST: ";
478     if (is_array($this-> _pv_last_word)) {
479         echo token_name($this-> _pv_last_word[0]) .
480             ' => | ' .
481             htmlspecialchars($this-> _pv_last_word[1]);
482     } else {
483         echo " | " . $this-> _pv_last_word;
484     }
485     echo "|\\n";
486     echo "PEVENT: " . $this-> getParserEventName($pevent) .
487     "\n";
488 ;
489     echo "LASTPEVENT: "
490     $this-> getParserEventName($this-> _last_pevent) . "\n";
491 //echo "LINE: " . $this->_line . "\n";
492 //echo "OUTPUT: " . $this->_output . "\n";
493 echo $dbg_lineno . '-' . $dbg_pos . ":" ;
494 if (is_array($word)) {
495     echo token_name($word[0]) . ' => | ' . htmlspecialchars($word[1]);
496 } else {
497     echo ' | ' . htmlspecialchars($word);
498 }
499 echo "|\\n" ;
500 $this-> _wp-> printState();
501 echo "NEXT TOKEN: ";
502 $tok1 = $this-> _pv_next_word;
503 $tok = $this-> _wp-> _all[$tok1[0]][$tok1[1]];
504 if (is_array($tok)) {
505     echo token_name($tok[0]) . ' => ' . $tok1[0] . '-' . $tok1[1] .
506     ' | ' . htmlspecialchars($tok[1]);
507 } else {
508     echo " | " . $tok;
509 }
510 echo "|\\n" ;
511 echo "-----\\n\\n\\n" ;
512 flush();
513 }
514 if ($word !== false && isset($this-> eventHandlers[$pevent])) {
515     $handle = $this-> eventHandlers[$pevent];
516     $this-> $handle($word, $pevent);
517 } elseif ($word !== false) {
518     debug('WARNING: possible error, no handler for event number ' .
519           $pevent);
520     if ($endrecur++ == 25) {
521         die("FATAL ERROR, recursion limit reached");
522     }
523 } while (!$word === false);
524 if (strlen($this-> _line)) {
525     $this-> newLineNum();
526 }
527 return $this-> _output;
528 }
529 /**
530 * Event Handlers
531 *
532 * All Event Handlers use {@link checkEventPush()} and
533 * {@link checkEventPop()} to set up the event stack and parser state.
534 *
535 * @param string/array $word token value
536 * @param int $pevent parser event from {@link Parser.inc}
537 *
538 * @return void
539 * @access private
540 */
541 /**
542 * Most tokens only need highlighting, and this method handles them
543 */
544 function defaultHandler($word, $pevent)
545 {

```

```

546     $this-> _addoutput($word);
547     if ($this-> checkEventPush($word, $pevent)) {
548         return;
549     }
550     $this-> checkEventPop($word, $pevent);
551 }
552 /**
553 * Handles global declarations in a function, like:
554 *
555 * <code>
556 * function foobar()
557 * {
558 *     global $_phpDocumentor_setting;
559 * }
560 * </code>
561 *
562 * @uses _globallink() instead of _addoutput(), to link to global variables
563 *      if they are used in a function
564 */
565 function handleFuncGlobal($word, $pevent)
566 {
567     if ($this-> checkEventPush($word, $pevent)) {
568         return;
569     }
570     $this-> _globallink($word);
571     $this-> checkEventPop($word, $pevent);
572 }
573 /**
574 * Handles strings in quotation marks and heredoc
575 *
576 * Special handling is needed for strings that contain variables like:
577 *
578 * <code>$a = "$test string"</code>
579 *
580 * The tokenizer parses out tokens ''',array(T_VARIABLE,'$test'),' string',
581 * and '''. Since it is possible to have $this->classvar in a string,
582 * we save a variable name just in case the next token is -> to allow linking
583 * to class members. Otherwise, the string is simply highlighted.
584 *
585 * constant strings (with no $variables in them) are passed as a single
586 * entity, and so will be saved in the last token parsed. This means the
587 * event handler must tell the word parser to re-retrieve the current token
588 * so that the correct event handler can process it.
589 */
590 function handleQuote($word, $pevent)
591 {
592     if ($this-> _pf_inmethod &&           is_array($word) &&           $word[0] ==
593 T_VARIABLE) {
594         $this-> _pv_lastvar = $word;
595     }
596     if ($this-> checkEventPush($word, $pevent)) {
597         $this-> _addoutput($word);
598         return;
599     }
600     if ($this-> _pf_quote_active &&
601         ((($this-> _pv_last_word == '') &&
602           $this-> _last_pevent != PARSER_EVENT_QUOTE) ||
603           (is_array($this-> _pv_last_word) &&
604             $this-> _pv_last_word[0] == T_END_HEREDOC &&
605             $this-> _last_pevent != PARSER_EVENT_EOFQUOTE)))
606     ) {
607         $this-> _pf_quote_active = false;
608         $wp-> backupPos($word);
609         $event_stack-> popEvent();
610         return;
611     }
612     if (!$this-> _pf_quote_active &&
613         ((($this-> _pv_last_word == '') &&
614           $this-> _last_pevent != PARSER_EVENT_QUOTE) ||
615           (is_array($this-> _pv_last_word) &&
616             $this-> _pv_last_word[0] == T_END_HEREDOC &&
617             $this-> _last_pevent != PARSER_EVENT_EOFQUOTE)))
618     ) {
619         if (is_array($word) &&           $word[0] == T_VARIABLE) {
620             $this-> _pv_lastvar = $word;
621         }
622         $this-> _pf_quote_active = true;
623         $this-> _save_highlight_state = $this-> _converter-> getHighlightState();

```

```

625         $this-> _converter-> startHighlight();
626         $this-> _addoutput($word);
627         $this-> checkEventPop($word, $pevent);
628         return;
629     } elseif (is_array($this-> _pv_last_word) &&
630             $this-> _pv_last_word[0] == T_CONSTANT_ENCAPSED_STRING
631     ) {
632         // $this-> _pv_quote_data = $this-> _pv_last_word[1];
633         $this-> _event_stack-> popEvent();
634         $this-> _wp-> backupPos($word);
635         return;
636     }
637     if ($this-> checkEventPop($word, $pevent)) {
638         $this-> _pf_quote_active = false;
639     }
640     $this-> _addoutput($word);
641 }
642
643 /**
644 * Handles {$variable} within a "quote"
645 *
646 * This is a simple handler, for a very complex
647 * array of legal syntax. It is legal to nest control structures
648 * inside the {}, and other weird stuff.
649 */
650 function handleQuoteVar($word, $pevent)
651 {
652     if ($this-> checkEventPop($word, $pevent)) {
653         $this-> _pf_quote_active = true;
654         $this-> _addoutput($word);
655         return;
656     }
657     if ($this-> _pf_inmethod && is_array($word) && $word[0] ==
658         T_VARIABLE) {
659         $this-> _pv_lastvar = $word;
660     }
661     if ($this-> checkEventPush($word, $pevent)) {
662         $this-> _pf_quote_active = false;
663         if (is_string($word) && ( $word == '{' || $word == '}' || $word ==
664             '''))
665         ) {
666             $this-> _pf_quote_active = true;
667             $this-> _pv_lastvar = false;
668         }
669     }
670     $this-> _addoutput($word);
671
672 /**
673 * Handles define() statements
674 *
675 * The only thing this handler cares about is retrieving the name of the
676 * define variable, and the end of the define statement, so after the name
677 * is found, it simply makes sure parentheses are matched as in this case:
678 *
679 * <code>
680 * define("test",array("hello",6 => 4, 5 => array('there')));
681 * </code>
682 *
683 * This handler and the DEFINE_PARAMS_PARENTHESIS handler (which is just
684 * {@link defaultHandler()} in this version, as nothing fancy is needed)
685 * work together to ensure proper parenthesis matching.
686 *
687 * If the define variable is documented, a link will be created to its
688 * documentation using the Converter passed.
689 */
690 function handleDefine($word, $pevent)
691 {
692     static $token_save;
693     if (!isset($token_save)) {
694         $token_save = array();
695     }
696     $e = $this-> checkEventPush($word, $pevent);
697     if ($e && $e != PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS) {
698         return;
699     }
700     if (!isset($this-> _pv_define_params_data)) {
701         $this-> _pv_define_params_data = '';
702     }

```

```

703
704     if ($this-> checkEventPop($word, $pevent)) {
705         unset($token_save);
706         $this-> _addoutput($word);
707     }
708     if ($this-> _pf_definename_isset) {
709         $this-> _addoutput($word);
710     } else {
711         if ($word != ",") {
712             $token_save[] = $word;
713             if (is_array($word)) {
714                 $word = $word[1];
715             }
716             $this-> _pv_define_params_data .= $word;
717         } else {
718             if (substr($this-> _pv_define_params_data, 0, 1) ==
719                 substr($this-> _pv_define_params_data,
720                     strlen($this-> _pv_define_params_data) - 1) &&
721                     in_array(substr($this-> _pv_define_params_data, 0, 1),
722                         array(' ', "\n")))
723             ) {
724                 // remove leading and ending quotation marks
725                 // if there are only two
726                 $a = substr($this-> _pv_define_params_data, 0, 1);
727                 $b = substr($this-> _pv_define_params_data, 1,
728                     strlen($this-> _pv_define_params_data) - 2);
729                 if (strpos($b, $a) == false) {
730                     $this-> _pv_define_params_data = $b;
731                 }
732             }
733             $this-> _pf_definename_isset = true;
734
735             $link = $this-> _converter-> getLink($this-> _pv_define_params_data);
736             foreach ($token_save as $token) {
737                 if (is_object($link)) {
738                     if (is_array($token)) {
739                         $token = $token[1];
740                     }
741                     $this-> _addoutput($this-> _converter-> returnSee($link,
742                         $token));
743                 } else {
744                     $this-> _addoutput($save, $token);
745                 }
746             }
747             $this-> _pv_define_params_data = '';
748         }
749     }
750 }
751 /**
752 * Handles normal global code. Special consideration is taken for DocBlocks
753 * as they need to retrieve the whole DocBlock before doing any output, so
754 * the parser flag {@link $_pf_no_output_yet} is set to tell
755 * {@link _addoutput()} not to spit anything out yet.
756 */
757
758 * @uses _link() make any global code that is a documentable element link
759 *       to the php manual or its documentation
760 */
761 function handlePhpCode($word, $pevent)
762 {
763     $test = $this-> checkEventPush($word, $pevent);
764     if ($test == PARSER_EVENT_DOCBLOCK || $test == PARSER_EVENT_COMMENT) {
765         if (substr($word[1], 0, 2) == '/*' && strpos($word[1], '*/')) {
766             $this-> _pv_last_word = $word;
767             if ($word[1] == '/*#@-*/') {
768                 $this-> _pf_docblock_template = true;
769             } else {
770                 $this-> _pf_docblock = true;
771             }
772             return $this-> handleDocBlock($word, PARSER_EVENT_DOCBLOCK);
773         }
774         $this-> _pf_no_output_yet = true;
775         $this-> _pv_saveline = $this-> _wp-> linenum + 1;
776         return;
777     }
778     if (is_array($word) && $word[0] == T_DOUBLE_COLON) {
779         $this-> _pf_colon_colon = true;
780     }
781     if (!$this-> _pf_colon_colon && is_array($word) && $word[0] ==
T_STRING) {

```

```

782         $this-> _pv_last_string = $word;
783     }
784     $this-> _link($word);
785     $this-> checkEventPop($word, $pevent);
786 }
787 /**
788 * Handle the function declaration header
789 *
790 * This handler only sees the "function name" portion of the function
791 * declaration. Handling of the function parameters is by
792 * {@link handleFunctionParams()}, and the function body is handled by
793 * {@link handleLogicBlock()}
794 */
795 function handleFunction($word, $pevent)
796 {
797     if ($this-> checkEventPush($word, $pevent)) {
798         $this-> _addoutput($word);
799         return;
800     }
801     if ($this-> checkEventPop($word, $pevent)) {
802         return;
803     }
804     $this-> _link($word);
805 }
806 /**
807 * Handle the method declaration header
808 *
809 * This handler only sees the "function name" portion of the method
810 * declaration. Handling of the method parameters is by
811 * {@link handleFunctionParams()}, and the method body is handled by
812 * {@link handleMethodLogicBlock()}
813 */
814 function handleMethod($word, $pevent)
815 {
816     if ($this-> checkEventPush($word, $pevent)) {
817         $this-> _addoutput($word);
818         return;
819     }
820     if ($this-> checkEventPop($word, $pevent)) {
821         if ($word == ';') {
822             $this-> _addoutput($word);
823         }
824         return;
825     }
826     $this-> _methodlink($word);
827 }
828 /**
829 * Handler for the stuff between ( and ) in a function declaration
830 *
831 * <code>
832 * function handles($only,$these,$parameters){...}
833 * </code>
834 */
835 function handleFunctionParams($word, $pevent)
836 {
837     if ($this-> checkEventPush($word, $pevent)) {
838         $this-> _addoutput($word);
839         return;
840     }
841     $this-> _addoutput($word);
842     $this-> checkEventPop($word, $pevent);
843 }
844 /**
845 * Handler for function body.
846 *
847 * The function body is checked for php functions, documented constants,
848 * functions, and indirectly for global statements. It hyperlinks to the
849 * documentation for detected elements is created. Everything else is
850 * highlighted normally.
851 */
852 function handleLogicBlock($word, $pevent)
853 {
854     if ($this-> checkEventPush($word, $pevent)) {
855         $this-> _addoutput($word);
856         return;
857     }

```

```

862     if (is_array($word) && $word[0] == T_DOUBLE_COLON) {
863         $this-> _pf_colon_colon = true;
864     }
865     if (!$this-> _pf_colon_colon && is_array($word) && $word[0] ==
T_STRING) {
866         $this-> _pv_last_string = $word;
867     }
868     $this-> _link($word);
869     if ($this-> checkEventPop($word, $pevent)) {
870         $e = $this-> _event_stack-> popEvent();
871         $this-> _event_stack-> pushEvent($e);
872         if ($e == PARSER_EVENT_FUNCTION) {
873             $this-> _wp-> backupPos($word);
874         }
875     }
876 }
877
878 /**
879 * Handler for method body.
880 *
881 * Like functions, the method body is checked for php functions, documented
882 * constants, functions, and indirectly for global statements. It also
883 * checks for "$this->XXXX" where XXXX is a class variable or method, and
884 * links to the documentation for detected elements is created. Everything
885 * else is highlighted normally.
886 */
887 function handleMethodLogicBlock($word, $pevent)
888 {
889     if (isset($this-> _pv_prev_var_type)) {
890         //debug('prevtype is set');
891         if (!is_array($word)) {
892             unset($this-> _pv_prev_var_type);
893         } else {
894             if ($word[0] != T_WHITESPACE &&
895                 $word[0] != T_STRING && $word[0] != T_OBJECT_OPERATOR
896             ) {
897                 //fancy_debug('unset', $word);
898                 unset($this-> _pv_prev_var_type);
899             }
900         }
901     }
902     $this-> _pf_inmethod = true;
903     if ($e = $this-> checkEventPush($word, $pevent)) {
904         $this-> _addoutput($word);
905         if ($e == PARSER_EVENT_CLASS_MEMBER) {
906             $this-> _pf_no_output_yet = true;
907         }
908         return;
909     }
910     if (is_array($word) && $word[0] == T_DOUBLE_COLON) {
911         $this-> _pf_colon_colon = true;
912     }
913     if (!$this-> _pf_colon_colon && is_array($word) && $word[0] ==
T_STRING) {
914         $this-> _pv_last_string = $word;
915     }
916     if (is_array($word) && $word[0] == T_VARIABLE) {
917         $this-> _pv_lastvar = $word;
918     }
919     $this-> _link($word);
920     if ($this-> checkEventPop($word, $pevent)) {
921         $this-> _pf_inmethod = false;
922         $e = $this-> _event_stack-> popEvent();
923         $this-> _event_stack-> pushEvent($e);
924         if ($e == PARSER_EVENT_METHOD) {
925             $this-> _wp-> backupPos($word);
926         }
927     }
928 }
929
930 /**
931 * Handles $obj->classmember in a method body
932 *
933 * This handler is responsible for linking to the documentation of a
934 * class member when it is used directly in a method body.
935 *
936 * There are two methods of determining whether to link:
937 * - $this->member
938 * - $this->member->submember
939 */

```

```

940 * The first case is handled by the $pv_lastvar variable, and the
941 * second case is handled by the $pv_prev_var_type variable. $pv_lastvar
942 * is always set to the value of the last T_VARIABLE token, if and only if
943 * no text has occurred between the variable and a T_OBJECT_OPERATOR token
944 * ">". handleClassMember will only link if the last variable encountered
945 * was $this.
946 *
947 * When $this->variable is encountered, the variable is looked up to see
948 * if it can be found, and if so, the contents of its @var tag are processed
949 * to see if the member variable is defined to have 1 and only 1 class.
950 * If so, the $pv_prev_var_type variable is set to this classname. When
951 * submember is processed, the HighlightParser checks to see if
952 * $pv_prev_var_type::submember() or $pv_prev_var_type::$submember exists,
953 * and if it does, it is linked to.
954 */
955 function handleClassMember($word, $pevent)
956 {
957     if (!isset($this-> _pv_lastvar) && !isset( $this-> _pv_prev_var_type)) {
958         //fancy_debug('returned from', $word, $this->_pv_prev_var_type);
959         $this-> _pf_no_output_yet = false;
960         $this-> _event_stack-> popEvent();
961         return $this-> defaultHandler($word, $pevent);
962     }
963     if (isset($this-> _pv_cm_name)) {
964         $this-> _pf_obj_op = false;
965         $name = $this-> _pv_cm_name;
966         unset($this-> _pv_cm_name);
967         //debug('unset pvcname');
968         $this-> _event_stack-> popEvent();
969         // control variable for _pv_prev_var_type
970         $setnow = false;
971         if ((isset($this-> _pv_lastvar) && $this-> _pv_lastvar[1] == '$this') ||
972             isset($this-> _pv_prev_var_type))
973     {
974         if (is_array($word) && $word[0] == T_WHITESPACE) {
975             // preserve value of _pv_prev_var_type
976             $setnow = true;
977             $save = $this-> _wp-> nextToken();
978             $temp = $this-> _wp-> getWord();
979             $this-> _wp-> backupPos($save, true);
980         }
981         if ((is_string($word) && $word == '(') || (isset($temp) &&
982             is_string($temp) && $temp == '(')
983         ) {
984             // it's a function
985             $this-> _pf_no_output_yet = false;
986             $this-> _methodlink($name);
987             unset($this-> _pv_prev_var_type);
988         } else {
989             // it's a variable
990             //fancy_debug('name is ', $name);
991             $this-> _pf_no_output_yet = false;
992             $this-> _varlink($name, true);
993             $templink =
994                 $this-> _converter-> getLink('object ' . $this-> _pv_class);
995             $class = false;
996             if (is_object($templink)) {
997                 $class = $this-> _converter-> classes
998                     -> getClass($templink-> name, $templink-> path);
999             }
1000             if ($class) {
1001                 $varname = $name;
1002                 if (is_array($varname)) {
1003                     $varname = $name[1];
1004                 }
1005                 if ($varname[0] != '$') {
1006                     $varname = '$' . $varname;
1007                 }
1008                 $var = $class-> getVar($this-> _converter, $varname);
1009             }
1010             if (is_object($var) && $var-> docblock-> var) {
1011                 $type = $var-> docblock-> var-> returnType;
1012             }
1013             if (isset($type)) {
1014                 if (strpos($type, 'object') === false) {
1015                     $type = 'object ' . $type;
1016                 }
1017                 $type = $this-> _converter-> getLink($type);
1018                 if (phpDocumentor_get_class($type) == 'classlink') {
1019                     // the variable's type is a class,

```

```

1020                                // save it for future ->
1021                                //fancy_debug('set prev_var_type!', $type->name);
1022                                $setnow = true;
1023                                $this-> _pv_prev_var_type = $type-> name;
1024                            } else {
1025                                unset($this-> _pv_prev_var_type);
1026                            }
1027                        } else {
1028                            unset($this-> _pv_prev_var_type);
1029                        }
1030                    } else {
1031                        unset($this-> _pv_prev_var_type);
1032                    }
1033                }
1034            } else {
1035                $this-> _pf_no_output_yet = false;
1036                // this does NewLinenum if necessary
1037                $this-> _wp-> backupPos($word);
1038                $this-> _wp-> getWord();
1039                $this-> _addoutput($name);
1040            }
1041            if (!$setnow) {
1042                //debug('unset prevtype, no setnow');
1043                unset($this-> _pv_prev_var_type);
1044            }
1045            unset($this-> _pv_lastvar);
1046            $this-> _pf_no_output_yet = false;
1047            // this does NewLinenum if necessary
1048            $this-> _wp-> backupPos($word);
1049            $this-> _wp-> getWord();
1050            if ($word[0] == T_OBJECT_OPERATOR) {
1051                $this-> _wp-> backupPos($word);
1052            } else {
1053                $this-> _addoutput($word);
1054            }
1055            return;
1056        }
1057        if (!$this-> _pf_obj_op && is_array($this-> _pv_last_word) &&
1058            $this-> _pv_last_word[0] == T_OBJECT_OPERATOR
1059        ) {
1060            if ((isset($this-> _pv_lastvar) && $this-> _pv_lastvar[1] == '$this') ||
1061                isset($this-> _pv_prev_var_type)
1062            ) {
1063                $this-> _pf_obj_op = true;
1064            } else {
1065                $this-> _pf_no_output_yet = false;
1066                // this does NewLinenum if necessary
1067                $this-> _wp-> backupPos($word);
1068                $this-> _wp-> getWord();
1069                $this-> _addoutput($word);
1070                $this-> _event_stack-> popEvent();
1071            }
1072        }
1073        if (is_array($word) && $word == T_WHITESPACE) {
1074            $this-> _pf_no_output_yet = false;
1075            // this does NewLinenum if necessary
1076            $this-> _wp-> backupPos($word);
1077            $this-> _wp-> getWord();
1078            $this-> _addoutput($word);
1079            return;
1080        }
1081        if ($this-> _pf_obj_op) {
1082            if (!(is_array($word) && ($word[0] == T_STRING || $word[0] == T_WHITESPACE)))
1083            ) {
1084                unset($this-> _pv_lastvar);
1085                //debug('unset lastvar');
1086                $this-> _event_stack-> popEvent();
1087                $this-> _pf_no_output_yet = false;
1088                // this does NewLinenum if necessary
1089                $this-> _wp-> backupPos($word);
1090                $this-> _wp-> getWord();
1091                $this-> _addoutput($word);
1092                return;
1093            }
1094            if ($word[0] == T_STRING) {
1095                //fancy_debug('set pvcname to', $word);
1096                $this-> _pv_cm_name = $word;
1097            } else {
1098                $this-> _pf_no_output_yet = false;

```

```

1100          // this does NewLinenum if necessary
1101          $this-> _wp-> backupPos($word);
1102          $this-> _wp-> getWord();
1103          $this-> _addoutput($word);
1104      }
1105  }
1106
1107 /**
1108 * Handles comments
1109 *
1110 * Comments are almost always single-line tokens, and so will be
1111 * in the last word. This handler checks to see if the current token
1112 * is in fact a comment, and if it isn't, it backs up and returns control
1113 * to the parent event handler with that word.
1114 */
1115 function handleComment($word, $pevent)
1116 {
1117     $w = $this-> _pv_last_word;
1118     // don't perform this check if this is a normal comment. Docblocks
1119     // have the _pf_no_output_yet variable set to true
1120     if ($this-> _pf_no_output_yet && is_array($w) &&
1121         (in_array($w[0], array(T_COMMENT, T_DOC_COMMENT)) &&
1122          strpos($w[1], '/**') === 0)
1123     ) {
1124         $this-> _event_stack-> popEvent();
1125         $this-> _event_stack-> pushEvent(PARSER_EVENT_DOCBLOCK);
1126         return $this-> handleDocBlock($word, PARSER_EVENT_DOCBLOCK);
1127     }
1128     if ($this-> _pf_no_output_yet) {
1129         $flag
1130             = 1;
1131         $this-> _pf_no_output_yet = false;
1132         $this-> _addoutput($this-> _pv_last_word);
1133     }
1134     if (!is_array($word) ||
1135         !in_array($word[0], array(T_COMMENT, T_DOC_COMMENT)) ||
1136         (in_array($word[0], array(T_COMMENT, T_DOC_COMMENT)) &&
1137          strpos($word[1], '/**') === 0)
1138     ) {
1139         $this-> _event_stack-> popEvent();
1140         if (strpos($this-> _pv_last_word[1], "\n") != false) {
1141             // $this-> _wp->linenum++;
1142             // $this-> newLineNum();
1143         }
1144         $this-> _wp-> backupPos($this-> _pv_last_word);
1145         $this-> _wp-> getWord();
1146         // var_dump($this-> _wp->nextToken());
1147         return;
1148     } elseif (isset($flag)) {
1149         $this-> newlineNum();
1150     }
1151     $this-> _addoutput($word);
1152     $this-> checkEventPop($word, $pevent);
1153     if (strpos($word[1], '/') === strlen($word[1]) - 2) {
1154         $this-> _event_stack-> popEvent();
1155     }
1156 }
1157 /**
1158 * Handle class declarations
1159 *
1160 * Handles the initial declaration line:
1161 *
1162 * <code>class X</code>
1163 *
1164 * or
1165 *
1166 * <code>class X extends Y implements I</code>
1167 *
1168 * @uses _classlink() to link to documentation for X and for Y class in
1169 *       "class X extends Y"
1170 */
1171 function handleClass($word, $pevent)
1172 {
1173     $this-> _pf_in_class = true;
1174     $a
1175         = $this-> checkEventPush($word, $pevent);
1176     if (!isset($this-> _pv_class) && is_array($word) && $word[0] ==
1177 T_STRING) {
1178         $this-> _pv_class = $this-> _converter-> class = $word[1];

```

```

1179         $this-> _classlink($word);
1180         return;
1181     }
1182
1183     if (is_array($word) &&
1184         in_array($word[0], array(T_PRIVATE, T_PROTECTED, T_PUBLIC)))
1185     {
1186         $starttok = $this-> _wp-> nextToken();
1187         $test    = array(T_WHITESPACE);
1188         while ($test && $test[0] == T_WHITESPACE) {
1189             $tok  = $this-> _wp-> nextToken();
1190             $test = $this-> _wp-> getWord();
1191         } // while
1192
1193         if (is_array($test) && $test[0] == T_VARIABLE) {
1194             $this-> _wp-> backupPos($tok, true);
1195             return;
1196         }
1197         $this-> _wp-> backupPos($starttok, true);
1198     }
1199
1200     if (@in_array($this-> _pv_last_word[0],
1201                  array(T_PRIVATE, T_PROTECTED, T_PUBLIC)))
1202     {
1203         if (is_array($word) && $word[0] == T_VARIABLE) {
1204             $this-> _wp-> backupPos($this-> _pv_last_word);
1205             $this-> _event_stack-> pushEvent(PARSER_EVENT_VAR);
1206             return;
1207         }
1208     }
1209
1210     if ($this-> _pf_extends_found &&
1211         is_array($word) &&
1212         $word[0] == T_STRING) {
1213         $this-> _classlink($word);
1214         return;
1215     }
1216     if (is_array($word) && $word[0] == T_EXTENDS) {
1217         $this-> _pf_extends_found = true;
1218     }
1219     if ($a == PARSER_EVENT_DOCBLOCK) {
1220         $this-> _pf_no_output_yet = true;
1221         $this-> _pv_saveline      = $this-> _wp-> linenum + 1;
1222         return;
1223     }
1224     $this-> _addoutput($word);
1225     if ($this-> checkEventPop($word, $pevent)) {
1226         $this-> _pf_in_class = false;
1227         unset($this-> _pv_class);
1228     }
1229
1230 /**
1231 * Handles class variable declaration
1232 *
1233 * <code>
1234 * class X
1235 * {
1236 *     var $Y;
1237 * </code>
1238 *
1239 * @uses _varlink() make a link to $Y documentation in class variable
1240 *       declaration "var $Y;"
```

```

1241 */
1242 function handleVar($word, $pevent)
1243 {
1244     if ($this-> checkEventPush($word, $pevent)) {
1245         $this-> _addoutput($word);
1246         return;
1247     }
1248     if (is_array($word) && $word[0] == T_VARIABLE) {
1249         return $this-> _varlink($word);
1250     }
1251     $this-> _addoutput($word);
1252     $this-> checkEventPop($word, $pevent);
1253 }
1254
1255 /**
1256 * This handler is responsible for highlighting DocBlocks
1257 *
```

```

1258 * handleDocBlock determines whether the docblock is normal or a template,
1259 * and gathers all the lines of the docblock together before doing any
1260 * processing
1261 *
1262 * As it is not possible to distinguish any comment token from a docblock
1263 * token, this handler is also called for comments, and will pass control
1264 * to {@link handleComment()} if the comment is not a DocBlock
1265 *
1266 * @uses commonDocBlock() once all lines of the DocBlock have been retrieved
1267 */
1268 function handleDocBlock($word, $event)
1269 {
    if (!($this-> _pf_docblock || $this-> _pf_docblock_template)) {
        if (strpos($this-> _pv_last_word[1], '/*+') !== 0) {
            // not a docblock
            $this-> _wp-> backupPos($this-> _pv_last_word);
            $this-> _event_stack-> popEvent();
            $this-> _event_stack-> pushEvent(PARSER_EVENT_COMMENT);
            $this-> _pf_no_output_yet = false;
            return;
        } else {
            $this-> _pf_no_output_yet = true;
            $this-> _pv_db_lines = array();
        }
    }
    $last_word = $this-> _pv_last_word[1];
    $dtype = '_pv_docblock';
    if ($last_word == '/*#@-*/') {
        // stop using docblock template
        $this-> _pf_no_output_yet = false;
        $this-> _addDocBlockOutput('closetemplate', $last_word);
        if ($this-> _pv_next_word !== false) {
            $this-> _wp-> backupPos($this-> _pv_next_word, true);
        }
        $this-> _event_stack-> popEvent();
        return;
    }
    if (!($this-> _pf_docblock || $this-> _pf_docblock_template)) {
        $this-> _pv_db_lines = array();
        if (strpos($last_word, '/*#+') === 0) {
            // docblock template definition
            $this-> _pf_docblock_template = true;
        } else {
            $this-> _pf_docblock = true;
        }
        $this-> _pv_db_lines[] = $last_word;
        if (strpos($last_word, '*/') !== false) {
            $this-> commonDocBlock();
            return;
        }
        $this-> _pv_db_lines[] = $word[1];
        if (strpos($word[1], '*/') !== false) {
            $this-> commonDocBlock();
        }
    } else {
        $this-> _pv_db_lines[] = $word[1];
    }
    if (($this-> _pf_docblock || $this-> _pf_docblock_template) &&
        (strpos($word[1], '*/') !== false))
    ) {
        $this-> commonDocBlock();
    }
}
/**#@-*/
1323 /**
 * This continuation of handleDocBlock splits DocBlock comments up into
 * phpDocumentor tokens. It highlights DocBlock templates in a different
 * manner from regular DocBlocks, recognizes inline tags, regular tags,
 * and distinguishes between standard core tags and other tags, and
 * recognizes parameters to tags like @var.
 *
 * the type in "@var type description" will be highlighted as a php type,
 * and the var in "@param type $var description" will be highlighted as a
 * php variable.
 *
 * @return void
 * @uses handleDesc() highlight inline tags in the description
 * @uses handleTags() highlight all tags
 * @access private

```

```

1338 */
1339 function commonDocBlock()
1340 {
1341     $this-> _event_stack-> popEvent();
1342     $lines = $this-> _pv_db_lines;
1343     $go = count($this-> _pv_db_lines);
1344     for ($i=0; $i < $go; $i++) {
1345         if (substr(trim($lines[$i]), 0, 2) == '*/' || 
1346             substr(trim($lines[$i]), 0, 1) != '*' &&
1347             substr(trim($lines[$i]), 0, 3) != '/*')
1348     ) {
1349         $lines[$i] = array($lines[$i], false);
1350     } elseif (substr(trim($lines[$i]), 0, 3) == '/*') {
1351         $linesi = array();
1352         // remove leading "/*"
1353         $linesi[1] = substr(trim($lines[$i]), 3);
1354         if (empty($linesi[1])) {
1355             $linesi[0] = $lines[$i];
1356         } else {
1357             $linesi[0] =
1358                 substr($lines[$i], 0, strpos($lines[$i], $linesi[1]));
1359         }
1360         $lines[$i] = $linesi;
1361     } else {
1362         $linesi = array();
1363         // remove leading "* "
1364         $linesi[1] = substr(trim($lines[$i]), 1);
1365         if (empty($linesi[1])) {
1366             $linesi[0] = $lines[$i];
1367         } else {
1368             $linesi[0] =
1369                 substr($lines[$i], 0, strpos($lines[$i], $linesi[1]));
1370         }
1371         $lines[$i] = $linesi;
1372     }
1373 }
1374 for ($i = 0; $i < count($lines); $i++) {
1375     if ($lines[$i][1] === false) {
1376         continue;
1377     }
1378     if (substr(trim($lines[$i][1]), 0, 1) == '@' &&
1379         substr(trim($lines[$i][1]), 0, 2) != '@ ')
1380     ) {
1381         $tagindex = $i;
1382         $i = count($lines);
1383     }
1384     if (isset($tagindex)) {
1385         $tags = array_slice($lines, $tagindex);
1386         $desc = array_slice($lines, 0, $tagindex);
1387     } else {
1388         $tags = array();
1389         $desc = $lines;
1390     }
1391     //var_dump($desc, $tags);
1392     $this-> _pf_no_output_yet = false;
1393     $save = $this-> _wp-> linenum = $this-> _wp-> linenum;
1394     $this-> _wp-> linenum = $this-> _pv_saveline;
1395     $this-> handleDesc($desc);
1396     $this-> handleTags($tags);
1397     $this-> _pv_db_lines = array();
1398     $this-> _wp-> linenum = $save;
1399     if (strpos($this-> _pv_last_word[1], '*/') !== false) {
1400         $this-> _wp-> backupPos($this-> _pv_next_word, true);
1401     }
1402     $this-> _pf_docblock = $this-> _pf_docblock_template = false;
1403 }
1404 */
1405 /**
1406 * Handle the description area of a DocBlock
1407 *
1408 * This method simply finds inline tags and highlights them
1409 * separately from the rest of the description.
1410 *
1411 * @param mixed $desc the description piece(s)
1412 *
1413 * @return void
1414 * @uses getInlineTags()
1415 * @access private
1416 */

```

```

1418     function handleDesc($desc)
1419     {
1420         $dbtype = 'docblock';
1421         $dbtype .= ($this-> _pf_docblock ? '' : 'template');
1422         foreach ($desc as $line) {
1423             $this-> getInlineTags($line[0] . $line[1]);
1424             if (strpos($line[0], '*/') === false &&
1425                 !(substr($line[0], 0, 2) == '/*' &&
1426                     strpos($line[1], '*/') != false)
1427             ) {
1428                 $this-> newLineNum();
1429                 $this-> _wp-> linenum++;
1430             }
1431         }
1432         if ($this-> _pf_internal) {
1433             $this-> _pf_internal = false;
1434         }
1435     }
1436
1437 /**
1438 * Handle phpDocumentor tags in a DocBlock
1439 *
1440 * This method uses the {@link $tagHandlers} array to determine which
1441 * method will handle tags found in the docblock, and passes the data to
1442 * the individual handlers one by one
1443 *
1444 * @param array $tags array of tags to handle
1445 *
1446 * @return void
1447 * @access private
1448 */
1449 function handleTags($tags)
1450 {
1451     $newtags = array();
1452     $curtag = array();
1453     for ($i=0; $i < count($tags); $i++) {
1454         $tagsi = trim($tags[$i][1]);
1455         if (substr($tagsi, 0, 1) == '@' && substr($tagsi, 0, 2) != '@ ') {
1456             // start a new tag
1457             $tags[$i][1] = array(substr($tags[$i][1], 0,
1458                                         strpos($tags[$i][1], $tagsi)), $tagsi);
1459             if (!empty($curtag)) {
1460                 $newtags[] = $curtag;
1461                 $curtag = array();
1462             }
1463             $curtag[] = $tags[$i];
1464         } else {
1465             $curtag[] = $tags[$i];
1466         }
1467     }
1468     if (!empty($curtag)) {
1469         $newtags[] = $curtag;
1470     }
1471     foreach ($newtags as $tag) {
1472         foreach ($tag as $i => $t) {
1473             if ($t[1] === false) {
1474                 continue;
1475             }
1476             if (is_array($t[1])) {
1477                 $tag[$i][1][1]
1478                     = explode(" ", $t[1][1], str_replace("\t", ' ', $t[1][1]));
1479                 $x = $tag[$i][1][1];
1480             }
1481             $tagname = substr(array_shift($x), 1);
1482             $restoftag = $tag;
1483             if (isset($this-> tagHandlers[$tagname])) {
1484                 $handle = $this-> tagHandlers[$tagname];
1485             } else {
1486                 $handle = $this-> tagHandlers['*'];
1487             }
1488             $this-> $handle($tagname, $restoftag);
1489         }
1490     }
1491 }
1492 /**
1493 * This handler recognizes all {@}inline} tags
1494 *
1495 * Normal inline tags are simply highlighted. the {@}internal}} inline
1496 * tag {@tutorial tags.inlineinternal.pkg} is highlighted differently

```

```

1498 * to distinguish it from other inline tags.
1499 *
1500 * @param mixed $value      the tag value
1501 * @param bool  $endinternal indicates the end of an @internal tag
1502 *
1503 * @return void
1504 * @access private
1505 */
1506 function getInlineTags($value, $endinternal = false)
1507 {
1508     if (!$value) {
1509         return;
1510     }
1511     if ($this-> _pf_internal && !$endinternal) {
1512         if (strpos($value, '{}') !== false) {
1513             $x = strpos($value, '{}');
1514             // add the rest of internal
1515             $this-> getInlineTags(substr($value, 0, $x + 3), true);
1516             // strip internal from value
1517             $value = substr($value, strpos($value, '{}') + 1);
1518             // turn off internal
1519             $this-> _pf_internal = false;
1520         }
1521     }
1522     if (!$value) {
1523         return;
1524     }
1525     $dbtype = 'docblock';
1526     $dbtype .= ($this-> _pf_docblock ? '' : 'template');
1527     $save = $value;
1528     $value = explode('@', $value);
1529     $newval = array();
1530     // everything before the first @ is normal text
1531     $this-> _addDocBlockoutput($dbtype, $value[0]);
1532     for ($i=1; $i < count($value); $i++) {
1533         if (substr($value[$i], 0, 1) == '}') {
1534             $this-> _addDocBlockoutput($dbtype, '@' . substr($value[$i], 1));
1535         } else {
1536             $save = $value[$i];
1537             $value[$i] = str_replace("\t", "        ", $value[$i]);
1538             $word = array_shift($value[$i]);
1539             $val = join(' ', $value[$i]);
1540             if ($word == 'internal') {
1541                 $this-> _pf_internal = true;
1542                 $this-> _addDocBlockoutput($dbtype, '{internal ');
1543                 $value[$i] = substr($save, strlen('internal') + 1);
1544                 // strip internal and cycle as if it were normal text.
1545                 $this-> _addDocBlockoutput($dbtype, $value[$i]);
1546                 continue;
1547             }
1548             if (in_array(str_replace('{}', '', $word), $this-> allowableInlineTags))
1549             {
1550                 if (strpos($word, '{}')) {
1551                     $word = str_replace('{}', '', $word);
1552                     $val = '}' . $val;
1553                 }
1554                 $val = explode('{}', $val);
1555                 if (count($val) == 1) {
1556                     //addError(PDERROR_UNTERMINATED_INLINE_TAG,
1557                     //      $word, '', $save);
1558                 }
1559                 $rest = $val;
1560                 $val = array_shift($rest);
1561                 if ($endinternal) {
1562                     $rest = join('}', $rest);
1563                 } else {
1564                     $rest = join(' ', $rest);
1565                 }
1566                 if (isset($this-> inlineTagHandlers[$word])) {
1567                     $handle = $this-> inlineTagHandlers[$word];
1568                 } else {
1569                     $handle = $this-> inlineTagHandlers['*'];
1570                 }
1571                 $this-> $handle($word, $val);
1572                 $this-> _addDocBlockoutput($dbtype, $rest);
1573             } else {
1574                 $val = $word . ' ' . $val;
1575                 $this-> _addDocBlockoutput($dbtype, '@' . $val);
1576             }
1577         }

```

```

1578         }
1579     }
1580 }
1581
1582 /**
1583 * Handles all inline tags
1584 *
1585 * @param string $name the tag name
1586 * @param mixed $value the tag value
1587 *
1588 * @return void
1589 * @access private
1590 */
1591 function handleDefaultInlineTag($name, $value)
1592 {
    $this-> _addDocBlockoutput('inlinetag', '@' . $name . ' ' . $value . '');
}
1595
1596 /**
1597 * @+
1598 * phpDocumentor DocBlock tag handlers
1599 *
1600 * @param string $name tag name
1601 * @param array $value array of lines contained in the tag description
1602 *
1603 * @return void
1604 * @access private
1605 */
1606 /**
1607 * Handle normal tags
1608 *
1609 * This handler adds to output all comment information before the tag begins
1610 * as in " * " before "@todo" in " * @todo"
1611 *
1612 * Then, it highlights the tag as a regular or coretag based on $coretag.
1613 * Finally, it uses getInlineTags to highlight the description
1614 *
1615 * @param bool $coretag whether this tag is a core tag or not
1616 *
1617 * @uses getInlineTags() highlight a tag description
1618 */
1619 function defaultTagHandler($name, $value, $coretag = false)
1620 {
    $dbtype = 'docblock';
    $dbtype .= ($this-> _pf_docblock ? '' : 'template');
    foreach ($value as $line) {
        $this-> _addDocBlockoutput($dbtype, $line[0]);
        if ($line[1] === false) {
            if (trim($line[0]) != '*/') {
                $this-> newLineNum();
                $this-> _wp-> linenum++;
            }
            continue;
        }
        $this-> _addDocBlockoutput($dbtype, $line[1][0]);
        $stored = '';
        if (is_array($line[1][1])) {
            foreach ($line[1][1] as $i => $part) {
                if ($part == '@' . $name && $i == 0) {
                    $tagname = 'tag';
                    if ($coretag) {
                        $tagname = 'coretag';
                    }
                    $this-> _addDocBlockoutput($tagname, '@' . $name);
                    continue;
                }
                $stored .= ' ' . $part;
            }
        } else {
            $stored = $line[1];
        }
        $this-> getInlineTags($stored);
        if (strpos($stored, '*/') === false) {
            $this-> newLineNum();
            $this-> _wp-> linenum++;
        }
    }
}
1656 /**

```

```

1658 * main handler for "core" tags
1659 *
1660 * @see defaultTagHandler()
1661 */
1662 function coreTagHandler($name, $value)
1663 {
1664     return $this-> defaultTagHandler($name, $value, true);
1665 }
1666
1667 /**
1668 * Handles @global
1669 *
1670 * This handler works like {@link defaultTagHandler()} except it highlights
1671 * the type and variable (if present) in "@global type $variable" or
1672 * "@global type description"
1673 */
1674 function globalTagHandler($name, $value)
1675 {
1676     $this-> paramTagHandler($name, $value);
1677 }
1678
1679 /**
1680 * Handles @param
1681 *
1682 * This handler works like {@link defaultTagHandler()} except it highlights
1683 * the type and variable (if present) in "@param type $variable description"
1684 * or "@param type description"
1685 *
1686 * @param bool $checkforvar private parameter, checks for $var or not
1687 */
1688 function paramTagHandler($name, $value, $checkforvar = true)
1689 {
1690     $dbtype = 'docblock';
1691     $dbtype .= ($this-> _pf_docblock ? '' : 'template');
1692     $ret = $this-> retrieveType($value, 0, $checkforvar);
1693     foreach ($value as $num => $line) {
1694         $this-> _addDocBlockoutput($dbtype, $line[0]);
1695         if ($line[1] === false) {
1696             if (trim($line[0]) != '*/') {
1697                 $this-> newLineNum();
1698                 $this-> _wp-> linenum++;
1699             }
1700             continue;
1701         }
1702         $this-> _addDocBlockoutput($dbtype, $line[1][0]);
1703         $stored = '';
1704         $typeloc = 1;
1705         $varloc = 2;
1706         if (is_array($line[1][1])) {
1707             $this-> _addDocBlockoutput('coretag', '@' . $name . ' ');
1708             foreach ($ret[0] as $text) {
1709                 if (is_string($text)) {
1710                     $this-> _addDocBlockoutput($dbtype, $text);
1711                 }
1712                 if (is_array($text)) {
1713                     if ($text[0] != 'desc') {
1714                         $this-> _addDocBlockoutput($text[0], $text[1]);
1715                     } else {
1716                         $stored .= $text[1];
1717                     }
1718                 }
1719             }
1720         } else {
1721             if (isset($ret[$num])) {
1722                 foreach ($ret[$num] as $text) {
1723                     if (is_string($text)) {
1724                         $this-> _addDocBlockoutput($dbtype, $text);
1725                     }
1726                     if (is_array($text)) {
1727                         if ($text[0] != 'desc') {
1728                             $this-> _addDocBlockoutput($text[0], $text[1]);
1729                         } else {
1730                             $stored .= $text[1];
1731                         }
1732                     }
1733                 }
1734             } else {
1735                 $stored = $line[1];
1736             }
1737         }
}

```

```

1738         $this-> getInlineTags($stored);
1739         if (strpos($stored, '*/*') === false) {
1740             $this-> newLineNum();
1741             $this-> _wp-> linenum++;
1742         }
1743     }
1744 }
1745
1746 /**
1747 * handles the @staticvar tag
1748 *
1749 * @see paramTagHandler()
1750 */
1751 function staticvarTagHandler($name, $value)
1752 {
1753     return $this-> paramTagHandler($name, $value);
1754 }
1755
1756 /**
1757 * handles the @var tag
1758 *
1759 * @see paramTagHandler()
1760 */
1761 function varTagHandler($name, $value)
1762 {
1763     return $this-> paramTagHandler($name, $value);
1764 }
1765
1766 /**
1767 * Handles @return
1768 *
1769 * This handler works like {@link defaultTagHandler()} except it highlights
1770 * the type in "@return type description"
1771 */
1772 function returnTagHandler($name, $value)
1773 {
1774     $this-> paramTagHandler($name, $value, false);
1775 }
1776
1777 /**
1778 * Handles @property(-read or -write) and @method magic tags
1779 */
1780 function propertyTagHandler($name, $value)
1781 {
1782     return $this-> paramTagHandler($name, $value, true);
1783 }
1784
1785 /**#@-*/
1786
1787 /**
1788 * Retrieve the type portion of a @tag type description
1789 *
1790 * Tags like @param, @return and @var all have a PHP type portion in their
1791 * description. Since the type may contain the expression "object blah"
1792 * where blah is a classname, it makes parsing out the type field complex.
1793 *
1794 * Even more complicated is the case where a tag variable can contain
1795 * multiple types, such as object blah/object blah2/false, and so this
1796 * method handles these cases.
1797 *
1798 * @param array $value      array of words that were separated by spaces
1799 * @param 0/1 $state        0 = find the type, 1 = find the var, if present
1800 * @param bool $checkforvar flag to determine whether to check for the end of a
1801 *                           type is defined by a $varname
1802 *
1803 * @return array Format: array(state (0 [find type], 1 [var], 2 [done]),
1804 * @access private
1805 */
1806 function retrieveType($value, $state = 0, $checkforvar = false)
1807 {
1808     $index = 0;
1809     $result = array();
1810     do {
1811         if (!isset($value[$index][1])) {
1812             return $result;
1813         }
1814         $val = $value[$index][1];
1815         if (empty($val)) {
1816             return $result;
1817         }

```

```

1818     if ($index == 0) {
1819         $val = $val[1];
1820         array_shift($val);
1821     } else {
1822         $val = explode(' ', $val);
1823     }
1824     $ret           = $this-> _retrieveType($val, $state, $checkforvar);
1825     $state          = $ret[0];
1826     $result[$index++] = $ret[1];
1827 } while ((!$checkforvar && $state < 1) || ($state < 2 &&
1828 $checkforvar));
1829     return $result;
1830 }
1831 /**
1832 * used by {@link retrieveType()} in its work
1833 *
1834 * @param array $value      array of words that were separated by spaces
1835 * @param 0/1 $state        0 = find the type, 1 = find the var, if present
1836 * @param bool $checkforvar flag to determine whether to check for the end of a
1837 *                           type is defined by a $varname
1838 *
1839 * @return array
1840 * @access private
1841 */
1842 function _retrieveType($value, $state, $checkforvar)
1843 {
1844     $result = array();
1845     $result[] = $this-> _removeWhiteSpace($value, 0);
1846     if ($state == 0) {
1847         if (!count($value)) {
1848             return array(2, $result);
1849         }
1850         $types = '';
1851         $index = 0;
1852         if (trim($value[0]) == 'object') {
1853             $result[] = array('tagphptype', $value[0] . ' ');
1854             $types .= array_shift($value) . ' ';
1855             $result[] = $this-> _removeWhiteSpace($value, 0);
1856             if (!count($value)) {
1857                 // was just passed "object"
1858                 return array(2, $result);
1859             }
1860             if ($value[0][0] == '$' || substr($value[0], 0, 2) == '&$' ) {
1861                 // was just passed "object"
1862                 // and the next thing is a variable name
1863                 if ($checkforvar) {
1864                     $result[] = array('tagvarname', $value[0] . ' ');
1865                     array_shift($value);
1866                 }
1867                 $result[] = array('desc', join(' ', $value));
1868                 return array(2, $result);
1869             }
1870         }
1871         $done = false;
1872         $loop = -1;
1873         do {
1874             // this loop checks for type/type/type and for
1875             // type/object classname/type/object classname2
1876             if (strpos($value[0], '|')) {
1877                 $temptypes = explode('|', $value[0]);
1878                 while (count($temptypes)) {
1879                     $type = array_shift($temptypes);
1880                     $result[] = array('tagphptype', $type);
1881                     if (count($temptypes)) {
1882                         $result[] = '|';
1883                     }
1884                 }
1885                 if (trim($type) == 'object') {
1886                     $result[] = array('tagphptype', $types . ' ');
1887                     $result[] = $this-> _removeWhiteSpace($value, 0);
1888                 } else {
1889                     $done = true;
1890                 }
1891                 array_shift($value);
1892                 if (count($value) && strlen($value[0]) && isset($value[0])
1893 && ($value[0][0] == '$' || substr($value[0], 0, 2) == '&$' ))
1894             } {
1895                 // was just passed "object"

```

```

1896 // and the next thing is a variable name
1897 $result[] = array('tagvarname' , $value[0] . ' ');
1898 array_shift($value);
1899 $result[] = array('desc' , join(' ', $value));
1900 return array(2, $result);
1901 }
1902 } else {
1903     $result[] = array('tagphptype' , $value[0] . ' ');
1904     array_shift($value);
1905     $done = true;
1906 }
1907 $loop++;
1908 } while (!$done && count($value));
1909 if ($loop) {
1910     $result[] = ' ';
1911 }
1912 // still searching for type
1913 if (!$done && ! count($value)) {
1914     return array(0, $result);
1915 }
1916 // still searching for var
1917 if ($done && ! count($value)) {
1918     return array(1, $result);
1919 }
1920 }
1921 $result[] = $this-> _removeWhiteSpace($value, 0);
1922 $state = 1;
1923 if ($checkforvar) {
1924     if (count($value)) {
1925         $state = 2;
1926         if (substr($value[0], 0, 1) == '$' ||
1927             substr($value[0], 0, 2) == '&$')
1928     } {
1929         $result[] = array('tagvarname' , $value[0] . ' ');
1930         array_shift($value);
1931     }
1932     } else {
1933         $state = 1;
1934     }
1935 }
1936 $result[] = array('desc' , join(' ', $value));
1937 return array($state, $result);
1938 }
1939 /**
1940 * captures trailing whitespace
1941 *
1942 * @param array &$value      array of string
1943 * @param int   $index      index to seek non-whitespace to
1944 *
1945 * @return string whitespace
1946 * @access private
1947 */
1948 function _removeWhiteSpace(& $value, $index)
1949 {
1950     $result = '';
1951     if (count($value) > $index && empty($value[$index])) {
1952         $found = false;
1953         for ($i = $index; $i < count($value) && ! strlen($value[$i]); $i++) {
1954             $result .= ' ';
1955         }
1956         array_splice($value, $index, $i - $index);
1957     }
1958     return $result;
1959 }
1960 /**
1961 **#@+
1962 * Link generation methods
1963 *
1964 * @param string/array $word token to try to link
1965 *
1966 * @access private
1967 */
1968 /**
1969 * Generate a link to documentation for an element
1970 *
1971 * This method tries to link to documentation for functions, methods,
1972 * PHP functions, class names, and if found, adds the links to output
1973 * instead of plain text
1974 */
1975

```

```

1976     function _link($word)
1977     {
1978         if (is_array($word) && $word[0] == T_STRING) {
1979             if ($this-> _pf_colon_colon) {
1980                 $this-> _pf_colon_colon = false;
1981
1982                 $combo = $this-> _pv_last_string[1] . ':' . $word[1] . '()' ;
1983                 //debug('testing ' . $combo);
1984                 $link = $this-> _converter-> getLink($combo);
1985                 if (is_object($link)) {
1986                     $this-> _addoutput($this-> _converter-> returnSee($link,
1987                         $word[1]), true);
1988                     return;
1989                 }
1990                 $this-> _addoutput($word);
1991                 return;
1992             }
1993             $link = $this-> _converter-> getLink($word[1] . '()' );
1994             if (is_object($link)) {
1995                 $this-> _addoutput($this-> _converter-> returnSee($link,
1996                     $word[1]), true);
1997                 return;
1998             } elseif (is_string($link) && strpos($link, 'http://')) {
1999                 $this-> _addoutput($this-> _converter-> returnLink($link,
2000                     $word[1]), true);
2001                 return;
2002             } else {
2003                 $link = $this-> _converter-> getLink($word[1]);
2004                 if (is_object($link)) {
2005                     $word[1] = $this-> _converter-> returnSee($link, $word[1]);
2006                 }
2007                 $this-> _addoutput($word, true);
2008                 return;
2009             }
2010         }
2011         $this-> _addoutput($word);
2012     }
2013
2014 /**
2015 * Works like {@link _link()} except it only links to global variables
2016 */
2017 function _globallink($word)
2018 {
2019     if (!is_array($word)) {
2020         return $this-> _addoutput($word);
2021     }
2022     if ($word[0] != T_VARIABLE) {
2023         return $this-> _addoutput($word);
2024     }
2025     if (is_array($word) && $word[0] == T_VARIABLE) {
2026         $link = $this-> _converter-> getLink('global ' . $word[1]);
2027         if (is_object($link)) {
2028             $this-> _addoutput($this-> _converter-> returnSee($link,
2029                 $word[1]), true);
2030             return;
2031         }
2032     }
2033     $this-> _addoutput($word);
2034 }
2035
2036 /**
2037 * Works like {@link _link()} except it only links to classes
2038 */
2039 function _classlink($word)
2040 {
2041     //debug("checking class " . $word[1]);
2042     if (is_array($word) && $word[0] == T_STRING) {
2043         $link = $this-> _converter-> getLink($word[1]);
2044         if (is_object($link)) {
2045             $this-> _addoutput($this-> _converter-> returnSee($link,
2046                 $word[1]), true);
2047             return;
2048         }
2049     }
2050     $this-> _addoutput($word);
2051 }
2052
2053 /**
2054 * Works like {@link _link()} except it only links to methods
2055 */

```

```

2056     function _methodlink($word)
2057     {
2058         if (is_array($word) && $word[0] == T_STRING) {
2059             //debug("checking method " . $this->_pv_class . '::' . $word[1] .
2060             '()' );
2061             if (isset($this-> _pv_prev_var_type)) {
2062                 $link = $this-> _converter-> getLink($this-> _pv_prev_var_type . '::' .
2063                 $word[1] . '()' );
2064             } else {
2065                 $link = $this-> _converter-> getLink($this-> _pv_class . '::' .
2066                 $word[1] . '()' );
2067             }
2068             if (is_object($link)) {
2069                 $this-> _addoutput($this-> _converter-> returnSee($link,
2070                     $word[1]), true);
2071             return;
2072         }
2073         if (isset($this-> _pv_prev_var_type)) {
2074             $this-> _addoutput($word);
2075             return;
2076         }
2077         //debug("checking method " . $word[1] . '()' );
2078         $link = $this-> _converter-> getLink($word[1] . '()' );
2079         if (is_object($link)) {
2080             $this-> _addoutput($this-> _converter-> returnSee($link,
2081                 $word[1]), true);
2082             return;
2083         }
2084     $this-> _addoutput($word);
2085 }
2086
2087 /**
2088 * Works like {@link _link()} except it only links to class variables
2089 *
2090 * @param bool $justasString true if the $word is only a string
2091 */
2092 function _varlink($word, $justasString=false)
2093 {
2094     if ($justasString) {
2095         $word[0] = T_VARIABLE;
2096     }
2097     if (is_array($word) && $word[0] == T_VARIABLE) {
2098         $x = ($justasString ? '$' : '');
2099         //debug("checking var " . $this->_pv_class . '::' . $x . $word[1]);
2100         if (isset($this-> _pv_prev_var_type)) {
2101             //debug("checking var " . $this->_pv_prev_var_type . '::' .
2102             // $x . $word[1]);
2103             $link = $this-> _converter-> getLink($this-> _pv_prev_var_type . '::' .
2104             $x . $word[1]);
2105         } else {
2106             $link = $this-> _converter-> getLink($this-> _pv_class . '::' .
2107             $x . $word[1]);
2108         }
2109         if (is_object($link)) {
2110             $this-> _addoutput($this-> _converter-> returnSee($link,
2111                 $word[1]), true);
2112             return;
2113         }
2114         //debug("checking var " . $x . $word[1]);
2115         if (isset($this-> _pv_prev_var_type)) {
2116             $this-> _addoutput($word);
2117             return;
2118         }
2119         $link = $this-> _converter-> getLink($x . $word[1]);
2120         if (is_object($link)) {
2121             $this-> _addoutput($this-> _converter-> returnSee($link,
2122                 $word[1]), true);
2123             return;
2124         }
2125     }
2126     $this-> _addoutput($word);
2127 }
2128 /**
2129 */
2130 /**
2131 * Output Methods
2132 * @access private
2133 */
2134 /**

```

```

2135 * This method adds output to {@link $_line}
2136 *
2137 * If a string with variables like "$test this" is present, then special
2138 * handling is used to allow processing of the variable in context.
2139 *
2140 * @param mixed $word      the string/array tag token and value
2141 * @param bool  $preformatted whether or not the $word is already formatted
2142 *
2143 * @return void
2144 * @see _flush_save()
2145 */
2146 function _addoutput($word, $preformatted = false)
2147 {
2148     if ($this-> _pf_no_output_yet) {
2149         return;
2150     }
2151     if ($this-> _pf_quote_active) {
2152         if (is_array($word)) {
2153             $this-> _save .= $this-> _converter-> highlightSource($word[0],
2154                                         $word[1]);
2155         } else {
2156             $this-> _save .= $this-> _converter-> highlightSource(false,
2157                                         $word, true);
2158         }
2159     } else {
2160         $this-> _flush_save();
2161         if (is_string($word) && trim($word) == '') {
2162             $this-> _line .= $this-> _converter-> postProcess($word);
2163             return;
2164         }
2165         if (is_array($word) && trim($word[1]) == '') {
2166             $this-> _line .= $this-> _converter-> postProcess($word[1]);
2167             return;
2168         }
2169         if (is_array($word)) {
2170             $this-> _line .= $this-> _converter-> highlightSource($word[0],
2171                                         $word[1], $preformatted);
2172         } else {
2173             $this-> _line .= $this-> _converter-> highlightSource(false,
2174                                         $word, $preformatted);
2175         }
2176     }
2177 }
2178 /**
2179 * Like {@link _output()}, but for DocBlock highlighting
2180 *
2181 * @param mixed $dbtype      the docblock type
2182 * @param mixed $word        the string/array tag token and value
2183 * @param bool  $preformatted whether or not the $word is already formatted
2184 *
2185 * @return void
2186 */
2187 function _addDocBlockoutput($dbtype, $word, $preformatted = false)
2188 {
2189     if ($this-> _pf_internal) {
2190         $this-> _line .= $this-> _converter-> highlightDocBlockSource('internal',
2191                                         $word, $preformatted);
2192     } else {
2193         $this-> _line .= $this-> _converter-> highlightDocBlockSource($dbtype,
2194                                         $word, $preformatted);
2195     }
2196 }
2197 /**
2198 * Flush a saved string variable highlighting
2199 *
2200 * {@source}
2201 *
2202 * @return void
2203 * @todo CS cleanup - rename to _flushSave() for camelCase rule
2204 */
2205 function _flush_save()
2206 {
2207     if (!empty($this-> _save)) {
2208         $this-> _save .= $this-> _converter-> flushHighlightCache();
2209         // clear the existing cache, reset it to the old value
2210         if (isset($this-> _save_highlight_state)) {
2211             $this-> _converter->
2212                 _setHighlightCache($this-> _save_highlight_state[0],

```

```

2215             $this-> _save_highlight_state[1]);
2216         }
2217         $this-> _line .= $this-> _converter->
2218             highlightSource(T_CONSTANT_ENCAPSED_STRING, $this-> _save, true);
2219         $this-> _save = '';
2220     }
2221 }
2222 /**
2223 */
2224 /**
2225 * Give the word parser necessary data to begin a new parse
2226 *
2227 * @param array &$data      all tokens separated by line number
2228 *
2229 * @return void
2230 */
2231 function configWordParser(&    $data)
2232 {
2233     $this-> _wp-> setup($data, $this);
2234     $this-> _wp-> setWhitespace(true);
2235 }
2236
2237 /**
2238 * Initialize all parser state variables
2239 *
2240 * @param bool      $inlinesourceparse true if we are highlighting an inline
2241 *                   {@source} tag's output
2242 * @param false/string$class           name of class we are going
2243 *                                   to start from
2244 *
2245 * @return void
2246 * @uses $_wp sets to a new {@link phpDocumentor_HighlightWordParser}
2247 */
2248 function setupStates($inlinesourceparse, $class)
2249 {
2250     $this-> _output = '';
2251     $this-> _line = '';
2252     unset($this-> _wp);
2253     $this-> _wp = new phpDocumentor_HighlightWordParser;
2254     $this-> _event_stack = new EventStack;
2255     if ($inlinesourceparse) {
2256         $this-> _event_stack-> pushEvent(PARSER_EVENT_PHPCODE);
2257         if ($class) {
2258             $this-> _event_stack-> pushEvent(PARSER_EVENT_CLASS);
2259             $this-> _pv_class = $class;
2260         }
2261     } else {
2262         $this-> _pv_class = null;
2263     }
2264
2265     $this-> _pv_define = null;
2266     $this-> _pv_define_name = null;
2267     $this-> _pv_define_value = null;
2268     $this-> _pv_define_params_data = null;
2269     $this-> _pv_dtype = null;
2270     $this-> _pv_docblock = null;
2271     $this-> _pv_dtemplate = null;
2272     $this-> _pv_func = null;
2273     $this-> _pv_global_name = null;
2274     $this-> _pv_global_val = null;
2275     $this-> _pv_globals = null;
2276     $this-> _pv_global_count = null;
2277     $this-> _pv_include_params_data = null;
2278     $this-> _pv_include_name = null;
2279     $this-> _pv_include_value = null;
2280     $this-> _pv_linenum = null;
2281     $this-> _pv_periodline = null;
2282     $this-> _pv_paren_count = 0;
2283     $this-> _pv_statics = null;
2284     $this-> _pv_static_count = null;
2285     $this-> _pv_static_val = null;
2286     $this-> _pv_quote_data = null;
2287     $this-> _pv_function_data = null;
2288     $this-> _pv_var = null;
2289     $this-> _pv_varname = null;
2290     $this-> _pf_definename_isset = false;
2291     $this-> _pf_extends_found = false;
2292     $this-> _pf_includename_isset = false;
2293     $this-> _pf_get_source = false;
2294     $this-> _pf_getting_source = false;

```

```

2295     $this->_pf_in_class          = false;
2296     $this->_pf_in_define         = false;
2297     $this->_pf_in_global          = false;
2298     $this->_pf_in_include         = false;
2299     $this->_pf_in_var             = false;
2300     $this->_pf_funcparam_val      = false;
2301     $this->_pf_quote_active       = false;
2302     $this->_pf_reset_quote_data   = true;
2303     $this->_pf_useperiod          = false;
2304     $this->_pf_var_equals         = false;
2305     $this->_pf_obj_op             = false;
2306     $this->_pf_docblock           = false;
2307     $this->_pf_docblock_template  = false;
2308     $this->_pf_colon_colon        = false;
2309     $this->_pv_last_string        = false;
2310     $this->_pf_inmethod           = false;
2311     $this->_pf_no_output_yet      = false;
2312     $this->_pv_saveline           = 0;
2313     $this->_pv_next_word          = false;
2314     $this->_save                  = '';
2315 }
2316
2317 /**
2318 * Initialize the {@link $tokenpushEvent, $wordpushEvent} arrays
2319 *
2320 * @return void
2321 */
2322 function phpDocumentor_HighlightParser()
2323 {
2324     if (!defined('T_INTERFACE')) {
2325         define('T_INTERFACE', -1);
2326     }
2327     $this->allowableTags
2328         = $GLOBALS['_phpDocumentor_tags_allowed'];
2329     $this->allowableInlineTags
2330         = $GLOBALS['_phpDocumentor_inline_doc_tags_allowed'];
2331     $this->inlineTagHandlers
2332         = array('*' => 'handleDefaultInlineTag');
2333     //*****
2334     $this->tokenpushEvent[PARSER_EVENT_NOEVENTS] =
2335         array(
2336             T_OPEN_TAG => PARSER_EVENT_PHPCODE,
2337         );
2338     //*****
2339
2340     $this->tokenpushEvent[PARSER_EVENT_PHPCODE] =
2341         array(
2342             T_FUNCTION    => PARSER_EVENT_FUNCTION,
2343             T_CLASS        => PARSER_EVENT_CLASS,
2344             T_INTERFACE     => PARSER_EVENT_CLASS,
2345             T_INCLUDE_ONCE  => PARSER_EVENT_INCLUDE,
2346             T_INCLUDE       => PARSER_EVENT_INCLUDE,
2347             T_START_HEREDOC => PARSER_EVENT_EOQUOTE,
2348             T_REQUIRE        => PARSER_EVENT_INCLUDE,
2349             T_REQUIRE_ONCE   => PARSER_EVENT_INCLUDE,
2350             T_COMMENT        => PARSER_EVENT_COMMENT,
2351             T_DOC_COMMENT    => PARSER_EVENT_DOCBLOCK,
2352         );
2353     $this->wordpushEvent[PARSER_EVENT_PHPCODE] =
2354         array(
2355             "define"        => PARSER_EVENT_DEFINE,
2356             '"'            => PARSER_EVENT_QUOTE,
2357             '\''           => PARSER_EVENT_QUOTE,
2358         );
2359     //*****
2360
2361     $this->wordpushEvent[PARSER_EVENT_FUNCTION] =
2362         array(
2363             '{'  => PARSER_EVENT_LOGICBLOCK,
2364             '('  => PARSER_EVENT_FUNCTION_PARAMS,
2365         );
2366     $this->tokenpushEvent[PARSER_EVENT_FUNCTION] =
2367         array(
2368             T_COMMENT      => PARSER_EVENT_COMMENT,
2369             T_DOC_COMMENT  => PARSER_EVENT_DOCBLOCK,
2370         );
2371     $this->wordpopEvent[PARSER_EVENT_FUNCTION] = array("}");
2372     //*****
2373
2374

```

```

2375
2376     $this-> tokenpopEvent[PARSER_EVENT_EOFQUOTE] = array(T_END_HEREDOC);
2377     /***** **** */
2378
2379     $this-> tokenpushEvent[PARSER_EVENT_FUNCTION_PARAMS] =
2380     array(
2381         T_CONSTANT_ENCAPSED_STRING => PARSER_EVENT_QUOTE,
2382         T_ARRAY => PARSER_EVENT_ARRAY,
2383         T_COMMENT => PARSER_EVENT_COMMENT,
2384         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2385     );
2386     $this-> wordpushEvent[PARSER_EVENT_FUNCTION_PARAMS] =
2387     array(
2388         " " => PARSER_EVENT_QUOTE,
2389         " " => PARSER_EVENT_QUOTE,
2390     );
2391     $this-> wordpopEvent[PARSER_EVENT_FUNCTION_PARAMS] = array(" ");
2392     /***** **** */
2393
2394     $this-> wordpushEvent[PARSER_EVENT_LOGICBLOCK] =
2395     array(
2396         "{" => PARSER_EVENT_LOGICBLOCK,
2397         " " => PARSER_EVENT_QUOTE,
2398     );
2399     $this-> tokenpushEvent[PARSER_EVENT_LOGICBLOCK] =
2400     array(
2401         T_GLOBAL => PARSER_EVENT_FUNC_GLOBAL,
2402         T_STATIC => PARSER_EVENT_STATIC_VAR,
2403         T_START_HEREDOC => PARSER_EVENT_EOFQUOTE,
2404         T_CURLY_OPEN => PARSER_EVENT_LOGICBLOCK,
2405         T_DOLLAR_OPEN_CURLY_BRACES => PARSER_EVENT_LOGICBLOCK,
2406     );
2407     $this-> wordpopEvent[PARSER_EVENT_LOGICBLOCK] = array(" ");
2408     $this-> tokenpopEvent[PARSER_EVENT_LOGICBLOCK] = array(T_CURLY_OPEN);
2409
2410     /***** **** */
2411
2412     $this-> tokenpushEvent[PARSER_EVENT_ARRAY] =
2413     array(
2414         T_COMMENT => PARSER_EVENT_COMMENT,
2415         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2416     );
2417     $this-> wordpopEvent[PARSER_EVENT_ARRAY] = array(" ");
2418     /***** **** */
2419
2420     $this-> tokenpushEvent[PARSER_EVENT_FUNC_GLOBAL] =
2421     array(
2422         T_COMMENT => PARSER_EVENT_COMMENT,
2423         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2424     );
2425     $this-> wordpopEvent[PARSER_EVENT_FUNC_GLOBAL] = array(";");
2426     /***** **** */
2427
2428     $this-> tokenpushEvent[PARSER_EVENT_STATIC_VAR] =
2429     array(
2430         T_CONSTANT_ENCAPSED_STRING => PARSER_EVENT_QUOTE,
2431         T_COMMENT => PARSER_EVENT_COMMENT,
2432         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2433     );
2434     $this-> wordpushEvent[PARSER_EVENT_STATIC_VAR] =
2435     array(
2436         "=" => PARSER_EVENT_STATIC_VAR_VALUE,
2437     );
2438     $this-> wordpopEvent[PARSER_EVENT_STATIC_VAR] = array(";");
2439     /***** **** */
2440
2441     $this-> tokenpushEvent[PARSER_EVENT_STATIC_VAR_VALUE] =
2442     array(
2443         T_CONSTANT_ENCAPSED_STRING => PARSER_EVENT_QUOTE,
2444         T_COMMENT => PARSER_EVENT_COMMENT,
2445         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2446         T_ARRAY => PARSER_EVENT_ARRAY,
2447     );
2448     $this-> wordpushEvent[PARSER_EVENT_STATIC_VAR_VALUE] =
2449     array(
2450         " " => PARSER_EVENT_QUOTE,
2451         " " => PARSER_EVENT_QUOTE,
2452     );
2453     $this-> wordpopEvent[PARSER_EVENT_STATIC_VAR_VALUE] = array(";");
2454

```

```

2454 //*****
2455 $this-> tokenpushEvent[PARSER_EVENT_QUOTE] =
2456     array(
2457         T_OBJECT_OPERATOR => PARSER_EVENT_CLASS_MEMBER,
2458         T_CURLY_OPEN => PARSER_EVENT_QUOTE_VAR,
2459     );
2460 $this-> wordpopEvent[PARSER_EVENT_QUOTE] = array('' );
2461 //*****
2462 $this-> tokenpushEvent[PARSER_EVENT_QUOTE_VAR] =
2463     array(
2464         T_OBJECT_OPERATOR => PARSER_EVENT_CLASS_MEMBER,
2465         T_CURLY_OPEN => PARSER_EVENT_QUOTE_VAR,
2466     );
2467 $this-> wordpushEvent[PARSER_EVENT_QUOTE_VAR] =
2468     array(
2469         ''{'' => PARSER_EVENT_QUOTE_VAR,
2470         ''::'' => PARSER_EVENT_QUOTE_VAR,
2471         ''::'' => PARSER_EVENT_QUOTE_VAR,
2472     );
2473 $this-> wordpopEvent[PARSER_EVENT_QUOTE_VAR] = array('}' );
2474 //*****
2475 $this-> tokenpushEvent[PARSER_EVENT_DEFINE] =
2476     array(
2477         T_COMMENT => PARSER_EVENT_COMMENT,
2478         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2479         T_CONSTANT_ENCAPSED_STRING => PARSER_EVENT_QUOTE,
2480     );
2481 $this-> wordpushEvent[PARSER_EVENT_DEFINE] =
2482     array(
2483         ''( '' => PARSER_EVENT_DEFINE_PARAMS,
2484     );
2485 $this-> wordpopEvent[PARSER_EVENT_DEFINE] = array(';');
2486 //*****
2487 $this-> tokenpushEvent[PARSER_EVENT_DEFINE_PARAMS] =
2488     array(
2489         T_COMMENT => PARSER_EVENT_COMMENT,
2490         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2491     );
2492 $this-> wordpushEvent[PARSER_EVENT_DEFINE_PARAMS] =
2493     array(
2494         ''( '' => PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS,
2495         ''::'' => PARSER_EVENT_QUOTE,
2496         ''::'' => PARSER_EVENT_QUOTE,
2497     );
2498 $this-> wordpopEvent[PARSER_EVENT_DEFINE_PARAMS] = array("");
2499 //*****
2500 $this-> tokenpushEvent[PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS] =
2501     array(
2502         T_COMMENT => PARSER_EVENT_COMMENT,
2503         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2504     );
2505 $this-> wordpushEvent[PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS] =
2506     array(
2507         ''( '' => PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS,
2508         ''::'' => PARSER_EVENT_QUOTE,
2509         ''::'' => PARSER_EVENT_QUOTE,
2510     );
2511 $this-> wordpopEvent[PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS] = array("");
2512 //*****
2513 $this-> tokenpushEvent[PARSER_EVENT_VAR] =
2514     array(
2515         T_COMMENT => PARSER_EVENT_COMMENT,
2516         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2517         T_ARRAY => PARSER_EVENT_ARRAY,
2518     );
2519 $this-> wordpopEvent[PARSER_EVENT_VAR] = array(";");
2520 //*****
2521 $this-> tokenpushEvent[PARSER_EVENT_CLASS] =
2522     array(
2523         T_FUNCTION => PARSER_EVENT_METHOD,
2524         T_VAR => PARSER_EVENT_VAR,
2525         T_COMMENT => PARSER_EVENT_DOCBLOCK,
2526         T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,
2527         T_CLOSE_TAG => PARSER_EVENT_OUTPHP,
2528     );

```

```

2534     $this-> wordpopEvent[PARSER_EVENT_CLASS] = array("}");  

2535     /******  

2536  

2537     $this-> wordpushEvent[PARSER_EVENT_METHOD] =  

2538         array(  

2539             '{' => PARSER_EVENT_METHOD_LOGICBLOCK,  

2540             '(' => PARSER_EVENT_FUNCTION_PARAMS,  

2541         );  

2542     $this-> tokenpushEvent[PARSER_EVENT_METHOD] =  

2543         array(  

2544             T_COMMENT => PARSER_EVENT_COMMENT,  

2545             T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,  

2546         );  

2547     $this-> wordpopEvent[PARSER_EVENT_METHOD] = array("}",";");  

2548     /******  

2549  

2550     $this-> wordpushEvent[PARSER_EVENT_METHOD_LOGICBLOCK] =  

2551         array(  

2552             "{" => PARSER_EVENT_METHOD_LOGICBLOCK,  

2553             '"' => PARSER_EVENT_QUOTE,  

2554         );  

2555     $this-> tokenpushEvent[PARSER_EVENT_METHOD_LOGICBLOCK] =  

2556         array(  

2557             T_OBJECT_OPERATOR => PARSER_EVENT_CLASS_MEMBER,  

2558             T_GLOBAL => PARSER_EVENT_FUNC_GLOBAL,  

2559             T_STATIC => PARSER_EVENT_STATIC_VAR,  

2560             T_CURLY_OPEN => PARSER_EVENT_LOGICBLOCK,  

2561             T_DOLLAR_OPEN_CURLY_BRACES => PARSER_EVENT_LOGICBLOCK,  

2562         );  

2563     $this-> wordpopEvent[PARSER_EVENT_METHOD_LOGICBLOCK] = array("}");  

2564     $this-> tokenpopEvent[PARSER_EVENT_METHOD_LOGICBLOCK] = array(T_CURLY_OPEN);  

2565     /******  

2566  

2567     $this-> tokenpushEvent[PARSER_EVENT_INCLUDE] =  

2568         array(  

2569             T_COMMENT => PARSER_EVENT_COMMENT,  

2570             T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,  

2571         );  

2572     $this-> wordpushEvent[PARSER_EVENT_INCLUDE] =  

2573         array(  

2574             "(" => PARSER_EVENT_INCLUDE_PARAMS,  

2575         );  

2576     $this-> wordpopEvent[PARSER_EVENT_INCLUDE] = array(";" );  

2577     /******  

2578  

2579     $this-> tokenpushEvent[PARSER_EVENT_INCLUDE_PARAMS] =  

2580         array(  

2581             T_COMMENT => PARSER_EVENT_COMMENT,  

2582             T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,  

2583         );  

2584     $this-> wordpushEvent[PARSER_EVENT_INCLUDE_PARAMS] =  

2585         array(  

2586             "(" => PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS,  

2587         );  

2588     $this-> wordpopEvent[PARSER_EVENT_INCLUDE_PARAMS] = array(")");  

2589     /******  

2590  

2591     $this-> tokenpushEvent[PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS] =  

2592         array(  

2593             T_COMMENT => PARSER_EVENT_COMMENT,  

2594             T_DOC_COMMENT => PARSER_EVENT_DOCBLOCK,  

2595         );  

2596     $this-> wordpushEvent[PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS] =  

2597         array(  

2598             "(" => PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS,  

2599         );  

2600     $this-> wordpopEvent[PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS] =  

2601         );  

2602     }  

2603 ?>

```

# File Source for find\_phpdoc.php

Documentation for this file is available at [find\\_phpdoc.php](#)

```
1  <?php
2  /**
3  * Utility file to locate phpDocumentor for a non-PEAR installation
4  *
5  * phpDocumentor :: automatic documentation generator
6  *
7  * PHP versions 4 and 5
8  *
9  * Copyright (c) 2002-2007 Gregory Beaver
10 *
11 * LICENSE:
12 *
13 * This library is free software; you can redistribute it
14 * and/or modify it under the terms of the GNU Lesser General
15 * Public License as published by the Free Software Foundation;
16 * either version 2.1 of the License, or (at your option) any
17 * later version.
18 *
19 * This library is distributed in the hope that it will be useful,
20 * but WITHOUT ANY WARRANTY; without even the implied warranty of
21 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
22 * Lesser General Public License for more details.
23 *
24 * You should have received a copy of the GNU Lesser General Public
25 * License along with this library; if not, write to the Free Software
26 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
27 *
28 * @category ToolsAndUtilities
29 * @package phpDocumentor
30 * @subpackage setup
31 * @author Gregory Beaver <cellog@php.net>
32 * @copyright 2002-2007 Gregory Beaver
33 * @license http://www.opensource.org/licenses/lgpl-license.php LGPL
34 * @version CVS: $Id: find_phpdoc.php 246330 2007-11-17 03:09:41Z ashnazg $
35 * @filesource
36 * @link http://www.phpdoc.org
37 * @link http://pear.php.net/PhpDocumentor
38 * @since 1.2
39 * @todo CS cleanup - change package to PhpDocumentor
40 * @todo CS cleanup - change subpackage to Setup
41 */
42 /**
43 * Dummy value
44 * @internal CS Exception - logic here necessitates using an unconditional "include"
45 */
46 @include '';
47 // value used to test whether include worked
48 return 6;
49 ?>
```

# Appendix D - Todo List

## In Package Converters

### In [XMLDocBookpeardoc2Converter::endClass\(\)](#)

- move class summary into an array to be written out at the end of parsing each package

### In [CHMdefaultConverter::Output\(\)](#)

- use to directly call html help compiler hhc.exe

### In [PDFdefaultConverter](#)

- Implement links to conflicts/inheritance

### In [XMLDocBookConverter](#)

- indexes for other DocBook converters not based on peardoc
- templates

## In Package phpDocumentor

### In [abstractLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [ErrorTracker::addError\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature?

In [addError\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature?

In [addErrorDie\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature?

In [ErrorTracker::addErrorReturn\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature?

In [addWarning\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature?

In [ErrorTracker::addWarning\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature?

In [phpDocumentor\\_TutorialHighlightParser::checkEventPop\(\)](#)

- CS cleanup - PHPCS needs to recognize docblock template tags

In [phpDocumentor\\_TutorialHighlightParser::checkEventPush\(\)](#)

- CS cleanup - PHP-CS needs to recognize docblock template tags

#### In [Classes](#)

- CS cleanup - change package to PhpDocumentor

#### In [Classes.inc](#)

- CS cleanup - change package to PhpDocumentor

#### In [classLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [clone.inc.php](#)

- CS cleanup - change package to PhpDocumentor

#### In [clone5.inc.php](#)

- CS cleanup - change package to PhpDocumentor

#### In [common.inc.php](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename constant to TOKENIZER\_EXT

#### In [constLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserLinkInlineTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserTutorialTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserSamp::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserSeeTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserAccessTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserUsesTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserUsedByTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [parserIdInlineTag::Convert\(\)](#)

- CS cleanup - rename to convert for camelCase rule

In [parserInheritdocInlineTag::Convert\(\)](#)

- CS cleanup - rename to convert for camelCase rule

In [parserFileSourceTag::Convert\(\)](#)

- CS cleanup - rename to convert for camelCase rule

In [parserPre::Convert\(\)](#)

- CS cleanup - rename method to convert()

In [parserTocInlineTag::Convert\(\)](#)

- CS cleanup - rename to convert for camelCase rule

In [parserB::Convert\(\)](#)

- CS cleanup - rename method to convert()

In [parserReturnTag::Convert\(\)](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserCData::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserStringWithInlineTags::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserEntity::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserBr::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserCode::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserNameTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserList::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserKbd::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserTutorialInlineTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserDescVar::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserI::Convert\(\)\*\*](#)

- CS cleanup - rename method to convert()

In [\*\*parserSourceInlineTag::Convert\(\)\*\*](#)

- CS cleanup - rename to convert for camelCase rule

In [\*\*parserLinkInlineTag::ConvertPart\(\)\*\*](#)

- CS cleanup - rename to convertPart for camelCase rule

In [\*\*parserFileSourceTag::ConvertSource\(\)\*\*](#)

- CS cleanup - rename to convertSource for camelCase rule
- what's up with all the "return" statements? can they \_all\_ be removed?

#### In [parserExampleTag::ConvertSource\(\)](#)

- CS cleanup - rename to convertSource for camelCase rule
- what's up with all the "return" statements? can they \_all\_ be removed?

#### In [debug\(\)](#)

- CS Cleanup - can't avoid "prefixed by package" error

#### In [defineLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [DescHTML.inc](#)

- CS cleanup - change package to PhpDocumentor

#### In [DocBlockTags.inc](#)

- CS cleanup - change package to PhpDocumentor

#### In [Errors.inc](#)

- CS cleanup - change package to PhpDocumentor

#### In [ErrorTracker](#)

- CS cleanup - change package to PhpDocumentor

In [EventStack](#)

- CS cleanup - change package to PhpDocumentor

In [EventStack.inc](#)

- CS cleanup - change package to PhpDocumentor

In [find\\_phpdoc.php](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - change subpackage to Setup

In [functionLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [ProceduralPages::getPathInfo\(\)](#)

- figure out what &\$c is and update the param tag

In [parserPage::getSourceLocation\(\)](#)

- determine if the str\_replace in the 'pear/' ELSE branch should be removed (see Documentation/tests/bug1574043.php). It does NOT exist in the similar function parserClass->getSourceLocation() in ParserElements.inc.

## In [globalLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserTag::HandleEvent\(\)](#)

- CS cleanup - rename to handleEvent for camelCase rule

## In [HighlightParser.inc](#)

- CS cleanup - change package to PhpDocumentor

## In [Classes::Inherit\(\)](#)

- CS Cleanup - rename to "inherit" for CamelCaps naming standard

## In [InlineTags.inc](#)

- CS cleanup - change package to PhpDocumentor

## In [LinkClasses.inc](#)

- CS cleanup - change package to PhpDocumentor

## In [methodLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*

- CS cleanup - change package to PhpDocumentor

In [new.phpdoc.php](#)

- CS cleanup - change package to PhpDocumentor

In [PackagePageElements.inc](#)

- CS cleanup - change package to PhpDocumentor

In [pageLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [phpDocumentor\\_TutorialHighlightParser::parse\(\)](#)

- CS cleanup - unable to get function signature below 85char wide

In [phpDocumentor\\_HighlightParser::parse\(\)](#)

- CS cleanup - rename tokenizer\_ext constant to uppercase

In [Io::parseArgv\(\)](#)

- replace with Console\_\* ?

In [parserAccessTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*

- CS cleanup - change package to PhpDocumentor

In [\*\*parserB\*\*](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserB

In [\*\*parserBase\*\*](#)

- CS cleanup - change package to PhpDocumentor

In [\*\*parserBr\*\*](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserBr

In [\*\*parserCData\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserCode\*\*](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserCode

In [\*\*parserData\*\*](#)

- CS cleanup - change package to PhpDocumentor

In [ParserData.inc](#)

- CS cleanup - change package to PhpDocumentor

In [parserDescVar](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserDescVar

In [parserEntity](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [parserExampleInlineTag::parserExampleInlineTag\(\)](#)

- replace tokenizer\_ext constant with TOKENIZER\_EXT for CS rule

In [parserExampleInlineTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [parserExampleTag::parserExampleTag\(\)](#)

- CS cleanup - rename constant to TOKENIZER\_EXT
- does this "x = y = z = false" still work as expected in PHP5?

## In [parserExampleTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserFileSourceTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserI](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserI

## In [parserIdInlineTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserInheritdocInlineTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserInlineTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserKbd](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserKbd

## In [parserLicenseTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserLinkInlineTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserLinkTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

## In [parserList](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserList

## In [parserMethodTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*

- CS cleanup - change package to PhpDocumentor

#### In [parserNameTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [parserPage](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [parserParamTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [parserPre](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserPre

#### In [parserPropertyReadTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [parserPropertyTag](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserPropertyWriteTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserReturnTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserSamp\*\*](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename class to ParserSamp

In [\*\*parserSeeTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserSourceInlineTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserStaticvarTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserStringWithInlineTags\*\*](#)

- CS cleanup - change package to PhpDocumentor

In [\*\*parserTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserTocInlineTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserTutorialInlineTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserTutorialTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserUsedByTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserUsesTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserVarTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [\*\*parserXMLDocBookTag::parserXMLDocBookTag\(\)\*\*](#)

- CS cleanup - rename to parserXmlDocBookTag for camelCase rule

In [\*\*parserXMLDocBookTag\*\*](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor
- CS cleanup - rename to parserXmlDocBookTag for camelCase rule

In [\*\*PDERROR CLASS PARENT NOT FOUND\*\*](#)

- I think this description is a copy/paste that was never updated

In [\*\*PDERROR GLOBAL NOT FOUND\*\*](#)

- I think this description is a copy/paste that was never updated

In [PDERROR\\_MALFORMED\\_GLOBAL\\_TAG](#)

- I think this description is a copy/paste that was never updated

In [PDERROR\\_MULTIPLE\\_GLOBAL\\_TAGS](#)

- I think this description is a copy/paste that was never updated

In [PDERROR\\_NEED\\_WHITESPACE](#):

- I think this description is a copy/paste that was never updated

In [PDERROR\\_PACKAGEOUTPUT\\_DELETES\\_PARENT\\_FILE](#)

- I think this description is a copy/paste that was never updated

In [PDERROR\\_UNKNOWN\\_COMMANDLINE](#)

- I think this description is a copy/paste that was never updated

In [phpdoc.inc](#)

- CS cleanup - change package to PhpDocumentor

In [phpdoc.php](#)

- CS cleanup - change package to PhpDocumentor

#### In [phpDocumentorTParser](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [phpDocumentorTParser.inc](#)

- CS cleanup - change package to PhpDocumentor

#### In [phpDocumentorTWordParser](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [phpDocumentorTWordParser.inc](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - PHPCS needs to ignore CVS Id length

#### In [phpDocumentor\\_clone\(\)](#)

- CS cleanup - rename function to PhpDocumentor\_clone

#### In [phpDocumentor\\_ConfigFileList\(\)](#)

- CS cleanup - rename function to PhpDocumentor\_ConfigFileList

#### In [phpDocumentor\\_get\\_class\(\)](#)

- CS cleanup - rename function to PhpDocumentor\_get\_class

#### In [phpDocumentor\\_HighlightParser](#)

- CS cleanup - change class name to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [phpDocumentor\\_HighlightWordParser](#)

- CS cleanup - change class name to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [phpDocumentor\\_parse\\_ini\\_file\(\)](#)

- CS cleanup - rename function to PhpDocumentor\_parse\_ini\_file

#### In [phpDocumentor\\_TutorialHighlightParser](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

#### In [ProceduralPages](#)

- CS cleanup - change package to PhpDocumentor

#### In [ProceduralPages.inc](#)

- CS cleanup - change package to PhpDocumentor

## In [Publisher](#)

- CS cleanup - change package to PhpDocumentor

## In [Publisher.inc](#)

- CS cleanup - change package to PhpDocumentor

## In [RecordError](#)

- CS cleanup - change package to PhpDocumentor

## In [RecordWarning::RecordWarning\(\)](#)

- CS Cleanup - do I need to add \$data to the method signature? to sprintf based on the error number

## In [RecordWarning](#)

- CS cleanup - change package to PhpDocumentor

## In [phpDocumentor\\_setup::setMemoryLimit\(\)](#)

- recognize "K" and "G" in memory\_limit settings, rather than just "M"

## In [parserInlineTag::Strlen\(\)](#)

- CS cleanup - rename to strLen for camelCase rule

In [Publisher::subscribe\(\)](#)

- CS Cleanup - there's no way I can get the &\$object desc under 85 chars

In [TutorialHighlightParser.inc](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - PHPCS needs to ignore CVS Id length

In [tutorialLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [varLink](#)

- CS cleanup - change classname to PhpDocumentor\_\*
- CS cleanup - change package to PhpDocumentor

In [WordParser](#)

- CS cleanup - change package to PhpDocumentor

In [WordParser.inc](#)

- CS cleanup - change package to PhpDocumentor

In [XMPackagePageParser](#)

- CS cleanup - change package to PhpDocumentor

In [XMLpackagePageParser.inc](#)

- CS cleanup - change package to PhpDocumentor
- CS cleanup - PHPCS needs to ignore CVS Id length

## In Package tests

In [ParserClassGetSourceLocationTests.php](#)

- research possibility of refactoring ParserClass->getSourceLocation() and ParserPage->getSourceLocation() into a common method... also, there might be more occurrences of similar getSourceLocation() methods in other classes.

In [ParserPageGetSourceLocationTests.php](#)

- research possibility of refactoring ParserClass->getSourceLocation() and ParserPage->getSourceLocation() into a common method... also, there might be more occurrences of similar getSourceLocation() methods in other classes.

In  
[tests\\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeTrue\(\)](#):

- Revisit this test... I think it highlights a bug in the getSourceLocation method. Compare it with the same test in bug1574047.php against similar method parserClass->getSourceLocation().

In  
[tests\\_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeTrue\(\)](#):

- Revisit this test... I think it highlights a bug in the getSourceLocation method. Compare it with the same test in bug1574047.php against similar method parserClass->getSourceLocation().

In

[\*\*tests\\_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeTrue\(\)\*\*](#):

- Revisit this test... I think it highlights a bug in the getSourceLocation method. Compare it with the same test in bug1574047.php against similar method parserClass->getSourceLocation().

## In Package XML\_Beautifier

### [\*\*In PHPDoc XML Beautifier Renderer Plain\*\*](#)

- automatically create <![CDATA[ ]]> sections
- option to specify inline tags
- option to specify treatment of whitespace in data sections

# Index

<a href="#">phpDocumentor Manual</a>	47
<a href="#">phpDocumentor Tutorial</a>	16

## @

<a href="#">@see</a>	80
<a href="#">@since</a>	82
<a href="#">@return</a>	78
<a href="#">@property</a>	76
<a href="#">@package</a>	71
<a href="#">@param</a>	73
<a href="#">@static</a>	83
<a href="#">@staticvar</a>	85
<a href="#">@var</a>	94
<a href="#">@version</a>	95
<a href="#">@uses</a>	91
<a href="#">@tutorial</a>	89
<a href="#">@subpackage</a>	86
<a href="#">@todo</a>	88
<a href="#">@name</a>	70
<a href="#">@method</a>	69
<a href="#">@deprecated</a>	57
<a href="#">@example</a>	58
<a href="#">@copyright</a>	56
<a href="#">@category</a>	55
<a href="#">@access</a>	52
<a href="#">@author</a>	54
<a href="#">@final</a>	60
<a href="#">@filesource</a>	61
<a href="#">@license</a>	67
<a href="#">@link</a>	68
<a href="#">@internal</a>	66
<a href="#">@ignore</a>	65
<a href="#">@global</a>	62
<a href="#">@abstract</a>	51

## A

<a href="#">abstractLink::\$name</a>	789
<a href="#">abstractLink::\$fileAlias</a>	789
<i>phpdoc alias _phpdoc_inc for phpdoc.inc</i>	
<a href="#">abstractLink::\$category</a>	789

<a href="#">abstractLink</a>	linking classes parent	789
<a href="#">abstractLink::\$package</a>		789
<a href="#">abstractLink::\$path</a>		789
<a href="#">abstractLink::addLink()</a>	sets up the link	790
<a href="#">abstractLink::\$type</a>	element type linked to.	789
<a href="#">abstractLink::\$subpackage</a>		789
<a href="#">addWarning()</a>	add a Warning	757
<a href="#">addErrorDie()</a>	like <a href="#">addError()</a> but exits parsing	756
<a href="#">a</a>	tests linking to methods and vars	530
<a href="#">aa_567059()</a>	test links to methods and vars	518
<a href="#">a()</a>	test links to methods and vars	517
<a href="#">a::\$a</a>		530
<a href="#">a::\$c</a>		530
<a href="#">addError()</a>	add an Error	756
<a href="#">actions.php</a>	phpDocumentor :: docBuilder Web Interface	619
<a href="#">a::b()</a>		530
<a href="#">adv_htmlentities()</a>	smart htmlentities, doesn't entity the allowed tags list	144

## B

<a href="#">brokenlinkstovars</a>		531
<a href="#">baby::\$oopsieindexing</a>		531
<a href="#">baby</a>		531
<a href="#">brokenlinkstovars::\$broken</a>		531
<a href="#">bug540341</a>		531
<a href="#">bug557861</a>	this page will not be shown in the package list and should be	532
<a href="#">bug540341::get_header2()</a>		532
<a href="#">b553607 Parser</a>		530
<a href="#">bug-shortdesc.php</a>		529
<a href="#">bug-loseprocedural.php</a>		526
<a href="#">bug-escaping.php</a>		525
<a href="#">bug-eofquotes.php</a>		524
<a href="#">bug-pageleveldocsblocks.php</a>	Im a page level docblock	527
<a href="#">bug-quote_new_parser.php</a>		528
<a href="#">bqnp_testie()</a>	The tokenizer splits up strings that have inline variables	528
<a href="#">bqnpTester</a>	The tokenizer splits up strings that have inline variables	528
<a href="#">bug_489398</a>		532

<a href="#">bug_489398::\$test_01</a>	. . . . .	532
	<i>Checking the single quote var case</i>	
<a href="#">builder.php</a>	. . . . .	621
	<i>phpDocumentor :: docBuilder Web Interface</i>	
<a href="#">bug_556894_sub2</a>	. . . . .	614
	<i>Subclass in different subpackage</i>	
<a href="#">bug_556894_sub1</a>	. . . . .	613
	<i>Subclass in same subpackage</i>	
<a href="#">bug_772441</a>	. . . . .	661
<a href="#">Beautifier.php</a>	. . . . .	863
	<i>XML/Beautifier.php</i>	
<a href="#">Beautifier.php</a>	. . . . .	1311
	<i>Source code</i>	
<a href="#">builder.php</a>	. . . . .	1169
	<i>Source code</i>	
<a href="#">bug_556894_base::test()</a>	. . . . .	613
	<i>I'm a test method</i>	
<a href="#">bug_556894_base::\$test</a>	. . . . .	613
	<i>I'm a test var</i>	
<a href="#">bug_489398::\$test_03</a>	. . . . .	533
	<i>Checking the no quote cause</i>	
<a href="#">bug_489398::\$test_02</a>	. . . . .	533
	<i>checking the double quote var case</i>	
<a href="#">bug_489398::\$test_04</a>	. . . . .	533
	<i>Checking the empty array case</i>	
<a href="#">bug_489398::\$test_05</a>	. . . . .	533
	<i>Checking the array with data case</i>	
<a href="#">bug_556894_base</a>	. . . . .	613
	<i>Base Class</i>	
<a href="#">bug-541886.php</a>	. . . . .	554
	<i>Test for bug #541886</i>	
<a href="#">bug-defineparse.php</a>	. . . . .	523
<a href="#">bug-authoremail.php</a>	. . . . .	522
<a href="#">bug-540368.php</a>	. . . . .	496
<a href="#">bug-445820.php</a>	. . . . .	495
<a href="#">bug-445305.php</a>	. . . . .	494
<a href="#">blah()</a>	. . . . .	496
<a href="#">bug-542586.php</a>	. . . . .	497
<a href="#">bug-551120.php</a>	. . . . .	499
<a href="#">bug-550489.php</a>	. . . . .	498
<a href="#">bug-445298.php</a>	. . . . .	493
<a href="#">bug-443153.php</a>	. . . . .	492
<a href="#">bug-441275.php</a>	. . . . .	486
<a href="#">block.textformat.php</a>	. . . . .	428
	<i>Smarty plugin</i>	
<a href="#">block.strip.php</a>	. . . . .	427
	<i>Smarty plugin</i>	
<a href="#">bug-441278.php</a>	. . . . .	487
<a href="#">bug-441287.php</a>	. . . . .	488
<a href="#">bug-441433.php</a>	. . . . .	491
<a href="#">bug-441289.php</a>	. . . . .	489
<a href="#">bug-553137.php</a>	. . . . .	500
<a href="#">bug-554712.php</a>	. . . . .	501

<a href="#">bug-566600.php</a>	517
<a href="#">bug-566200.php</a>	516
<a href="#">bug-562997.php</a>	515
<i>This page returns a class with name "%s"\n", and shouldn't find class at all</i>	
<a href="#">bug-567059.php</a>	518
<a href="#">bug-645588.php</a>	519
<i>Global functions</i>	
<a href="#">bug698356_Output()</a>	521
<i>Create the phpdoc.hhp, contents.hhc files needed by MS HTML Help Compiler to create a CHM file</i>	
<a href="#">bug-698356.php</a>	521
<i>This is a test of bug 698356. Must be parsed with -pp on to test</i>	
<a href="#">bug-560595.php</a>	514
<a href="#">bug-560578.php</a>	513
<i>page-level stuff</i>	
<a href="#">bug-558031.php</a>	503
<a href="#">bug-557861.php</a>	502
<i>bug is fixed. to test, remove this page-level docblock</i>	
<a href="#">bug-558051.php</a>	504
<a href="#">bug-559467.php</a>	505
<a href="#">bug-559668.php</a>	507
<a href="#">bug-559494.php</a>	506
<i>tests variable names with the word 'array' in them</i>	
<a href="#">bug-904820.php</a>	365
<i>This is a test of a package with a . in its name</i>	

## C

<a href="#">Cpdf::o_destination()</a>	341
<i>destination object, used to specify the location for the user to jump to, presently on opening</i>	
<a href="#">Cpdf::o_encryption()</a>	342
<i>encryption object.</i>	
<a href="#">Cpdf::o_font()</a>	342
<i>an object to hold the font description</i>	
<a href="#">Cpdf::o_contents()</a>	341
<i>the contents objects hold all of the content which appears on pages</i>	
<a href="#">Cpdf::o_catalog()</a>	341
<i>define the document catalog, the overall controller for the document</i>	
<a href="#">Cpdf::o_action()</a>	340
<i>an action object, used to link to URLs initially</i>	
<a href="#">Cpdf::o_annotation()</a>	341
<i>an annotation object, this will add an annotation to the current page.</i>	
<a href="#">Cpdf::o_fontDescriptor()</a>	342
<i>a font descriptor, needed for including additional fonts</i>	
<a href="#">Cpdf::o_fontEncoding()</a>	342
<i>the font encoding</i>	
<a href="#">Cpdf::o_pages()</a>	344
<i>object which is a parent to the pages in the document</i>	
<a href="#">Cpdf::o_procset()</a>	344
<i>the document procset, solves some problems with printing to old PS printers</i>	

<a href="#">Cpdf::o_page()</a>	343
<i>a page object, it also creates a contents object to hold its contents</i>	
<a href="#">Cpdf::o_outlines()</a>	343
<i>define the outlines in the doc, empty for now</i>	
<a href="#">Cpdf::o_image()</a>	343
<i>an image object, will be an XObject in the document, includes description and data</i>	
<a href="#">Cpdf::o_info()</a>	343
<i>define the document information</i>	
<a href="#">Cpdf::output()</a>	340
<i>return the pdf stream as a string returned from the function</i>	
<a href="#">Cpdf::openObject()</a>	340
<i>make a loose object, the output will go into this object, until it is closed, then will revert to the current one.</i>	
<a href="#">Cpdf::encryptInit()</a>	337
<i>initialize the encryption for processing a particular object</i>	
<a href="#">Cpdf::filledEllipse()</a>	338
<i>draw a filled ellipse</i>	
<a href="#">Cpdf::filledRectangle()</a>	338
<i>a filled rectangle, note that it is the width and height of the rectangle which are the secondary parameters, not</i>	
<a href="#">Cpdf::ellipse()</a>	337
<i>draw an ellipse</i>	
<i>note that the part and filled ellipse are just special cases of this function</i>	
<a href="#">Cpdf::curve()</a>	336
<i>draw a bezier curve based on 4 control points</i>	
<a href="#">Cpdf::checkAllHere()</a>	336
<i>should be used for internal checks, not implemented as yet</i>	
<a href="#">Cpdf::closeObject()</a>	336
<i>close an object</i>	
<a href="#">Cpdf::getFirstPageId()</a>	338
<i>function for the user to find out what the ID is of the first page that was created during startup - useful if they wish to add something to it later.</i>	
<a href="#">Cpdf::getFontDecender()</a>	338
<i>return the font decender, this will normally return a negative number</i>	
<a href="#">Cpdf::newPage()</a>	339
<i>add a new page to the document</i>	
<a href="#">Cpdf::openHere()</a>	340
<i>specify where the document should open when it first starts</i>	
<a href="#">Cpdf::md5_16()</a>	339
<i>calculate the 16 byte version of the 128 bit md5 digest of the string</i>	
<a href="#">Cpdf::line()</a>	339
<i>draw a line from one set of coordinates to another</i>	
<a href="#">Cpdf::getFontHeight()</a>	339
<i>return the height in units of the current font in the given size</i>	
<a href="#">Cpdf::getTextWidth()</a>	339
<i>calculate how wide a given text string will be on a page, at a given size.</i>	
<a href="#">Cpdf::o_viewerPreferences()</a>	344
<i>set the viewer preferences</i>	
<a href="#">Cpdf::partEllipse()</a>	344
<i>draw a part of an ellipse</i>	
<a href="#">constructor HTML_TreeNode::HTML_TreeNode()</a>	361
<i>Constructor</i>	
<a href="#">Config_File.class.php</a>	367

<i>Config_File</i>	371
<i>Config file reading class</i>	
<b>constructor</b> <i>HTML_TreeMenu_Presentation::HTML_TreeMenu_Presentation()</i>	359
<i>Base constructor simply sets the menu object</i>	
<b>constructor</b> <i>HTML_TreeMenu_Listbox::HTML_TreeMenu_Listbox()</i>	358
<i>Constructor</i>	
<b>constructor</b> <i>HTML_TreeMenu::HTML_TreeMenu()</i>	354
<i>Constructor</i>	
<b>constructor</b> <i>HTML_TreeMenu_DHTML::HTML_TreeMenu_DHTML()</i>	356
<i>Constructor, takes the tree structure as</i>	
<b>Config_File::\$booleanize</b>	371
<i>Controls whether config values of on/true/yes and off/false/no get converted to boolean values automatically.</i>	
<b>Config_File::\$fix_newlines</b>	371
<i>Controls whether or not to fix mac or dos formatted newlines.</i>	
<b>Config_File::clear()</b>	372
<i>Clear loaded config data for a certain file or all files.</i>	
<b>Config_File::get()</b>	372
<i>Retrieves config info based on the file, section, and variable name.</i>	
<b>constructor Config_File::Config_File()</b>	372
<i>Constructs a new config file class.</i>	
<b>Config_File::\$ config_data</b>	372
<b>Config_File::\$overwrite</b>	371
<i>Controls whether variables with the same name overwrite each other.</i>	
<b>Config_File::\$read_hidden</b>	372
<i>Controls whether hidden config sections/vars are read from the file.</i>	
<b>constructor DirNode::DirNode()</b>	353
<b>Cpdf::transaction()</b>	349
<i>a few functions which should allow the document to be treated transactionally.</i>	
<b>Cpdf::saveState()</b>	346
<i>this will be called at a new page to return the state to what it was on the</i>	
<b>Cpdf::selectFont()</b>	346
<i>if the font is not loaded then load it and make the required object</i>	
<b>Cpdf::restoreState()</b>	345
<i>restore a previously saved state</i>	
<b>Cpdf::reopenObject()</b>	345
<i>open an existing object for editing</i>	
<b>Cpdf::polygon()</b>	345
<i>draw a polygon, the syntax for this is similar to the GD polygon command</i>	
<b>Cpdf::rectangle()</b>	345
<i>draw a rectangle, note that it is the width and height of the rectangle which are the secondary paramaters, not</i>	
<b>Cpdf::setColor()</b>	346
<i>sets the colour for fill operations</i>	
<b>Cpdf::setEncryption()</b>	347
<i>set the encryption of the document</i>	
<i>can be used to turn it on and/or set the passwords which it will have.</i>	
<b>Cpdf::stopObject()</b>	348
<i>stop an object from appearing on pages from this point on</i>	
<b>Cpdf::stream()</b>	348
<i>output the pdf code, streaming it to the browser</i>	
<b>Cpdf::setStrokeColor()</b>	348

<i>sets the colour for stroke operations</i>	
<a href="#">Cpdf::setPreferences()</a>	348
<i>set the viewer preferences of the document, it is up to the browser to obey these.</i>	
<a href="#">Cpdf::setFontFamily()</a>	347
<i>define font families, this is used to initialize the font families for the default fonts and for the user to add new ones for their fonts. The default bahaviour can be overridden should that be desired.</i>	
<a href="#">Cpdf::setLineStyle()</a>	347
<i>this sets the line drawing style.</i>	
<a href="#">Cpdf::ARC4_init()</a>	336
<i>initialize the ARC4 encryption</i>	
<a href="#">Cpdf::ARC4()</a>	336
<i>ARC4 encrypt a text string</i>	
<a href="#">Cpdf::\$currentColour</a>	328
<i>current colour for fill operations, defaults to inactive value, all three components should be between 0 and 1 inclusive when active</i>	
<a href="#">Cpdf::\$currentContents</a>	328
<i>object number of the currently active contents block</i>	
<a href="#">Cpdf::\$currentFont</a>	328
<i>a record of the current font</i>	
<a href="#">Cpdf::\$currentBaseFont</a>	328
<i>the current base font</i>	
<a href="#">Cpdf::\$checkpoint</a>	327
<i>store the stack for the transaction commands, each item in here is a record of the values of all the variables within the class, so that the user can rollback at will (from each 'start' command) note that this includes the objects array, so these can be large.</i>	
<a href="#">Cpdf::\$callback</a>	327
<i>array which forms a stack to keep track of nested callback functions</i>	
<a href="#">Cpdf::\$catalogId</a>	327
<i>the objectid (number within the objects array) of the document catalog</i>	
<a href="#">Cpdf::\$currentFontNum</a>	328
<i>the number of the current font within the font array</i>	
<a href="#">Cpdf::\$currentLineStyle</a>	328
<i>current style that lines are drawn in</i>	
<a href="#">Cpdf::\$destinations</a>	329
<i>store label-&gt;id pairs for named destinations, these will be used to replace internal links</i>	
<a href="#">Cpdf::\$encrypted</a>	329
<i>a flag to say if a document is to be encrypted or not</i>	
<a href="#">Cpdf::\$currentTextState</a>	329
<i>track if the current font is bolded or italicised</i>	
<a href="#">Cpdf::\$currentStrokeColour</a>	329
<i>current colour for stroke operations (lines etc.)</i>	
<a href="#">Cpdf::\$currentNode</a>	328
<a href="#">Cpdf::\$currentPage</a>	329
<i>object number of the current page</i>	
<a href="#">Cpdf::\$arc4_objnum</a>	327
<i>the object Id of the encryption information</i>	
<a href="#">Cpdf::\$arc4</a>	327
<i>the encryption array for the document encryption is stored here</i>	
<a href="#">CeZpdf::ezStartPageNumbers()</a>	325
<a href="#">CeZpdf::ezStopPageNumbers()</a>	325

<a href="#">Cezpdf::ezStream()</a>	325
<a href="#">Cezpdf::ezSetY()</a>	324
<a href="#">Cezpdf::ezSetMargins()</a>	324
<a href="#">Cezpdf::ezSetCmMargins()</a>	324
<a href="#">Cezpdf::ezSetDy()</a>	324
<a href="#">Cezpdf::ezTable()</a>	325
<a href="#">Cezpdf::ezText()</a>	325
<a href="#">Cpdf</a>	326
<a href="#">Cpdf</a>	
<a href="#">Cpdf::\$addLooseObjects</a>	327
<i>array contains information about how the loose objects are to be added to the document</i>	
<a href="#">Cezpdf::underline()</a>	326
<a href="#">Cezpdf::loadTemplate()</a>	326
<a href="#">Cezpdf::ezWhatPageNumber()</a>	326
<a href="#">Cezpdf::ilink()</a>	326
<a href="#">Cpdf::\$encryptionKey</a>	329
<i>the encryption key for the encryption of all the document content (structure is not encrypted)</i>	
<a href="#">Cpdf::\$fileIdentifier</a>	329
<i>the file identifier, used to uniquely identify a pdf document</i>	
<a href="#">Cpdf::addDestination()</a>	333
<i>create a labelled destination within the document</i>	
<a href="#">Cpdf::addImage()</a>	333
<i>add an image into the document, from a GD object</i>	
<a href="#">Cpdf::addInfo()</a>	333
<i>add content to the documents info object</i>	
<a href="#">constructor Cpdf::Cpdf()</a>	332
<i>class constructor</i>	
<a href="#">Cpdf::\$wordSpaceAdjust</a>	332
<i>used to track the last used value of the inter-word spacing, this is so that it is known when the spacing is changed.</i>	
<a href="#">Cpdf::\$stack</a>	332
<i>object Id storage stack</i>	
<a href="#">Cpdf::\$stateStack</a>	332
<i>an array which is used to save the state of the document, mainly the colours and styles</i>	
<a href="#">Cpdf::addInternalLink()</a>	334
<i>add a link in the document to an internal destination (ie. within the document)</i>	
<a href="#">Cpdf::addJpegFromFile()</a>	334
<i>add a JPEG image into the document, from a file</i>	
<a href="#">Cpdf::addText()</a>	335
<i>add text to the document, at a specified location, size and angle on the page</i>	
<a href="#">Cpdf::addTextWrap()</a>	335
<i>add text to the page, but ensure that it fits within a certain width if it does not fit then put in as much as possible, splitting at word boundaries and return the remainder.</i>	
<a href="#">Cpdf::addPngFromFile()</a>	335
<i>add a PNG image into the document, from a file</i>	
<a href="#">Cpdf::addObject()</a>	335
<i>after an object has been created, it will only show if it has been added, using this function.</i>	
<a href="#">Cpdf::addLink()</a>	334
<i>add a link in the document to an external URL</i>	
<a href="#">Cpdf::addMessage()</a>	334
<i>used to add messages for use in debugging</i>	

<a href="#">Cpdf::\$procsetId</a>	332
<i>the object Id of the procset object</i>	
<a href="#">Cpdf::\$options</a>	332
<i>an array containing options about the document</i>	
<a href="#">Cpdf::\$looseObjects</a>	330
<i>an array which contains information about the objects which are not firmly attached to pages</i>	
<a href="#">Cpdf::\$messages</a>	330
<i>messages are stored here during processing, these can be selected afterwards to give some useful debug information</i>	
<a href="#">Cpdf::\$infoObject</a>	330
<i>the objectid of the information object for the document this contains authorship, title etc.</i>	
<a href="#">Cpdf::\$fonts</a>	330
<i>array carrying information about the fonts that the system currently knows about</i>	
<a href="#">Cpdf::\$firstPageId</a>	330
<i>the objectid of the first page of the document</i>	
<a href="#">Cpdf::\$fontFamilies</a>	330
<i>store the information about the relationship between font families this used so that the code knows which font is the bold version of another font, etc.</i>	
<a href="#">Cpdf::\$nCallback</a>	330
<i>the number of callback functions in the callback array</i>	
<a href="#">Cpdf::\$nStack</a>	331
<i>number of elements within the object Id storage stack</i>	
<a href="#">Cpdf::\$numPages</a>	331
<i>number of page objects within the document</i>	
<a href="#">Cpdf::\$objects</a>	331
<i>this array contains all of the pdf objects, ready for final assembly</i>	
<a href="#">Cpdf::\$numObj</a>	331
<i>the current number of pdf objects in the document</i>	
<a href="#">Cpdf::\$numImages</a>	331
<i>number of images being tracked within the document</i>	
<a href="#">Cpdf::\$nStateStack</a>	331
<i>number of elements within the state stack</i>	
<a href="#">Cpdf::\$numFonts</a>	331
<i>number of fonts within the system</i>	
<a href="#">Config_File::get_file_names()</a>	373
<i>Get all loaded config file names.</i>	
<a href="#">Config_File::get_key()</a>	373
<i>Retrieves config info based on the key.</i>	
<a href="#">constructor parserSeeTag::parserSeeTag()</a>	741
<i>sets up the see tag</i>	
<a href="#">constructor parserTag::parserTag()</a>	743
<i>Set up the tag</i>	
<a href="#">constructor parserUsedByTag::parserUsedByTag()</a>	746
<i>set up the usedby tag</i>	
<a href="#">constructor parserReturnTag::parserReturnTag()</a>	740
<i>set up the tag</i>	
<a href="#">constructor parserPropertyTag::parserPropertyTag()</a>	737
<i>set up the property tag</i>	
<a href="#">constructor parserLinkTag::parserLinkTag()</a>	733
<i>sets up the link tag</i>	
<a href="#">constructor parserNameTag::parserNameTag()</a>	735

<i>set up the name tag</i>	
<a href="#">constructor parserUsesTag::parserUsesTag()</a>	747
<i>    set up the uses tag</i>	
<a href="#">constructor RecordWarning::RecordWarning()</a>	771
<i>        Constructor</i>	
<a href="#">constructor parserLinkInlineTag::parserLinkInlineTag()</a>	780
<i>         sets up the tag</i>	
<a href="#">constructor parserSourceInlineTag::parserSourceInlineTag()</a>	783
<i>         constructor</i>	
<a href="#">constructor parserInlineTag::parserInlineTag()</a>	779
<i>         sets up the tag</i>	
<a href="#">constructor parserInheritdocInlineTag::parserInheritdocInlineTag()</a>	778
<i>         Does nothing, overrides parent constructor</i>	
<a href="#">constructor parserExampleInlineTag::parserExampleInlineTag()</a>	774
<i>         constructor</i>	
<a href="#">constructor parserIdInlineTag::parserIdInlineTag()</a>	776
<i>         constructor</i>	
<a href="#">constructor parserLicenseTag::parserLicenseTag()</a>	732
<i>         set up the license tag</i>	
<a href="#">constructor parserFileSourceTag::parserFileSourceTag()</a>	730
<i>         Set \$source to \$value, and set up path</i>	
<a href="#">Classes::getRoots()</a>	665
<i>         Get a list of all root classes indexed by package. Used to generate class trees by Converter</i>	
<a href="#">Classes::Inherit()</a>	665
<i>         Main processing engine for setting up class inheritance.</i>	
<a href="#">Classes::nextFile()</a>	666
<i>         Prepare to parse a new file</i>	
<a href="#">Classes::getParentClass()</a>	665
<i>         Find the parent class of a class in file \$file</i>	
<a href="#">Classes::getDefiniteChildren()</a>	664
<i>         Get all classes confirmed in parsing to be descended class \$parclass in file \$file</i>	
<a href="#">Classes::getClassesInPath()</a>	664
<i>         Used by parserData::getClasses() to retrieve classes defined in file \$path</i>	
<a href="#">Classes::getConflicts()</a>	664
<i>         If a package contains two classes with the same name, this function finds that conflict</i>	
<a href="#">Classes::processChild()</a>	666
<i>         This function recursively climbs up the class tree, setting inherited information like package and adds the elements to phpDocumentor_IntermediateParser.</i>	
<a href="#">Classes::setClassParent()</a>	667
<i>         Find the parent class of \$class, and set up structures to note this fact</i>	
<a href="#">constructor parserAccessTag::parserAccessTag()</a>	726
<i>         checks \$value to make sure it is private, protected or public, otherwise it's not a valid @access tag</i>	
<a href="#">constructor parserExampleTag::parserExampleTag()</a>	728
<i>         Reads and parses the example file indicated</i>	
<a href="#">constructor parserList::parserList()</a>	722
<i>         Constructor - create a new list</i>	
<a href="#">constructor phpDocumentor_setup::phpDocumentor_setup()</a>	696

<i>Checks PHP version, makes sure it is 4.2.0+, and chooses the constructor</i>	<a href="#">lo::lo()</a>	669
<i>creates an array \$this-&gt;phpDocOptions and sets program options in it.</i>		
<i>sets up basic data structures</i>	<a href="#">constructor phpDocumentor_IntermediateParser::phpDocumentor_IntermediateParser()</a>	682
<i>constructor parserTocInlineTag::parserTocInlineTag()</i>	<a href="#">constructor parserTocInlineTag::parserTocInlineTag()</a>	785
<i>constructor</i>		
<i>constructor parserTutorialInlineTag::parserTutorialInlineTag()</i>	<a href="#">constructor parserTutorialInlineTag::parserTutorialInlineTag()</a>	787
<i>constructor</i>		
<i>constructor parserXMLDocBookTag::parserXMLDocBookTag()</i>	<a href="#">constructor parserXMLDocBookTag::parserXMLDocBookTag()</a>	919
<i>sets up the tag</i>		
<i>constructor ObjectWordParser::ObjectWordParser()</i>	<a href="#">constructor ObjectWordParser::ObjectWordParser()</a>	925
<i>ChangeLog</i>	<a href="#">ChangeLog</a>	952
<i>constructor parserEntity::parserEntity()</i>	<a href="#">constructor parserEntity::parserEntity()</a>	917
<i>sets up the entity</i>		
<i>constructor XMLPackagePageParser::XMLPackagePageParser()</i>	<a href="#">constructor XMLPackagePageParser::XMLPackagePageParser()</a>	911
<i>Set up the wordparser</i>		
<i>constructor</i>		
<i>phpDocumentor_XML_Beautifier_Tokenizer::phpDocumentor_XML_Beautifier_Tokenizer()</i>	<a href="#">phpDocumentor_XML_Beautifier_Tokenizer::phpDocumentor_XML_Beautifier_Tokenizer()</a>	906
<i>Initialize the \$tokenpushEvent, \$wordpushEvent arrays</i>		
<i>constructor ppageParser::ppageParser()</i>	<a href="#">constructor ppageParser::ppageParser()</a>	909
<i>set up invariant Parser variables</i>		
<i>Converter.inc</i>	<a href="#">Converter.inc</a>	960
<i>Source code</i>		
<i>CHMdefaultConverter.inc</i>	<a href="#">CHMdefaultConverter.inc</a>	1034
<i>Source code</i>		
<i>clone5.inc.php</i>	<a href="#">clone5.inc.php</a>	1204
<i>Source code</i>		
<i>common.inc.php</i>	<a href="#">common.inc.php</a>	1205
<i>Source code</i>		
<i>clone.inc.php</i>	<a href="#">clone.inc.php</a>	1203
<i>Source code</i>		
<i>Classes.inc</i>	<a href="#">Classes.inc</a>	1185
<i>Source code</i>		
<i>class.phpdocpdf.php</i>	<a href="#">class.phpdocpdf.php</a>	1113
<i>Source code</i>		
<i>config.php</i>	<a href="#">config.php</a>	1171
<i>Source code</i>		
<i>constructor</i>		
<i>phpDocumentor_TutorialHighlightParser::phpDocumentor_TutorialHighlightParser()</i>	<a href="#">phpDocumentor_TutorialHighlightParser::phpDocumentor_TutorialHighlightParser()</a>	902
<i>Initialize the \$tokenpushEvent, \$wordpushEvent arrays</i>		
<i>constructor phpDocumentor_HighlightParser::phpDocumentor_HighlightParser()</i>	<a href="#">constructor phpDocumentor_HighlightParser::phpDocumentor_HighlightParser()</a>	897
<i>Initialize the \$tokenpushEvent, \$wordpushEvent arrays</i>		
<i>constructor parserPage::parserPage()</i>	<a href="#">constructor parserPage::parserPage()</a>	806
<i>sets package to default package</i>		
<i>constructor parserDocBlock::parserDocBlock()</i>	<a href="#">constructor parserDocBlock::parserDocBlock()</a>	817
<i>sets package to default</i>		
<i>constLink::\$type</i>	<a href="#">constLink::\$type</a>	791
<i>constLink</i>	<a href="#">constLink</a>	791
<i>class constant link</i>		
<i>classLink</i>	<a href="#">classLink</a>	790
<i>class link</i>		
<i>classLink::\$type</i>	<a href="#">classLink::\$type</a>	790

<a href="#">constructor parserConst::parserConst()</a>	838
<a href="#">constructor parserMethod::parserMethod()</a>	851
<a href="#">constructor parserDescParser::parserDescParser()</a>	893
<i>sets \$wp to be a ObjectWordParser</i>	
<a href="#">constructor phpDocumentorTParser::phpDocumentorTParser()</a>	896
<i>Constructor</i>	
<a href="#">constructor Parser::Parser()</a>	887
<i>Set up invariant parsing variables</i>	
<a href="#">constructor parserVar::parserVar()</a>	860
<a href="#">constructor parserPackagePage::parserPackagePage()</a>	854
<a href="#">constructor parserTutorial::parserTutorial()</a>	857
<a href="#">Classes::getClassByPackage()</a>	664
<i>Search for a class in a package</i>	
<a href="#">Classes::getClass()</a>	663
<i>Get the parserClass representation of a class from its name and file</i>	
<a href="#">core.rm_auto.php</a>	420
<i>Smarty plugin</i>	
<a href="#">core.run_insert_handler.php</a>	421
<i>Smarty plugin</i>	
<a href="#">core.smarty_include_php.php</a>	422
<i>Smarty plugin</i>	
<a href="#">core.rmdir.php</a>	419
<i>Smarty plugin</i>	
<a href="#">core.read_cache_file.php</a>	418
<i>Smarty plugin</i>	
<a href="#">core.process_cached_inserts.php</a>	416
<i>Smarty plugin</i>	
<a href="#">core.process_compiled_include.php</a>	417
<i>Smarty plugin</i>	
<a href="#">core.write_cache_file.php</a>	423
<i>Smarty plugin</i>	
<a href="#">core.write_compiled_include.php</a>	424
<i>Smarty plugin</i>	
<a href="#">Config</a>	508
<i>Config file for MLib</i>	
<a href="#">constructor brokenlinkstovars::brokenlinkstovars()</a>	531
<a href="#">CONSTANT2</a>	505
<i>blah blah short description</i>	
<a href="#">CONSTANT1</a>	505
<i>blah blah short description</i>	
<a href="#">core.write_compiled_resource.php</a>	425
<i>Smarty plugin</i>	
<a href="#">core.write_file.php</a>	426
<i>Smarty plugin</i>	
<a href="#">core.load_resource_plugin.php</a>	415
<i>Smarty plugin</i>	
<a href="#">core.load_plugins.php</a>	414
<i>Smarty plugin</i>	
<a href="#">constructor Smarty::Smarty()</a>	382
<i>The class constructor.</i>	
<a href="#">constructor Smarty_Compiler::Smarty_Compiler()</a>	396
<i>The class constructor.</i>	
<a href="#">core.assemble_plugin_filepath.php</a>	405

<i>Smarty plugin</i>	
<a href="#">Config_File::set_path()</a>	374
<i>Set the path where configuration files can be found.</i>	
<a href="#">Config_File::load_file()</a>	373
<i>Load a configuration file manually.</i>	
<a href="#">Config_File::get_section_names()</a>	373
<i>Get all section names from a loaded file.</i>	
<a href="#">Config_File::get_var_names()</a>	373
<i>Get all global or section variable names.</i>	
<a href="#">core.assign_smarty_interface.php</a>	406
<i>Smarty plugin</i>	
<a href="#">core.create_dir_structure.php</a>	407
<i>Smarty plugin</i>	
<a href="#">core.is_secure.php</a>	412
<i>Smarty plugin</i>	
<a href="#">core.is_trusted.php</a>	413
<i>Smarty plugin</i>	
<a href="#">core.get_php_resource.php</a>	411
<i>Smarty plugin</i>	
<a href="#">core.get microtime.php</a>	410
<i>Smarty plugin</i>	
<a href="#">core.display_debug_console.php</a>	408
<i>Smarty plugin</i>	
<a href="#">core.get_include_path.php</a>	409
<i>Smarty plugin</i>	
<a href="#">childofpriv</a>	533
<i>what happens here? testing bug 564135</i>	
<a href="#">ClubBase</a>	534
<a href="#">Classes.inc</a>	631
<i>Intermediate class parsing structure.</i>	
<a href="#">clone.inc.php</a>	632
<i>Object clone method</i>	
<a href="#">clone5.inc.php</a>	634
<i>Object clone method</i>	
<a href="#">config.php</a>	623
<i>phpDocumentor :: docBuilder Web Interface</i>	
<a href="#">constructor testClass::testClass()</a>	552
<a href="#">ctest::btest()</a>	538
<a href="#">constructor kiddie_b587733::kiddie_b587733()</a>	548
<a href="#">common.inc.php</a>	635
<i>Common information needed by all portions of the application</i>	
<a href="#">checkForBugCondition()</a>	652
<i>Crash in case of known, dangerous bug condition</i>	
<a href="#">Classes::addPackageToFile()</a>	663
<i>Mark a package as being used in a class</i>	
<a href="#">Classes::addVar()</a>	663
<i>While parsing, add a variable to the list of parsed variables</i>	
<a href="#">Classes::addMethod()</a>	662
<i>While parsing, add a method to the list of parsed methods</i>	
<a href="#">Classes::addConst()</a>	662
<i>While parsing, add a variable to the list of parsed variables</i>	
<a href="#">Classes</a>	661
<i>Intermediate class parsing structure.</i>	

<a href="#">Classes::addClass()</a>	662
<i>While parsing, add a class to the list of parsed classes</i>	
<a href="#">constructor ctest::ctest()</a>	538
<a href="#">ctest::\$t3</a>	538
<a href="#">ClubBase::getPropType()</a>	535
<i>getPropType gibt den Datentyp eines Objekt-Eigenschaft zurueck.</i>	
<a href="#">ClubBase::loadClass()</a>	535
<a href="#">ClubBase::getProperty()</a>	535
<i>getProperty() Gibt den Wert einer Eigenschaft zurueck</i>	
<a href="#">ClubBase::getAllProperties()</a>	534
<i>getAllProperties() Gibt die Wert aller Eigenschaften einer Klasse zurueck</i>	
<a href="#">ClubBase::\$ bPrintFlush</a>	534
<a href="#">constructor ClubBase::ClubBase()</a>	534
<i>ClubBase() Der Konstruktor</i>	
<a href="#">ClubBase::printFlush()</a>	536
<a href="#">ClubBase::printVar()</a>	536
<a href="#">ctest</a>	538
<a href="#">ctest::\$t1</a>	538
<a href="#">ClubBase::_PHPDOCUMENTOR_DEBUG()</a>	537
<i>_PHPDOCUMENTOR_DEBUG() Debugmeldungen ausgeben</i>	
<a href="#">ClubBase::_ERROR()</a>	537
<i>_ERROR() Fehler registrieren und ggf. zur Debug-Ausgabe weiterleiten</i>	
<a href="#">ClubBase::setDebug()</a>	536
<i>set_debug() Debug-Level fuer die Klasse setzen</i>	
<a href="#">ClubBase::setProperty()</a>	536
<i>setProperty() Setzt den Wert einer Eigenschaft</i>	
<a href="#">Cezpdf::ezPrvtTableDrawLines()</a>	324
<a href="#">Cezpdf::ezPrvtTableColumnHeadings()</a>	323
<a href="#">Converter::getFormattedDescMethods()</a>	169
<i>Get a list of methods in child classes that override this method</i>	
<a href="#">Converter::getFormattedDescVars()</a>	170
<i>Get a list of vars in child classes that override this var</i>	
<a href="#">Converter::getFormattedImplements()</a>	170
<i>returns an array containing the class inheritance tree from the root object to the class.</i>	
<a href="#">Converter::getFormattedConflicts()</a>	169
<a href="#">Converter::getFileSourcePath()</a>	169
<i>Return the fixed path to the source-code file folder.</i>	
<a href="#">Converter::getDefineLink()</a>	168
<i>return false or a <a href="#">defineLink</a> to \$expr</i>	
<a href="#">Converter::getFileSourceName()</a>	169
<i>Translate the path info into a unique file name for the highlighted source code.</i>	
<a href="#">Converter::getFormattedInheritedConsts()</a>	170
<i>Return template-enabled list of inherited class constants</i>	
<a href="#">Converter::getFormattedInheritedMethods()</a>	171
<i>Return template-enabled list of inherited methods</i>	
<a href="#">Converter::getGlobalLink()</a>	173
<i>return false or a <a href="#">globalLink</a> to \$expr</i>	
<a href="#">Converter::getGlobalValue()</a>	173
<i>Parse a global variable's default value for class initialization.</i>	
<a href="#">Converter::getFunctionLink()</a>	172
<i>return false or a <a href="#">functionLink</a> to \$expr</i>	

<a href="#">Converter::getFormattedOverrides()</a>	172
<i>Get the method this method overrides, if any</i>	
<a href="#">Converter::getFormattedInheritedVars()</a>	171
<i>Return template-enabled list of inherited variables</i>	
<a href="#">Converter::getFormattedMethodImplements()</a>	172
<i>Get the method this method(s) implemented from an interface, if any</i>	
<a href="#">Converter::getCurrentPageURL()</a>	168
<i>Return the path to the current</i>	
<a href="#">Converter::getCurrentPageLink()</a>	168
<a href="#">Converter::flushHighlightCache()</a>	164
<i>Return the close text for the current token</i>	
<a href="#">Converter::formatIndex()</a>	165
<i>Called by <a href="#">walk()</a> while converting.</i>	
<a href="#">Converter::formatLeftIndex()</a>	165
<i>Called by <a href="#">walk()</a> while converting.</i>	
<a href="#">Converter::exampleProgramExample()</a>	164
<i>Used to convert the {@example} inline tag in a docblock.</i>	
<a href="#">Converter::endPage()</a>	164
<i>Called by <a href="#">walk()</a> while converting, when the last procedural page element has been parsed.</i>	
<a href="#">Converter::EncloseParagraph()</a>	163
<i>Used to enclose a paragraph in a docblock</i>	
<a href="#">Converter::endClass()</a>	163
<i>Called by <a href="#">walk()</a> while converting, when the last class element has been parsed.</i>	
<a href="#">Converter::formatPkgIndex()</a>	165
<i>Called by <a href="#">walk()</a> while converting.</i>	
<a href="#">Converter::formatTutorialTOC()</a>	165
<i>Creates a table of contents for a {@toc} inline tag in a tutorial</i>	
<a href="#">Converter::getConstLink()</a>	167
<i>return false or a <a href="#">constLink</a> to \$expr in \$class</i>	
<a href="#">Converter::getConverterDir()</a>	168
<i>Get the absolute path to the converter's base directory</i>	
<a href="#">Converter::getClassLink()</a>	167
<i>return false or a <a href="#">classLink</a> to \$expr</i>	
<a href="#">Converter::getClassesOnPage()</a>	167
<i>gets a list of all classes declared on a procedural page represented by</i>	
<a href="#">Converter::generateChildClassList()</a>	166
<i>returns a list of child classes</i>	
<a href="#">Converter::generateFormattedClassTree()</a>	166
<i>returns an array containing the class inheritance tree from the root object to the class.</i>	
<a href="#">Converter::getHighlightState()</a>	173
<a href="#">Converter::getId()</a>	173
<i>take <a href="#">abstractLink</a> descendant and text \$eltext and return a</i>	
<a href="#">Converter::Output()</a>	183
<i>do all necessary output</i>	
<a href="#">Converter::postProcess()</a>	184
<i>This version does nothing</i>	
<a href="#">Converter::prepareDocBlock()</a>	184
<i>convert the element's DocBlock for output</i>	
<a href="#">Converter::newSmarty()</a>	183
<i>Return a Smarty template object to operate with</i>	

<a href="#">Converter::ListItem()</a>	183
Used to convert the contents of <<i>> in a docblock	
<a href="#">Converter::Italicize()</a>	182
Used to convert the contents of <<i>> in a docblock	
<a href="#">Converter::Kbdize()</a>	182
Used to convert the contents of <<kbd>> in a docblock	
<a href="#">Converter::PreserveWhiteSpace()</a>	184
Used to convert the contents of <<pre>> in a docblock	
<a href="#">Converter::ProgramExample()</a>	185
Used to convert the <<code>> tag in a docblock	
<a href="#">Converter::setTargetDir()</a>	186
Sets the output directory for generated documentation	
<a href="#">Converter::setTemplateBase()</a>	187
Set the template directory with a different template base directory	
<a href="#">Converter::setSourcePaths()</a>	186
Mark a file as having had source code highlighted	
<a href="#">Converter::Sampize()</a>	186
Used to convert the contents of <<samp>> in a docblock	
<a href="#">Converter::returnLink()</a>	185
take URL \$link and text \$text and return a link in the format needed for the Converter	
<a href="#">Converter::returnSee()</a>	185
take <a href="#">abstractLink</a> descendant and text \$eltext and return a link	
<a href="#">Converter::highlightTutorialSource()</a>	182
Used to allow converters to format the source code of Tutorial XML the way they'd like.	
<a href="#">Converter::highlightSource()</a>	181
Used to allow converters to format the source code the way they'd like.	
<a href="#">Converter::getSortedClassTreeFromClass()</a>	176
Return a tree of all classes that extend this class	
<a href="#">Converter::getSourceLink()</a>	178
<a href="#">Converter::getPageLink()</a>	176
return false or a <a href="#">pageLink</a> to \$expr	
<a href="#">Converter::getMethodLink()</a>	175
return false or a <a href="#">methodLink</a> to \$expr in \$class	
<a href="#">Converter::getIncludeValue()</a>	174
Parse an include's file to see if it is a file documented in this project	
<a href="#">Converter::getLink()</a>	174
The meat of the @see tag and inline {@link} tag	
<a href="#">Converter::getState()</a>	178
Return parserStringWithInlineTags::Convert() cache state	
<a href="#">Converter::getTutorialId()</a>	178
Return a converter-specific id to distinguish tutorials and their sections	
<a href="#">Converter::hasTutorial()</a>	181
<a href="#">Converter::highlightDocBlockSource()</a>	181
Used to allow converters to format the source code of DocBlocks the way they'd like.	
<a href="#">Converter::hasSourceCode()</a>	181
Determine whether an element's file has generated source code, used for linking to line numbers of source.	
<a href="#">Converter::getVarLink()</a>	180
return false or a <a href="#">varLink</a> to \$expr in \$class	
<a href="#">Converter::getTutorialLink()</a>	179

<i>The meat of the @tutorial tag and inline {@tutorial} tag</i>	
<a href="#">Converter::getTutorialTree()</a>	180
Get a tree structure representing the hierarchy of tutorials	
<a href="#">Converter::EncloseList()</a>	163
Used to convert the contents of <> or <> in a docblock	
<a href="#">Converter::createParentDir()</a>	163
Recursively creates all subdirectories that don't exist in the \$dir path	
<a href="#">Converter::\$package_output</a>	150
set to value of -po commandline	
<a href="#">Converter::\$package_parents</a>	150
Hierarchy of packages	
<a href="#">Converter::\$page</a>	151
name of current page being converted	
<a href="#">Converter::\$package_elements</a>	150
alphabetical index of all elements sorted by package, subpackage, page, and class.	
<a href="#">Converter::\$packageCategories</a>	149
Packages associated with categories	
<a href="#">Converter::\$outputformat</a>	149
output format of this converter	
<a href="#">Converter::\$package</a>	149
package name currently being converted	
<a href="#">Converter::\$page_contents</a>	151
alphabetical index of all elements on a page by package/subpackage	
<a href="#">Converter::\$page_data</a>	151
template for the procedural page currently being processed	
<a href="#">Converter::\$processSpecialRoots</a>	152
This converter knows about the new root tree processing	
<a href="#">Converter::\$quietmode</a>	152
controls display of progress information while parsing.	
<a href="#">Converter::\$pkg_elements</a>	152
alphabetical index of all elements, indexed by package/subpackage	
<a href="#">Converter::\$path</a>	152
path of current page being converted	
<a href="#">Converter::\$page_elements</a>	151
alphabetized index of procedural pages by package	
<a href="#">Converter::\$parseprivate</a>	152
controls processing of elements marked private with @access private	
<a href="#">Converter::\$leftindex</a>	148
Controls which of the one-element-only indexes are generated.	
<a href="#">Converter::\$highlightingSource</a>	148
Flag used to help converters determine whether to do special source highlighting	
<a href="#">Converter::\$class</a>	146
set to a classname if currently parsing a class, false if not	
<a href="#">Converter::\$classes</a>	146
All class information, organized by path, and by package	
<a href="#">Converter::\$class_contents</a>	146
alphabetical index of all methods and vars in a class by package/subpackage	
<a href="#">Converter::\$all_packages</a>	145
All packages encountered in parsing	
<a href="#">Converter</a>	145
Base class for all output converters.	
<a href="#">Converter Manual</a>	128

<a href="#">Converter.inc</a>	Base class for all Converters	143
<a href="#">Converter::\$class_data</a>	template for the class currently being processed	146
<a href="#">Converter::\$class_elements</a>	alphabetized index of classes by package	147
<a href="#">Converter::\$function_elements</a>	alphabetized index of functions by package	148
<a href="#">Converter::\$global_elements</a>	alphabetized index of global variables by package	148
<a href="#">Converter::\$elements</a>	alphabetical index of all elements	147
<a href="#">Converter::\$define_elements</a>	alphabetized index of defines by package	147
<a href="#">Converter::\$curfile</a>	full path of the current file being converted	147
<a href="#">Converter::\$curpage</a>	current procedural page being processed	147
<a href="#">Converter::\$smarty_dir</a>	Directory that the smarty templates are in	153
<a href="#">Converter::\$sort_absolutely_everything</a>	This is used if the content must be passed in the order it should be read, i.e. by package, procedural then classes	153
<a href="#">Converter::convertFunction()</a>	Converts function for template output	159
<a href="#">Converter::convertGlobal()</a>	Converts global variables for template output	159
<a href="#">Converter::convertInclude()</a>	Converts includes for template output	160
<a href="#">Converter::ConvertErrorLog()</a>	Convert the phpDocumentor parsing/conversion error log	158
<a href="#">Converter::convertDefine()</a>	Converts defines for template output	158
<a href="#">Converter::convertClass()</a>	Default Class Handler	157
<a href="#">Converter::convertConst()</a>	Converts class constants for template output.	158
<a href="#">Converter::convertMethod()</a>	Converts method for template output	160
<a href="#">Converter::convertPage()</a>	Default Page Handler	160
<a href="#">Converter::Convert_RIC()</a>	Convert README/INSTALL/CHANGELOG file contents to output format	162
<a href="#">Converter::copyFile()</a>	Copies a file from the template directory to the target directory	162
<a href="#">Converter::convertVar()</a>	Converts class variables for template output.	162
<a href="#">Converter::convertTutorial()</a>	Default Tutorial Handler	161
<a href="#">Converter::ConvertTitle()</a>	Convert the title of a Tutorial docbook tag section	161
<a href="#">Converter::ConvertTodoList()</a>	Convert the list of all @todo tags	161

<a href="#">Converter::Convert()</a>	157
<i>Convert all elements to output format</i>	
<a href="#">Converter::cleanup()</a>	157
<i>Finish up parsing/cleanup directories</i>	
<a href="#">Converter::\$templateDir</a>	154
<i>Directory that the template is in, relative to phpDocumentor root directory</i>	
<a href="#">Converter::\$templateName</a>	154
<i>Name of the template, from last part of -o</i>	
<a href="#">Converter::\$targetDir</a>	154
<i>directory that output is sent to. -t command-line sets this.</i>	
<a href="#">Converter::\$subpackage</a>	153
<i>subpackage name currently being converted</i>	
<a href="#">Converter::\$sort_page_contents_by_type</a>	153
<i>This determines whether the <a href="#">\$page_contents</a> array should be sorted by element type as well as alphabetically by name</i>	
<a href="#">Converter::\$sourcePaths</a>	153
<i>A list of files that have had source code generated</i>	
<a href="#">Converter::\$template_options</a>	154
<i>Options for each template, parsed from the options.ini file in the template base directory</i>	
<a href="#">Converter::\$title</a>	154
<a href="#">Converter::Br()</a>	156
<i>Used to convert &lt;&gt;br&gt;&lt;/i&gt; in a docblock</i>	
<a href="#">Converter::checkState()</a>	156
<i>Compare parserStringWithInlineTags::Convert() cache state to \$state</i>	
<a href="#">Converter::Bolden()</a>	156
<i>Used to convert the contents of &lt;&gt;b&gt;&lt;/i&gt; in a docblock</i>	
<a href="#">Converter::AttrToString()</a>	155
<i>Convert the attribute of a Tutorial docbook tag's attribute list</i>	
<a href="#">Converter::\$todoList</a>	155
<i>List of all @todo tags and a link to the element with the @todo</i>	
<a href="#">constructor Converter::Converter()</a>	155
<i>Initialize Converter data structures</i>	
<a href="#">Converter::setTemplateDir()</a>	187
<i>sets the template directory based on the <a href="#">\$outputformat</a> and \$name</i>	
<a href="#">Converter::sortPageContentsByElementType()</a>	187
<i>sorts <a href="#">\$page_contents</a> by element type as well as alphabetically</i>	
<a href="#">CHMdefaultConverter::ProgramExample()</a>	213
<i>Used to convert the &lt;&gt;code&gt;&lt;/i&gt; tag in a docblock</i>	
<a href="#">CHMdefaultConverter::rcNatCmp()</a>	213
<i>does a nat case sort on the specified second level value of the array</i>	
<a href="#">CHMdefaultConverter::rcNatCmp1()</a>	213
<i>does a nat case sort on the specified second level value of the array.</i>	
<a href="#">CHMdefaultConverter::postProcess()</a>	213
<i>Uses htmlspecialchars() on the input</i>	
<a href="#">CHMdefaultConverter::Output()</a>	212
<i>Create the phpdoc.hhp, contents.hhc files needed by MS HTML Help Compiler to create a CHM file</i>	
<a href="#">CHMdefaultConverter::getTutorialId()</a>	211
<a href="#">CHMdefaultConverter::getVarLink()</a>	212
<a href="#">CHMdefaultConverter::returnLink()</a>	214
<a href="#">CHMdefaultConverter::returnSee()</a>	214
<i>This function takes an <a href="#">abstractLink</a> descendant and returns an html link</i>	
<a href="#">CHMdefaultConverter::TutorialExample()</a>	215

<a href="#">CHMdefaultConverter::unmangle()</a>	215
<a href="#">CHMdefaultConverter::sourceLine()</a>	215
<i>Return a line of highlighted source code with formatted line number</i>	
<a href="#">CHMdefaultConverter::SmartyInit()</a>	214
<a href="#">CHMdefaultConverter::setTargetDir()</a>	214
<i>calls the converter setTargetDir, and then copies any template images and the stylesheet if they haven't been copied</i>	
<a href="#">CHMdefaultConverter::setTemplateDir()</a>	214
<a href="#">CHMdefaultConverter::getSourceLink()</a>	211
<a href="#">CHMdefaultConverter::getSourceAnchor()</a>	211
<i>Retrieve a Converter-specific anchor to a segment of a source code file parsed via a <a href="#">@filesource</a> tag.</i>	
<a href="#">CHMdefaultConverter::getCurrentPageLink()</a>	208
<a href="#">CHMdefaultConverter::getDefineLink()</a>	208
<a href="#">CHMdefaultConverter::getExampleLink()</a>	208
<a href="#">CHMdefaultConverter::getConstLink()</a>	207
<a href="#">CHMdefaultConverter::getClassLink()</a>	207
<a href="#">CHMdefaultConverter::generateTOC()</a>	207
<a href="#">CHMdefaultConverter::getCDATA()</a>	207
<a href="#">CHMdefaultConverter::getFunctionLink()</a>	208
<a href="#">CHMdefaultConverter::getGlobalLink()</a>	209
<a href="#">CHMdefaultConverter::getPageName()</a>	210
<a href="#">CHMdefaultConverter::getRootTree()</a>	211
<i>return formatted class tree for the Class Trees page</i>	
<a href="#">CHMdefaultConverter::getPageLink()</a>	210
<a href="#">CHMdefaultConverter::getMethodLink()</a>	210
<a href="#">CHMdefaultConverter::getId()</a>	209
<a href="#">CHMdefaultConverter::getIndexInformation()</a>	209
<i>Generate indexing information for given element</i>	
<a href="#">CHMdefaultConverter::writeExample()</a>	215
<a href="#">CHMdefaultConverter::writefile()</a>	216
<i>Write a file to disk, and add it to the \$hhp_files list of files to include in the generated CHM</i>	
<a href="#">Cezpdf::ezColumnsStop()</a>	322
<a href="#">Cezpdf::ezGetCurrentPageNumber()</a>	322
<a href="#">Cezpdf::ezImage()</a>	322
<a href="#">Cezpdf::ezColumnsStart()</a>	322
<a href="#">Cezpdf::execTemplate()</a>	322
<a href="#">constructor Cezpdf::Cezpdf()</a>	321
<a href="#">Cezpdf::alink()</a>	322
<a href="#">Cezpdf::ezInsertMode()</a>	322
<a href="#">Cezpdf::ezNewPage()</a>	323
<a href="#">Cezpdf::ezPrvtGetTextWidth()</a>	323
<a href="#">Cezpdf::ezPRVTpageNumberSearch()</a>	323
<a href="#">Cezpdf::ezPRVTcleanUp()</a>	323
<a href="#">Cezpdf::ezPRVTaddPageNumbers()</a>	323
<a href="#">Cezpdf::ezOutput()</a>	323
<a href="#">Cezpdf::ezProcessText()</a>	323
<a href="#">Cezpdf::\$y</a>	321
<a href="#">Cezpdf::\$ezPages</a>	321
<a href="#">class.phpdocpdf.php</a>	258
<i>Cezpdf callback class customized for phpDocumentor</i>	
<a href="#">constructor PDFdefaultConverter::PDFdefaultConverter()</a>	263

<a href="#">constructor HTMLSmartyConverter::HTMLSmartyConverter()</a>	240
<a href="#">constructor HTMLframesConverter::HTMLframesConverter()</a>	221
sets <i>\$base_dir</i> to <i>\$targetDir</i>	
<a href="#">CHMdefaultConverter::writeNewPPage()</a>	216
<a href="#">CHMdefaultConverter::writeSource()</a>	216
<a href="#">constructor PDFParser::PDFParser()</a>	273
<i>Sets up the wordparser for this class</i>	
<a href="#">constructor phpdcpdf::phpdcpdf()</a>	275
<a href="#">Cezpdf::\$ez</a>	321
<a href="#">Cezpdf::\$ezPageCount</a>	321
<a href="#">Cezpdf</a>	321
<i>This class will take the basic interaction facilities of the Cpdf class and make more useful functions so that the user does not have to know all the ins and outs of pdf presentation to produce something pretty.</i>	
<a href="#">class.ezpdf.php</a>	320
<a href="#">constructor XMLDocBookConverter::XMLDocBookConverter()</a>	285
sets <i>\$base_dir</i> to <i>\$targetDir</i>	
<a href="#">constructor XMLDocBookpeardoc2Converter::XMLDocBookpeardoc2Converter()</a>	303
sets <i>\$base_dir</i> to <i>\$targetDir</i>	
<a href="#">CHMdefaultConverter::generatePkgElementIndexes()</a>	207
<a href="#">CHMdefaultConverter::generatePkgElementIndex()</a>	206
<i>Generate alphabetical index of all elements by package and subpackage</i>	
<a href="#">CHMdefaultConverter::\$class_dir</a>	195
<i>output directory for the current class being processed</i>	
<a href="#">CHMdefaultConverter::\$current</a>	195
<i>contains all of the template procedural page element loop data needed for the current template</i>	
<a href="#">CHMdefaultConverter::\$currentclass</a>	195
<i>contains all of the template class element loop data needed for the current template</i>	
<a href="#">CHMdefaultConverter::\$base_dir</a>	195
<i>target directory passed on the command-line.</i>	
<a href="#">CHMdefaultConverter</a>	194
<i>Generates files that MS HTML Help Worshop can use to create a MS Windows compiled help file (CHM)</i>	
<a href="#">Converter:: tutorial_path()</a>	193
<i>Returns the path to this tutorial as a string</i>	
<a href="#">CHMdefaultConverter.inc</a>	194
<i>CHM (Compiled Help Manual) output converter for Smarty Template.</i>	
<a href="#">CHMdefaultConverter::\$juststarted</a>	196
<i>controls formatting of parser informative output</i>	
<a href="#">CHMdefaultConverter::\$KLinks</a>	196
<i>Table of Contents entry for index.hhk</i>	
<a href="#">CHMdefaultConverter::\$page_dir</a>	197
<i>output directory for the current procedural page being processed</i>	
<a href="#">CHMdefaultConverter::\$ric_set</a>	197
<a href="#">CHMdefaultConverter::\$package_pages</a>	196
<i>array of converted package page names.</i>	
<a href="#">CHMdefaultConverter::\$outputformat</a>	196
<a href="#">CHMdefaultConverter::\$leftindex</a>	196
<i>indexes of elements by package that need to be generated</i>	
<a href="#">CHMdefaultConverter::\$name</a>	196
<a href="#">Converter:: setHighlightCache()</a>	193
<a href="#">Converter:: rmdir()</a>	193

<i>Completely remove a directory and its contents</i>	
<a href="#">Converter::TutorialExample()</a>	189
<a href="#">Converter::type_adjust()</a>	189
<i>Called by <a href="#">parserReturnTag::Convert()</a> to allow converters to change type names to desired formatting</i>	
<a href="#">Converter::TranslateTag()</a>	188
<i>Used to translate an XML DocBook tag from a tutorial by reading the options.ini file for the template.</i>	
<a href="#">Converter::TranslateEntity()</a>	188
<i>Used to translate an XML DocBook entity like &amp;rdquo; from a tutorial by reading the options.ini file for the template.</i>	
<a href="#">Converter::sourceLine()</a>	188
<i>Return a line of highlighted source code with formatted line number</i>	
<a href="#">Converter::startHighlight()</a>	188
<i>Initialize highlight caching</i>	
<a href="#">Converter::unmangle()</a>	189
<i>Called by <a href="#">parserSourceInlineTag::stringConvert()</a> to allow converters to format the source code the way they'd like.</i>	
<a href="#">Converter::vardump_tree()</a>	190
<i>Debugging function for dumping \$tutorial_tree</i>	
<a href="#">Converter::writeFile()</a>	192
<i>Writes a file to target dir</i>	
<a href="#">Converter::writeSource()</a>	192
<i>Write out the formatted source code for a php file</i>	
<a href="#">Converter::writeExample()</a>	191
<i>Write out the formatted source code for an example php file</i>	
<a href="#">Converter::walk_everything()</a>	191
<i>walk over elements by package rather than page</i>	
<a href="#">Converter::Varize()</a>	190
<i>Used to convert the contents of &lt;&gt; in a docblock</i>	
<a href="#">Converter::walk()</a>	190
<i>called by <a href="#">phpDocumentor_InterpreterParser::Convert()</a> to traverse the array of pages and their elements, converting them to the output format</i>	
<a href="#">CHMdefaultConverter::\$sort_page_contents_by_type</a>	197
<i>CHMdefaultConverter wants elements sorted by type as well as alphabetically</i>	
<a href="#">CHMdefaultConverter::\$wrote</a>	197
<a href="#">CHMdefaultConverter::endClass()</a>	203
<i>Writes out the template file of <a href="#">\$class_data</a> and unsets the template to save memory</i>	
<a href="#">CHMdefaultConverter::endPage()</a>	203
<i>Writes out the template file of <a href="#">\$page_data</a> and unsets the template to save memory</i>	
<a href="#">CHMdefaultConverter::formatIndex()</a>	203
<i>CHMdefaultConverter uses this function to format template index.html and packages.html</i>	
<a href="#">CHMdefaultConverter::copyMediaRecursively()</a>	203
<a href="#">CHMdefaultConverter::Convert_RIC()</a>	203
<i>Convert README/INSTALL/CHANGELOG file contents to output format</i>	
<a href="#">CHMdefaultConverter::convertTutorial()</a>	202
<a href="#">CHMdefaultConverter::convertVar()</a>	202
<i>Converts class variables for template output</i>	
<a href="#">CHMdefaultConverter::formatLeftIndex()</a>	204
<i>Generate indexes for li_package.html and classtree output files</i>	
<a href="#">CHMdefaultConverter::formatPkgIndex()</a>	204
<i>CHMdefaultConverter chooses to format both package indexes and the complete index here</i>	

<a href="#">CHMdefaultConverter::generateFormattedInterfaceTrees()</a>	206
<i>returns a template-enabled array of interface inheritance trees</i>	
<a href="#">CHMdefaultConverter::generateKLinks()</a>	206
<i>Get the table of contents for index.hhk</i>	
<a href="#">CHMdefaultConverter::generateFormattedClassTrees()</a>	205
<i>returns a template-enabled array of class trees</i>	
<a href="#">CHMdefaultConverter::generateFormattedClassTree()</a>	205
<i>returns an array containing the class inheritance tree from the root object to the class</i>	
<a href="#">CHMdefaultConverter::formatTutorialTOC()</a>	205
<i>Use the template tutorial_toc.tpl to generate a table of contents for HTML</i>	
<a href="#">CHMdefaultConverter::generateElementIndex()</a>	205
<i>Generate alphabetical index of all elements</i>	
<a href="#">CHMdefaultConverter::ConvertTodoList()</a>	202
<a href="#">CHMdefaultConverter::convertPage()</a>	202
<i>converts procedural pages for template output</i>	
<a href="#">CHMdefaultConverter::addTOC()</a>	199
<a href="#">CHMdefaultConverter::convertClass()</a>	199
<i>Converts class for template output</i>	
<a href="#">CHMdefaultConverter::addSourceTOC()</a>	198
<a href="#">CHMdefaultConverter::addKLink()</a>	198
<i>Add an item to the index.hhk file</i>	
<a href="#">constructor CHMdefaultConverter::CHMdefaultConverter()</a>	197
<i>sets \$base_dir to \$targetDir</i>	
<a href="#">CHMdefaultConverter::addHHP()</a>	198
<a href="#">CHMdefaultConverter::convertConst()</a>	199
<i>Converts class constants for template output</i>	
<a href="#">CHMdefaultConverter::convertDefine()</a>	200
<i>Converts defines for template output</i>	
<a href="#">CHMdefaultConverter::convertMethod()</a>	201
<i>Converts class methods for template output</i>	
<a href="#">CHMdefaultConverter::convertPackagepage()</a>	201
<i>Converts package page and sets its package as used in \$package_pages</i>	
<a href="#">CHMdefaultConverter::convertInclude()</a>	201
<i>Converts include elements for template output</i>	
<a href="#">CHMdefaultConverter::convertGlobal()</a>	201
<i>Converts global variables for template output</i>	
<a href="#">CHMdefaultConverter::ConvertErrorLog()</a>	200
<i>Create errors.html template file output</i>	
<a href="#">CHMdefaultConverter::convertFunction()</a>	200
<i>Converts function for template output</i>	
<a href="#">Converter Default Template Variables</a>	117

## D

<a href="#">defineLink</a>	791
<i>define link</i>	
<a href="#">DocBlockTags.inc</a>	725
<i>All abstract representations of DocBlock tags are defined by the classes in this file</i>	
<a href="#">defineLink::\$type</a>	791
<a href="#">DescHTML.inc</a>	1253
<i>Source code</i>	

<a href="#">DocBlockTags.inc</a>	. . . . .	1259
Source code		
<a href="#">DescHTML.inc</a>	. . . . .	715
<i>All abstract representations of html tags in DocBlocks are handled by the classes in this file</i>		
<a href="#">decideOnOrOff()</a>	. . . . .	652
<i>Fuzzy logic to interpret the boolean args' intent</i>		
<a href="#">DirNode::\$path</a>	. . . . .	353
<i>full path to this node</i>		
<a href="#">DirNode</a>	. . . . .	353
<i>Directory Node</i>		
<a href="#">DIR_SEP</a>	. . . . .	368
<i>DIR_SEP isn't used anymore, but third party apps might</i>		
<a href="#">dummy()</a>	. . . . .	527
<i>I'm a dummy function currently you need at least one function in your procedural page for this to work</i>		
<a href="#">debug()</a>	. . . . .	635
<i>Debugging output</i>		
<a href="#">Documentable PHP Elements</a>	. . . . .	36

## E

<a href="#">ErrorTracker::addErrorReturn()</a>	. . . . .	768
<i>add a new error to the \$errors array and returns the error string</i>		
<a href="#">ErrorTracker::addWarning()</a>	. . . . .	768
<i>add a new warning to the \$warnings array</i>		
<a href="#">ErrorTracker::addError()</a>	. . . . .	767
<i>add a new error to the \$errors array</i>		
<a href="#">ErrorTracker::\$warnings</a>	. . . . .	767
<i>array of RecordWarnings</i>		
<a href="#">ErrorTracker::\$linenum</a>	. . . . .	767
<a href="#">ErrorTracker::handleEvent()</a>	. . . . .	768
<i>This function subscribes to two events in the Parser in order to keep track of line number information and file name.</i>		
<a href="#">ErrorTracker::returnErrors()</a>	. . . . .	769
<i>Get sorted array of all non-fatal errors in parsing/conversion</i>		
<a href="#">EventStack.inc</a>	. . . . .	1210
Source code		
<a href="#">Errors.inc</a>	. . . . .	1277
Source code		
<a href="#">ErrorTracker::returnWarnings()</a>	. . . . .	769
<i>Get sorted array of all warnings in parsing/conversion</i>		
<a href="#">ErrorTracker::returnLastWarning()</a>	. . . . .	769
<i>Get the warning message of the last warning</i>		
<a href="#">ErrorTracker::returnLastError()</a>	. . . . .	769
<i>Get the error message of the last error</i>		
<a href="#">ErrorTracker::\$lastwarning</a>	. . . . .	767
<i>index in \$warnings of last warning triggered</i>		
<a href="#">ErrorTracker::\$lasterror</a>	. . . . .	767
<i>index in \$errors of last error triggered</i>		
<a href="#">EventStack::\$num</a>	. . . . .	668
<i>The number of events in the stack</i>		

<a href="#">EventStack::\$stack</a>	. . . . .	668
<i>The stack</i>		
<a href="#">EventStack</a>	. . . . .	667
<i>An event Stack</i>		
<a href="#">EventStack.inc</a>	. . . . .	638
<i>An Event Stack for inter-program communication, particularly for parsing</i>		
<a href="#">ezStartPageNumbers()</a>	. . . . .	516
<i>tests function param bug</i>		
<a href="#">EventStack::getEvent()</a>	. . . . .	668
<i>Get the current event</i>		
<a href="#">EventStack::popEvent()</a>	. . . . .	668
<i>Pop an event from the stack</i>		
<a href="#">ErrorTracker::\$curfile</a>	. . . . .	767
<a href="#">ErrorTracker::\$errors</a>	. . . . .	767
<i>array of RecordErrors</i>		
<a href="#">ErrorTracker</a>	. . . . .	766
<i>contains all the errors/warnings</i>		
<a href="#">Errors.inc</a>	. . . . .	750
<i>Error handling for phpDocumentor</i>		
<a href="#">EventStack::pushEvent()</a>	. . . . .	668
<i>Push an event onto the stack</i>		
<a href="#">element</a>	. . . . .	365
<i>I'm an element</i>		

## F

<a href="#">func3()</a>	. . . . .	500
<a href="#">few</a>	. . . . .	538
<a href="#">few::\$pfh</a>	. . . . .	539
<i>****==really fancy==****</i>		
<a href="#">func2()</a>	. . . . .	500
<a href="#">func1()</a>	. . . . .	500
<i>func 1</i>		
<a href="#">function.popup_init.php</a>	. . . . .	455
<i>Smarty plugin</i>		
<a href="#">function.var_dump.php</a>	. . . . .	456
<a href="#">func()</a>	. . . . .	497
<a href="#">functionincomment</a>	. . . . .	539
<a href="#">functionincomment::process()</a>	. . . . .	539
<a href="#">FAQ</a>	. . . . .	954
<a href="#">file_dialog.php</a>	. . . . .	1164
<i>Source code</i>		
<a href="#">find_phpdoc.php</a>	. . . . .	1356
<i>Source code</i>		
<a href="#">find_phpdoc.php</a>	. . . . .	914
<i>Utility file to locate phpDocumentor for a non-PEAR installation</i>		
<a href="#">functionLink::\$type</a>	. . . . .	792
<a href="#">fancy_debug()</a>	. . . . .	636
<i>Returns a formatted var_dump for debugging purposes.</i>		
<a href="#">functionLink</a>	. . . . .	792
<i>function link</i>		
<a href="#">function.popup.php</a>	. . . . .	454

<i>Smarty plugin</i>	
<a href="#">function.math.php</a>	453
<i>Smarty plugin</i>	
<a href="#">function.cycle.php</a>	433
<i>Smarty plugin</i>	
<a href="#">function.debug.php</a>	435
<i>Smarty plugin</i>	
<a href="#">function.eval.php</a>	436
<i>Smarty plugin</i>	
<a href="#">function.counter.php</a>	432
<i>Smarty plugin</i>	
<a href="#">function.config_load.php</a>	431
<i>Smarty plugin</i>	
<a href="#">function.assign.php</a>	429
<i>Smarty plugin</i>	
<a href="#">function.assign_debug_info.php</a>	430
<i>Smarty plugin</i>	
<a href="#">function.fetch.php</a>	437
<i>Smarty plugin</i>	
<a href="#">function.html_checkboxes.php</a>	438
<i>Smarty plugin</i>	
<a href="#">function.html_select_time.php</a>	448
<i>Smarty plugin</i>	
<a href="#">function.html_table.php</a>	449
<i>Smarty plugin</i>	
<a href="#">function.mailto.php</a>	451
<i>Smarty plugin</i>	
<a href="#">function.html_select_date.php</a>	446
<i>Smarty plugin</i>	
<a href="#">function.html_radios.php</a>	444
<i>Smarty plugin</i>	
<a href="#">function.html_image.php</a>	440
<i>Smarty plugin</i>	
<a href="#">function.html_options.php</a>	442
<i>Smarty plugin</i>	
<a href="#">file_dialog.php</a>	351
<i>phpDocumentor :: docBuilder Web Interface</i>	

## G

<a href="#">global \$phpDocumentor_errors</a>	750
<a href="#">global \$_phpDocumentor_setting</a>	652
<i>\$_phpDocumentor_setting is either the value from the web interface, or is set up by /o::parseArgv()</i>	
<a href="#">global \$phpDocumentor_error_descrip</a>	754
<i>Error messages for phpDocumentor parser errors</i>	
<a href="#">global \$phpDocumentor_warning_descrip</a>	756
<i>Error messages for phpDocumentor parser warnings</i>	
<a href="#">globalLink::\$type</a>	793
<a href="#">globalLink</a>	792
<i>global variable link</i>	
<a href="#">global \$phpDocumentor_DefaultPackageName</a>	652

<i>default package name, set using -dn --defaultpackagename</i>	
<a href="#">global \$phpDocumentor_DefaultCategoryName</a>	651
<i>default package name, set using -dn --defaultcategoryname</i>	
<a href="#">globalSetVar()</a>	519
<i>Set a variable, used by the template engine but available to all scripts</i>	
<a href="#">globalGetVar()</a>	519
<i>Get a variable's value</i>	
<a href="#">global \$bqnp_tester</a>	528
<i>The tokenizer splits up strings that have inline variables</i>	
<a href="#">getDir()</a>	625
<a href="#">global \$interface</a>	651
<i>\$interface is either 'web' or is not set at all</i>	
<a href="#">get_include_path()</a>	640
<a href="#">global \$ LIBRARY_DSN</a>	508
<i>Global DSN to be used by classes</i>	

## H

<a href="#">HTMLSmartyConverter::getPageContents()</a>	252
<a href="#">HTMLSmartyConverter::getMethodLink()</a>	251
<a href="#">HTMLSmartyConverter::getPageLeft()</a>	252
<a href="#">HTMLSmartyConverter::getPageLink()</a>	252
<a href="#">HTMLSmartyConverter::getPageName()</a>	252
<a href="#">HTMLSmartyConverter::getIndexInformation()</a>	251
<i>Generate indexing information for given element</i>	
<a href="#">HTMLSmartyConverter::getId()</a>	251
<a href="#">HTMLSmartyConverter::getDefineLink()</a>	250
<a href="#">HTMLSmartyConverter::getExampleLink()</a>	250
<a href="#">HTMLSmartyConverter::getFunctionLink()</a>	250
<a href="#">HTMLSmartyConverter::getGlobalLink()</a>	250
<a href="#">HTMLSmartyConverter::getRootTree()</a>	252
<i>return formatted class tree for the Class Trees page</i>	
<a href="#">HTMLSmartyConverter::getSourceAnchor()</a>	253
<i>Retrieve a Converter-specific anchor to a segment of a source code file parsed via a @filesource tag.</i>	
<a href="#">HTMLSmartyConverter::postProcess()</a>	254
<i>Uses htmlspecialchars() on the input</i>	
<a href="#">HTMLSmartyConverter::Output()</a>	254
<i>This function is not used by HTMLDefaultConverter, but is required by Converter</i>	
<a href="#">HTMLSmartyConverter::ProgramExample()</a>	254
<i>Used to convert the &lt;&gt;code&lt;&gt; tag in a docblock</i>	
<a href="#">HTMLSmartyConverter::rcNatCmp()</a>	254
<i>does a nat case sort on the specified second level value of the array</i>	
<a href="#">HTMLSmartyConverter::rcNatCmp1()</a>	255
<i>does a nat case sort on the specified second level value of the array.</i>	
<a href="#">HTMLSmartyConverter::makeLeft()</a>	254
<a href="#">HTMLSmartyConverter::getVarLink()</a>	253
<a href="#">HTMLSmartyConverter::getSourceLink()</a>	253
<a href="#">HTMLSmartyConverter::getTutorialId()</a>	253
<a href="#">HTMLSmartyConverter::getTutorialList()</a>	253
<a href="#">HTMLSmartyConverter::getTutorialTree()</a>	253
<a href="#">HTMLSmartyConverter::getCurrentPageLink()</a>	249

<a href="#">HTMLSmartyConverter::getConstLink()</a>	249
<a href="#">HTMLSmartyConverter::endClass()</a>	245
<i>Writes out the template file of <code>\$class_data</code> and unsets the template to save memory</i>	
<a href="#">HTMLSmartyConverter::copyMediaRecursively()</a>	245
<a href="#">HTMLSmartyConverter::endPage()</a>	245
<i>Writes out the template file of <code>\$page_data</code> and unsets the template to save memory</i>	
<a href="#">HTMLSmartyConverter::formatIndex()</a>	245
<i>HTMLdefaultConverter uses this function to format template index.html and packages.html</i>	
<a href="#">HTMLSmartyConverter::formatLeftIndex()</a>	246
<i>Generate indexes for li_package.html and classtree output files</i>	
<a href="#">HTMLSmartyConverter::Convert_RIC()</a>	245
<i>Convert README/INSTALL/CHANGELOG file contents to output format</i>	
<a href="#">HTMLSmartyConverter::convertVar()</a>	244
<i>Converts class variables for template output</i>	
<a href="#">HTMLSmartyConverter::convertPackagepage()</a>	243
<i>Converts package page and sets its package as used in <code>\$package_pages</code></i>	
<a href="#">HTMLSmartyConverter::convertPage()</a>	244
<i>converts procedural pages for template output</i>	
<a href="#">HTMLSmartyConverter::ConvertTodoList()</a>	244
<a href="#">HTMLSmartyConverter::convertTutorial()</a>	244
<a href="#">HTMLSmartyConverter::formatPkgIndex()</a>	246
<i>HTMLdefaultConverter chooses to format both package indexes and the complete index here</i>	
<a href="#">HTMLSmartyConverter::formatTutorialTOC()</a>	247
<i>Use the template tutorial_toc.tpl to generate a table of contents for HTML</i>	
<a href="#">HTMLSmartyConverter::getCDATA()</a>	249
<a href="#">HTMLSmartyConverter::getClassContents()</a>	249
<a href="#">HTMLSmartyConverter::getClassLeft()</a>	249
<a href="#">HTMLSmartyConverter::getClassLink()</a>	249
<a href="#">HTMLSmartyConverter::generatePkgElementIndexes()</a>	248
<a href="#">HTMLSmartyConverter::generatePkgElementIndex()</a>	248
<i>Generate alphabetical index of all elements by package and subpackage</i>	
<a href="#">HTMLSmartyConverter::generateElementIndex()</a>	247
<i>Generate alphabetical index of all elements</i>	
<a href="#">HTMLSmartyConverter::generateFormattedClassTree()</a>	247
<i>returns an array containing the class inheritance tree from the root object to the class</i>	
<a href="#">HTMLSmartyConverter::generateFormattedClassTrees()</a>	247
<i>returns a template-enabled array of class trees</i>	
<a href="#">HTMLSmartyConverter::generateFormattedInterfaceTrees()</a>	248
<i>returns a template-enabled array of interface inheritance trees</i>	
<a href="#">HTMLSmartyConverter::returnLink()</a>	255
<a href="#">HTMLSmartyConverter::returnSee()</a>	255
<i>This function takes an <code>abstractLink</code> descendant and returns an html link</i>	
<a href="#">HTML_TreeNode::\$events</a>	360
<i>Javascript event handlers;</i>	
<a href="#">HTML_TreeNode::\$ensureVisible</a>	360
<i>Should this node be made visible?</i>	
<a href="#">HTML_TreeNode::\$expanded</a>	360
<i>Whether this node is expanded or not</i>	
<a href="#">HTML_TreeNode::\$icon</a>	360
<i>The icon for this node.</i>	
<a href="#">HTML_TreeNode::\$isDynamic</a>	361
<i>Whether this node is dynamic or not</i>	

<a href="#">HTML_TreeNode::\$cssClass</a>	360
<i>The css class for this node</i>	
<a href="#">HTML_TreeNode</a>	360
<i>HTML_TreeNode class</i>	
<a href="#">HTML_TreeMenu_Listbox::toHTML()</a>	358
<i>Returns the HTML generated</i>	
<a href="#">HTML_TreeMenu_Presentation</a>	358
<i>HTML_TreeMenu_Presentation class</i>	
<a href="#">HTML_TreeMenu_Presentation::\$menu</a>	359
<i>The TreeMenu structure</i>	
<a href="#">HTML_TreeMenu_Presentation::printMenu()</a>	359
<i>Prints the HTML generated by the toHTML() method.</i>	
<a href="#">HTML_TreeNode::\$items</a>	361
<i>Indexed array of subnodes</i>	
<a href="#">HTML_TreeNode::\$link</a>	361
<i>The link for this node.</i>	
<a href="#">HighlightParser.inc</a>	868
<i>Source Code Highlighting</i>	
<a href="#">HighlightParserTests.php</a>	655
<a href="#">HTMLframesConverter.inc</a>	1060
<i>Source code</i>	
<a href="#">HTMLSmartyConverter.inc</a>	1086
<i>Source code</i>	
<a href="#">HighlightParser.inc</a>	1323
<i>Source code</i>	
<a href="#">htmlArraySelect()</a>	625
<i>Returns a select box based on an key,value array where selected is based on key</i>	
<a href="#">HighlightParserGetInlineTagsTests.php</a>	555
<i>Unit Tests for the HighlightParser-&gt;getInlineTags() method</i>	
<a href="#">HTML_TreeNode::\$parent</a>	361
<i>The parent node. Null if top level</i>	
<a href="#">HTML_TreeNode::\$text</a>	361
<i>The text for this node.</i>	
<a href="#">HTML_TreeNode::addItem()</a>	362
<i>Adds a new subnode to this node.</i>	
<a href="#">HTML_TreeNode::setOption()</a>	362
<i>Allows setting of various parameters after the initial</i>	
<a href="#">HTML_TreeMenu_Listbox::\$promoText</a>	358
<i>The text that is displayed in the first option</i>	
<a href="#">HTML_TreeMenu_Listbox::\$linkTarget</a>	358
<i>Target for the links generated</i>	
<a href="#">HTMLSmartyConverter::writeRIC()</a>	257
<a href="#">HTMLSmartyConverter::writeNewPPage()</a>	257
<a href="#">HTMLSmartyConverter::writeSource()</a>	257
<a href="#">HTML_TreeMenu</a>	354
<i>HTML_TreeMenu Class</i>	
<a href="#">HTML_TreeMenu::\$items</a>	354
<i>Indexed array of subnodes</i>	
<a href="#">HTMLSmartyConverter::writeExample()</a>	257
<a href="#">HTMLSmartyConverter::unmangle()</a>	256
<a href="#">HTMLSmartyConverter::setTargetDir()</a>	255
<i>calls the converter setTargetDir, and then copies any template images and the stylesheet if they haven't been copied</i>	

<a href="#">HTMLSmartyConverter::SmartyInit()</a>	256
<a href="#">HTMLSmartyConverter::sourceLine()</a>	256
<i>Return a line of highlighted source code with formatted line number</i>	
<a href="#">HTMLSmartyConverter::TutorialExample()</a>	256
<a href="#">HTML_TreeMenu::addItem()</a>	354
<i>This function adds an item to the tree.</i>	
<a href="#">HTML_TreeMenu_DHTML</a>	355
<i>HTML_TreeMenu_DHTML class</i>	
<a href="#">HTML_TreeMenu_DHTML::toHTML()</a>	357
<i>Returns the HTML for the menu. This method can be used instead of printMenu() to use the menu system with a template system.</i>	
<a href="#">HTML_TreeMenu_Listbox</a>	357
<i>HTML_TreeMenu_Listbox class</i>	
<a href="#">HTML_TreeMenu_Listbox::\$IndentChar</a>	357
<i>The character used for indentation</i>	
<a href="#">HTML_TreeMenu_Listbox::\$IndentNum</a>	357
<i>How many of the indent chars to use</i>	
<a href="#">HTML_TreeMenu_DHTML::\$userPersistence</a>	356
<i>Whether to use clientside persistence or not</i>	
<a href="#">HTML_TreeMenu_DHTML::\$NoTopLevelImages</a>	356
<i>Whether to skip first level branch images</i>	
<a href="#">HTML_TreeMenu_DHTML::\$defaultClass</a>	355
<i>The default CSS class for the nodes</i>	
<a href="#">HTML_TreeMenu_DHTML::\$images</a>	355
<i>Path to the images</i>	
<a href="#">HTML_TreeMenu_DHTML::\$isDynamic</a>	356
<i>Dynamic status of the treemenu. If true (default) this has no effect. If false it will override all dynamic status vars and set the menu to be fully expanded an non-dynamic.</i>	
<a href="#">HTML_TreeMenu_DHTML::\$linkTarget</a>	356
<i>Target for the links generated</i>	
<a href="#">HTMLSmartyConverter::convertMethod()</a>	243
<i>Converts class methods for template output</i>	
<a href="#">HTMLSmartyConverter::convertInclude()</a>	243
<i>Converts include elements for template output</i>	
<a href="#">HTMLframesConverter::endPage()</a>	226
<i>Writes out the template file of <a href="#">\$page_data</a> and unsets the template to save memory</i>	
<a href="#">HTMLframesConverter::endClass()</a>	226
<i>Writes out the template file of <a href="#">\$class_data</a> and unsets the template to save memory</i>	
<a href="#">HTMLframesConverter::formatIndex()</a>	226
<i>HTMLDefaultConverter uses this function to format template index.html and packages.html</i>	
<a href="#">HTMLframesConverter::formatLeftIndex()</a>	227
<i>Generate indexes for li_package.html and classtree output files</i>	
<a href="#">HTMLframesConverter::formatPkgIndex()</a>	227
<i>HTMLDefaultConverter chooses to format both package indexes and the complete index here</i>	
<a href="#">HTMLframesConverter::copyMediaRecursively()</a>	226
<a href="#">HTMLframesConverter::Convert_RIC()</a>	225
<i>Convert README/INSTALL/CHANGELOG file contents to output format</i>	
<a href="#">HTMLframesConverter::convertPage()</a>	224
<i>converts procedural pages for template output</i>	
<a href="#">HTMLframesConverter::ConvertTodoList()</a>	225

<a href="#">HTMLframesConverter::convertTutorial()</a>	225
<a href="#">HTMLframesConverter::convertVar()</a>	225
Converts class variables for template output	
<a href="#">HTMLframesConverter::formatTutorialTOC()</a>	227
Use the template <code>tutorial_toc.tpl</code> to generate a table of contents for HTML	
<a href="#">HTMLframesConverter::generateElementIndex()</a>	228
Generate alphabetical index of all elements	
<a href="#">HTMLframesConverter::getConstLink()</a>	230
<a href="#">HTMLframesConverter::getClassLink()</a>	229
<a href="#">HTMLframesConverter::getCurrentPageLink()</a>	230
<a href="#">HTMLframesConverter::getDefineLink()</a>	230
<a href="#">HTMLframesConverter::getExampleLink()</a>	231
<a href="#">HTMLframesConverter::getCData()</a>	229
<a href="#">HTMLframesConverter::generatePkgElementIndexes()</a>	229
<a href="#">HTMLframesConverter::generateFormattedClassTree()</a>	228
returns an array containing the class inheritance tree from the root object to the class	
<a href="#">HTMLframesConverter::generateFormattedClassTrees()</a>	228
returns a template-enabled array of class trees	
<a href="#">HTMLframesConverter::generateFormattedInterfaceTrees()</a>	228
returns a template-enabled array of interface inheritance trees	
<a href="#">HTMLframesConverter::generatePkgElementIndex()</a>	229
Generate alphabetical index of all elements by package and subpackage	
<a href="#">HTMLframesConverter::convertPackagepage()</a>	224
Converts package page and sets its package as used in <a href="#">\$package_pages</a>	
<a href="#">HTMLframesConverter::convertMethod()</a>	224
Converts class methods for template output	
<a href="#">HTMLframesConverter::\$leftindex</a>	220
indexes of elements by package that need to be generated	
<a href="#">HTMLframesConverter::\$juststarted</a>	220
controls formatting of parser informative output	
<a href="#">HTMLframesConverter::\$name</a>	220
<a href="#">HTMLframesConverter::\$outputformat</a>	220
<a href="#">HTMLframesConverter::\$package_pages</a>	220
array of converted package page names.	
<a href="#">HTMLframesConverter::\$currentclass</a>	219
contains all of the template class element loop data needed for the current template	
<a href="#">HTMLframesConverter::\$current</a>	219
contains all of the template procedural page element loop data needed for the current template	
<a href="#">HTMLSmartyConverter.inc</a>	218
HTML output converter for Smarty Template.	
<a href="#">HTMLframesConverter</a>	219
HTML output converter.	
<a href="#">HTMLframesConverter::\$base_dir</a>	219
target directory passed on the command-line.	
<a href="#">HTMLframesConverter::\$class_dir</a>	219
output directory for the current class being processed	
<a href="#">HTMLframesConverter::\$page_dir</a>	220
output directory for the current procedural page being processed	
<a href="#">HTMLframesConverter::\$processSpecialRoots</a>	221
This converter knows about the new root tree processing	
<a href="#">HTMLframesConverter::ConvertErrorLog()</a>	223
Create errors.html template file output	

<a href="#">HTMLframesConverter::convertFunction()</a>	Converts function for template output	223
<a href="#">HTMLframesConverter::convertGlobal()</a>	Converts global variables for template output	223
<a href="#">HTMLframesConverter::convertInclude()</a>	Converts include elements for template output	224
<a href="#">HTMLframesConverter::convertDefine()</a>	Converts defines for template output	222
<a href="#">HTMLframesConverter::convertConst()</a>	Converts class variables for template output	222
<a href="#">HTMLframesConverter::\$ric_set</a>		221
<a href="#">HTMLframesConverter::\$sort_page_contents_by_type</a>	Smarty Converter wants elements sorted by type as well as alphabetically	221
<a href="#">HTMLframesConverter::\$wrote</a>		221
<a href="#">HTMLframesConverter::convertClass()</a>	Converts class for template output	222
<a href="#">HTMLframesConverter::getFunctionLink()</a>		231
<a href="#">HTMLframesConverter::getGlobalLink()</a>		231
<a href="#">HTMLSmartyConverter::\$juststarted</a>	controls formatting of parser informative output	239
<a href="#">HTMLSmartyConverter::\$currentclass</a>	contains all of the template class element loop data needed for the current template	239
<a href="#">HTMLSmartyConverter::\$leftindex</a>	indexes of elements by package that need to be generated	239
<a href="#">HTMLSmartyConverter::\$name</a>		239
<a href="#">HTMLSmartyConverter::\$outputformat</a>		239
<a href="#">HTMLSmartyConverter::\$current</a>	contains all of the template procedural page element loop data needed for the current template	239
<a href="#">HTMLSmartyConverter::\$class_dir</a>	output directory for the current class being processed	239
<a href="#">HTMLframesConverter::writeNewPPage()</a>		237
<a href="#">HTMLframesConverter::writeSource()</a>		238
<a href="#">HTMLSmartyConverter</a>	HTML output converter.	238
<a href="#">HTMLSmartyConverter::\$base_dir</a>	target directory passed on the command-line.	238
<a href="#">HTMLSmartyConverter::\$package_pages</a>	array of converted package page names.	239
<a href="#">HTMLSmartyConverter::\$page_dir</a>	output directory for the current procedural page being processed	240
<a href="#">HTMLSmartyConverter::convertDefine()</a>	Converts defines for template output	241
<a href="#">HTMLSmartyConverter::ConvertErrorLog()</a>	Create errors.html template file output	242
<a href="#">HTMLSmartyConverter::convertFunction()</a>	Converts function for template output	242
<a href="#">HTMLSmartyConverter::convertGlobal()</a>	Converts global variables for template output	242
<a href="#">HTMLSmartyConverter::convertConst()</a>	Converts class variables for template output	241
<a href="#">HTMLSmartyConverter::convertClass()</a>	Converts class for template output	241

<a href="#">HTMLSmartyConverter::\$processSpecialRoots</a>	240
<i>This converter knows about the new root tree processing</i>	
<a href="#">HTMLSmartyConverter::\$ric_set</a>	240
<a href="#">HTMLSmartyConverter::\$sort_page_contents_by_type</a>	240
<i>Smarty Converter wants elements sorted by type as well as alphabetically</i>	
<a href="#">HTMLSmartyConverter::\$wrote</a>	240
<a href="#">HTMLframesConverter::writeExample()</a>	237
<a href="#">HTMLframesConverter::unmangle()</a>	237
<a href="#">HTMLframesConverter::getSourceLink()</a>	234
<a href="#">HTMLframesConverter::getSourceAnchor()</a>	233
<i>Retrieve a Converter-specific anchor to a segment of a source code file         parsed via a <a href="#">@filesource</a> tag.</i>	
<a href="#">HTMLframesConverter::getTutorialId()</a>	234
<a href="#">HTMLframesConverter::getTutorialTree()</a>	234
<a href="#">HTMLframesConverter::getVarLink()</a>	234
<a href="#">HTMLframesConverter::getRootTree()</a>	233
<i>return formatted class tree for the Class Trees page</i>	
<a href="#">HTMLframesConverter::getPageName()</a>	233
<a href="#">HTMLframesConverter::getId()</a>	232
<a href="#">HTMLframesConverter::getIndexInformation()</a>	232
<i>Generate indexing information for given element</i>	
<a href="#">HTMLframesConverter::getMethodLink()</a>	232
<a href="#">HTMLframesConverter::getPageLink()</a>	232
<a href="#">HTMLframesConverter::makeLeft()</a>	235
<a href="#">HTMLframesConverter::Output()</a>	235
<i>This function is not used by HTMLdefaultConverter, but is required by Converter</i>	
<a href="#">HTMLframesConverter::setTargetDir()</a>	236
<i>calls the converter setTargetDir, and then copies any template images and the stylesheet         if they haven't been copied</i>	
<a href="#">HTMLframesConverter::SmartyInit()</a>	236
<a href="#">HTMLframesConverter::sourceLine()</a>	237
<i>Return a line of highlighted source code with formatted line number</i>	
<a href="#">HTMLframesConverter::TutorialExample()</a>	237
<a href="#">HTMLframesConverter::returnSee()</a>	236
<i>This function takes an <a href="#">abstractLink</a> descendant and returns an html link</i>	
<a href="#">HTMLframesConverter::returnLink()</a>	236
<a href="#">HTMLframesConverter::postProcess()</a>	235
<i>Uses htmlspecialchars() on the input</i>	
<a href="#">HTMLframesConverter::ProgramExample()</a>	235
<i>Used to convert the &lt;&gt;code&lt;&gt; tag in a docblock</i>	
<a href="#">HTMLframesConverter::rcNatCmp()</a>	235
<i>does a nat case sort on the specified second level value of the array</i>	
<a href="#">HTMLframesConverter::rcNatCmp1()</a>	235
<i>does a nat case sort on the specified second level value of the array.</i>	
<a href="#">HTMLframesConverter.inc</a>	217
<i>HTML original framed output converter, modified to use Smarty Template.</i>	

<a href="#">Io::\$ignore</a>	669
<i>Format: array(array(regexp-ready string to search for whole path,</i>	
<a href="#">Io</a>	668

<i>Class to handle file and user io operations</i>	
<a href="#">IntermediateParserTests.php</a>	656
<a href="#">Io::\$phpDocOptions</a>	669
<i>Holds all the options that are available to the cmd line interface</i>	
<a href="#">Io::\$valid_booleans</a>	669
<i>A specific array of values that boolean-based arguments can understand, aided by the <a href="#">decideOnOrOff()</a> helper method.</i>	
<a href="#">Io::dirList()</a>	670
<a href="#">Io::checkIgnore()</a>	670
<i>Tell whether to ignore a file or a directory allows * and ? wildcards</i>	
<a href="#">Ioinc_sortfiles()</a>	641
<i>Sorting functions for the file list</i>	
<a href="#">Ioinc_mystrcsort()</a>	640
<i>Sorting functions for the file list</i>	
<a href="#">IParser</a>	547
<a href="#">iNewRender::\$vars</a>	547
<i>array of class variables by package, subpackage and class</i>	
<a href="#">iNewRender::\$targetDir</a>	546
<i>used to set the output directory</i>	
<a href="#">iparserElement</a>	547
<a href="#">IntermediateParserAddPrivatePageTests.php</a>	557
<i>Unit Tests for the IntermediateParser-&gt;addPrivatePage() method</i>	
<a href="#">Io.inc</a>	640
<i>File and input handling routines</i>	
<a href="#">IntermediateParser.inc</a>	639
<i>The phpDocumentor_IntermediateParser Class</i>	
<a href="#">Io::displayHelpMsg()</a>	670
<i>create the help message for display on the command-line</i>	
<a href="#">Io::getAllFiles()</a>	670
<i>Take a filename with wildcards and return all files that match the</i>	
<a href="#">InlineTags.inc</a>	773
<i>All abstract representations of inline tags are in this file</i>	
<a href="#">Io:: setupIgnore()</a>	674
<i>Construct the <a href="#">\$ignore</a> array</i>	
<a href="#">Io::removeNonMatches()</a>	673
<i>Removes files from the \$dir array that do not match the search string in</i>	
<a href="#">INSTALL</a>	952
<a href="#">IntermediateParser.inc</a>	1212
<i>Source code</i>	
<a href="#">InlineTags.inc</a>	1292
<i>Source code</i>	
<a href="#">Io.inc</a>	1238
<i>Source code</i>	
<a href="#">Io::readPhpFile()</a>	673
<i>Reads a file and returns it as a string</i>	
<i>Does basic error checking</i>	
<a href="#">Io::parseArgv()</a>	673
<i>Parses \$_SERVER['argv'] and creates a setup array</i>	
<a href="#">Io::getDirTree()</a>	671
<a href="#">Io::getBase()</a>	671
<i>Retrieve common directory (case-insensitive in windows)</i>	
<a href="#">Io::getReadmeInstallChangelog()</a>	672

<a href="#">Io::getRegExableSearchString()</a>	672
Converts \$s into a string that can be used with preg_match	
<a href="#">Io::isIncludeable()</a>	672
calls <a href="http://www.php.net/file_exists">http://www.php.net/file_exists</a> for each value in include_path,	
<a href="#">Io::getTutorials()</a>	672
Retrieve tutorial subdirectories and their contents from the list of	
<a href="#">iNewRender::\$quietMode</a>	546
set in <a href="#">phppdoc.inc</a> to the value of the quitemode commandline option.	
<a href="#">iNewRender::\$private_class</a>	546
this variable is used to prevent parsing of private elements if \$parsePrivate is false.	
<a href="#">iHTMLConverter::Convert()</a>	541
<a href="#">iHTMLConverter</a>	540
Base class for all output converters.	
<a href="#">iConverter::walk()</a>	540
<a href="#">iiparserBase</a>	541
<a href="#">iiparserBase::\$type</a>	542
always base	
<a href="#">iiparserBase::getType()</a>	542
<a href="#">iiparserBase::\$value</a>	542
set to different things by its descendants	
<a href="#">iConverter</a>	539
Base class for all output converters.	
<a href="#">inline {@tutorial}</a>	111
<a href="#">inline {@internal}}</a>	101
<a href="#">inline {@id}</a>	99
<a href="#">inline {@inheritdoc}</a>	102
<a href="#">inline {@link}</a>	104
<a href="#">inline {@toc}</a>	109
<a href="#">inline {@source}</a>	106
<a href="#">iiparserBase::getValue()</a>	542
<a href="#">iiparserBase::setValue()</a>	542
<a href="#">iNewRender::\$methods</a>	545
array of methods by package, subpackage and class	
<a href="#">iNewRender::\$links</a>	544
the workhorse of linking.	
<a href="#">iNewRender::\$event_handlers</a>	544
the functions which handle output from the <a href="#">Parser</a>	
<a href="#">iNewRender::\$packageoutput</a>	545
array of packages to parser and output documentation for, if not all packages should be documented	
<a href="#">iNewRender::\$pages</a>	545
<a href="#">iNewRender::\$pkg_elements</a>	546
<a href="#">iNewRender::\$parsePrivate</a>	546
set in <a href="#">phppdoc.inc</a> to the value of the parserprivate commandline option.	
<a href="#">iNewRender::\$elements</a>	544
<a href="#">iNewRender::\$data</a>	544
data contains parsed structures for the current page being parsed	
<a href="#">iNewRender::\$classpackage</a>	542
set to the name of the package of the current class being parsed	
<a href="#">iNewRender</a>	542
<a href="#">iNewRender::\$classpackages</a>	543
used in Converter::getClassPackage() to inherit package from parent classes.	
<a href="#">iNewRender::\$classsubpackage</a>	543

*set to the name of the subpackage of the current class being parsed*

[iNewRender::\\$class\\_children](#) . . . . . 543

*An array of extended classes by package and parent class*

*Format:*

*array(packagename => array(parentclass => array(childclassname1,  
childclassname2,...*

[iNewRender::\\$classtree](#) . . . . . 543

*a tree of class inheritance by name.*

[inline {@example}](#) . . . . . 97

## K

[kiddie\\_b587733](#) . . . . . 548

## L

[LinkClasses.inc](#) . . . . . 1306

*Source code*

[LinkClasses.inc](#) . . . . . 788

*Linking to element documentation is performed by the classes in this file.*

## M

[MLIB\\_LOG\\_FILE](#) . . . . . 509

*Default log file*

[MLIB\\_SYSLOG\\_PRIORITY](#) . . . . . 509

*Syslog priority*

[MLIB\\_TEMPLATE\\_PATH](#) . . . . . 509

*Template path*

[MLIB\\_USE\\_SYSLOG](#) . . . . . 510

*Use syslog*

[MLIB\\_INCLUDE\\_PATH](#) . . . . . 509

*MLib include path*

[MLIB\\_GLOBAL\\_DEBUG](#) . . . . . 508

*Global debugging*

[modifier.truncate.php](#) . . . . . 478

*Smarty plugin*

[modifier.upper.php](#) . . . . . 479

*Smarty plugin*

[modifier.wordwrap.php](#) . . . . . 480

*Smarty plugin*

[mama](#) . . . . . 548

[metoo](#) . . . . . 548

[methodLink::\\$class](#) . . . . . 793

[methodLink::\\$type](#) . . . . . 793

[methodLink::addLink\(\)](#) . . . . . 793

*sets up the link*

[methodLink](#) . . . . . 793

*method link*

[multipl::func\(\)](#) . . . . . 549

<a href="#">metoo::\$mine</a>	548
<a href="#">multipl</a>	549
<a href="#">multipl::\$manyvars</a>	549
<a href="#">modifier.strip_tags.php</a>	477
<i>Smarty plugin</i>	
<a href="#">modifier.strip.php</a>	476
<i>Smarty plugin</i>	
<a href="#">modifier.date_format.php</a>	463
<i>Smarty plugin</i>	
<a href="#">modifier.debug_print_var.php</a>	464
<i>Smarty plugin</i>	
<a href="#">modifier.default.php</a>	465
<i>Smarty plugin</i>	
<a href="#">modifier.count_words.php</a>	462
<i>Smarty plugin</i>	
<a href="#">modifier.count_sentences.php</a>	461
<i>Smarty plugin</i>	
<a href="#">modifier.cat.php</a>	458
<i>Smarty plugin</i>	
<a href="#">modifier.count_characters.php</a>	459
<i>Smarty plugin</i>	
<a href="#">modifier.count_paragraphs.php</a>	460
<i>Smarty plugin</i>	
<a href="#">modifier.escape.php</a>	466
<i>Smarty plugin</i>	
<a href="#">modifier.htmlentities.php</a>	467
<a href="#">modifier.replace.php</a>	473
<i>Smarty plugin</i>	
<a href="#">modifier.spacify.php</a>	474
<i>Smarty plugin</i>	
<a href="#">modifier.string_format.php</a>	475
<i>Smarty plugin</i>	
<a href="#">modifier.regex_replace.php</a>	472
<i>Smarty plugin</i>	
<a href="#">modifier.rawurlencode.php</a>	471
<a href="#">modifier.indent.php</a>	468
<i>Smarty plugin</i>	
<a href="#">modifier.lower.php</a>	469
<i>Smarty plugin</i>	
<a href="#">modifier.nl2br.php</a>	470
<i>Smarty plugin</i>	
<a href="#">modifier.capitalize.php</a>	457
<i>Smarty plugin</i>	

## N

<a href="#">new_phpdoc.php</a>	1183
<i>Source code</i>	
<a href="#">new_phpdoc.php</a>	629
<i>Advanced Web Interface to phpDocumentor</i>	
<a href="#">notseen</a>	554

## O

<a href="#">ObjectWordParser::nextIsObjectOrNonNL()</a>	925
<i>Determine if the next word is an inline tag</i>	
<a href="#">ObjectWordParser::setup()</a>	925
<i>Set the word parser to go.</i>	
<a href="#">ObjectWordParser::getWord()</a>	925
<a href="#">ObjectWordParser</a>	924
<i>Like WordParser but designed to handle an array with strings and</i>	
<a href="#">one()</a>	504
<a href="#">one</a>	514
<a href="#">outputfilter.trim whitespace.php</a>	481
<i>Smarty plugin</i>	

## P

<a href="#">parserClass::getInheritedMethods()</a>	832
<a href="#">parserClass::getInheritedConsts()</a>	832
<a href="#">parserClass::getInheritedVars()</a>	832
<a href="#">parserClass::getLink()</a>	833
<i>quick way to link to this element</i>	
<a href="#">parserClass::getMethodNames()</a>	833
<a href="#">parserClass::getMethod()</a>	833
<a href="#">parserClass::getImplements()</a>	832
<a href="#">parserClass::getExtends()</a>	832
<a href="#">parserClass::addTutorial()</a>	831
<a href="#">parserClass::addImplements()</a>	830
<a href="#">parserClass::getChildClassList()</a>	831
<i>returns a list of all child classes of this class</i>	
<a href="#">parserClass::getConflicts()</a>	831
<i>Returns all classes in other packages that have the same name as this class</i>	
<a href="#">parserClass::getConsts()</a>	832
<a href="#">parserClass::getConstNames()</a>	831
<a href="#">parserClass::getMethods()</a>	833
<a href="#">parserClass::getModifiers()</a>	833
<i>Get the PHP5+ modifiers for this class</i>	
<a href="#">parserClass::hasVar()</a>	835
<a href="#">parserClass::hasMethod()</a>	835
<a href="#">parserClass::isInterface()</a>	835
<a href="#">parserClass::setAccessModifiers()</a>	836
<i>Use this method to set access modifiers for a class</i>	
<a href="#">parserClass::setInterface()</a>	836
<i>Use this method to set the type of class to be an interface</i>	
<a href="#">parserClass::setExtends()</a>	836
<a href="#">parserClass::hasConst()</a>	835
<a href="#">parserClass::getVars()</a>	835
<a href="#">parserClass::getParentClassTree()</a>	834
<a href="#">parserClass::getParent()</a>	833
<i>retrieve object that represents the parent class</i>	
<a href="#">parserClass::getSourceLocation()</a>	834
<a href="#">parserClass::getTutorial()</a>	834
<i>Get the associated tutorial for this class, if any</i>	

<a href="#">parserClass::getVarNames()</a>	835
<a href="#">parserClass::getVar()</a>	835
<a href="#">parserClass::\$_implements</a>	830
<a href="#">parserClass::\$type</a>	830
<i>Type is used by many functions to skip the hassle of if <code>phpDocumentor_get_class(\$blah) == 'parserBlah'</code></i>	
<a href="#">parserDocBlock::replaceInheritDoc()</a>	824
<i>Wrapper for <a href="#">parserDesc::replaceInheritDoc()</a></i>	
<a href="#">parserDocBlock::postProcess()</a>	824
<i>Parse out any html tags from doc comments, and make them into</i>	
<a href="#">parserDocBlock::resetParams()</a>	825
<a href="#">parserDocBlock::setDesc()</a>	825
<a href="#">parserDocBlock::setExplicitCategory()</a>	825
<i>Used if this docblock has a @category tag.</i>	
<a href="#">parserDocBlock::setEndLineNumber()</a>	825
<i>Sets the ending line number for the DocBlock</i>	
<a href="#">parserDocBlock::overridePackage()</a>	824
<i>set the element's package to the passed values. Used in <a href="#">phpDocumentor_IntermediateParser</a> to align package of</i>	
<a href="#">parserDocBlock::listTags()</a>	824
<a href="#">parserDocBlock::getSDesc()</a>	823
<a href="#">parserDocBlock::getLineNumber()</a>	823
<i>Retrieve starting line number</i>	
<a href="#">parserDocBlock::getType()</a>	823
<a href="#">parserDocBlock::hasInheritDoc()</a>	823
<i>Wrapper for <a href="#">parserDesc::hasInheritDoc()</a></i>	
<a href="#">parserDocBlock::listProperties()</a>	824
<a href="#">parserDocBlock::listParams()</a>	824
<a href="#">parserDocBlock::setExplicitPackage()</a>	825
<i>Used if this docblock has a @package tag.</i>	
<a href="#">parserDocBlock::setLineNumber()</a>	825
<i>Sets the starting line number for the DocBlock</i>	
<a href="#">parserClass::\$extends</a>	829
<a href="#">parserClass::\$curfile</a>	829
<a href="#">parserClass::\$ignore</a>	829
<i>Used to determine whether a class should be ignored or not. Helps maintain integrity of parsing</i>	
<a href="#">parserClass::\$parent</a>	829
<i>Format: array(file, parent) where parent class is found or false if no parent</i>	
<a href="#">parserClass::\$tutorial</a>	830
<a href="#">parserClass::\$sourceLocation</a>	830
<a href="#">parserClass</a>	828
<a href="#">ParserElements.inc</a>	828
<i>Parser Elements, all classes representing documentable elements</i>	
<a href="#">parserDocBlock::setSource()</a>	826
<i>Passes to <a href="#">parserStringWithInlineTags::setSource()</a></i>	
<a href="#">parserDocBlock::setShortDesc()</a>	826
<i>Set the short description of the DocBlock</i>	
<a href="#">parserDocBlock::updateGlobals()</a>	826
<i>replaces nameless global variables in the <a href="#">\$funcglobals</a> array with their names</i>	
<a href="#">parserDocBlock::updateModifiers()</a>	827
<a href="#">parserDocBlock::updateStatics()</a>	827
<i>replaces nameless static variables in the <a href="#">\$statics</a> array with their names</i>	

<a href="#">parserDocBlock::updateParams()</a>	827
<i>replaces nameless parameters in the \$params array with their names</i>	
<a href="#">parserClass::setModifiers()</a>	836
<i>Set the PHP5+ modifiers for this class</i>	
<a href="#">parserClass::setParent()</a>	836
<a href="#">parserFunction::addStatics()</a>	845
<i>Add all "static \$var, \$var2 = 6" declarations to this function</i>	
<a href="#">parserFunction::addSource()</a>	845
<i>Set the source code. Always array in PHP 4.3.0+</i>	
<a href="#">parserFunction::getConflicts()</a>	845
<i>Returns all functions in other packages that have the same name as this function</i>	
<a href="#">parserFunction::getFunctionCall()</a>	846
<i>Get a human-friendly description of the function call</i>	
<a href="#">parserFunction::getLink()</a>	846
<i>quick way to link to this element</i>	
<a href="#">parserFunction::getIntricateFunctionCall()</a>	846
<i>Like getFunctionCall(), but has no English or pre-determined formatting.</i>	
<a href="#">parserFunction::addParam()</a>	845
<a href="#">parserFunction::addGlobals()</a>	844
<i>Add all "global \$var, \$var2" declarations to this function</i>	
<a href="#">parserFunction::\$params</a>	843
<i>parameters parsed from function definition.</i>	
<a href="#">parserFunction::\$globals</a>	843
<i>global declarations parsed from function definition</i>	
<a href="#">parserFunction::\$returnsreference</a>	844
<i>Function returns a reference to an element, instead of a value</i>	
<a href="#">parserFunction::\$source</a>	844
<a href="#">parserFunction::\$type</a>	844
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'</i>	
<a href="#">parserFunction::\$statics</a>	844
<i>static variable declarations parsed from function definition</i>	
<a href="#">parserFunction::getParam()</a>	846
<a href="#">parserFunction::getReturnsReference()</a>	846
<a href="#">parserGlobal::getConflicts()</a>	847
<i>Returns all global variables in other packages that have the same name as this global variable</i>	
<a href="#">parserGlobal::\$type</a>	847
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'</i>	
<a href="#">parserGlobal::getDataType()</a>	848
<i>Retrieve converter-specific representation of the data type</i>	
<a href="#">parserGlobal::getLink()</a>	848
<i>quick way to link to this element</i>	
<a href="#">parserInclude</a>	849
<a href="#">parserGlobal::setDataType()</a>	848
<i>Sets the name of the global variable's type</i>	
<a href="#">parserGlobal::\$datatype</a>	847
<i>Name of the global's data type</i>	
<a href="#">parserGlobal</a>	847
<a href="#">parserFunction::hasSource()</a>	846
<i>Determine whether the source code has been requested via {@source}</i>	
<a href="#">parserFunction::getSource()</a>	846

<a href="#">parserFunction::listGlobals()</a>	847
<a href="#">parserFunction::listParams()</a>	847
<a href="#">parserFunction::setReturnsReference()</a>	847
sets <a href="#">\$returnsreference</a> to true	
<a href="#">parserFunction::listStatics()</a>	847
<a href="#">parserFunction</a>	843
<a href="#">parserElement::setPath()</a>	842
<a href="#">parserDefine::getConflicts()</a>	839
<i>Returns all defines in other packages that have the same name as this define</i>	
<a href="#">parserDefine::\$type</a>	839
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'</i>	
<a href="#">parserDefine::getLink()</a>	839
<i>quick way to link to this element</i>	
<a href="#">parserElement</a>	840
<i>all elements except <a href="#">parserPackagePage</a> descend from this abstract class</i>	
<a href="#">parserElement::\$docblock</a>	840
<a href="#">parserElement::\$conflicts</a>	840
<a href="#">parserDefine</a>	839
<a href="#">parserConst::getLink()</a>	838
<i>quick way to link to this element</i>	
<a href="#">parserClass::setSourceLocation()</a>	837
<a href="#">parserClass::setParentNoClass()</a>	837
<a href="#">parserConst</a>	837
<a href="#">parserConst::\$class</a>	837
<a href="#">parserConst::getClass()</a>	838
<i>Retrieve the class name</i>	
<a href="#">parserConst::\$type</a>	838
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'</i>	
<a href="#">parserElement::\$endlinenumber</a>	841
<i>line number on file where this element stops</i>	
<a href="#">parserElement::\$file</a>	841
<i>location of this element (filename)</i>	
<a href="#">parserElement::setDocBlock()</a>	842
<a href="#">parserElement::getPath()</a>	841
<a href="#">parserElement::setEndLineNumber()</a>	842
<i>Sets the ending line number of elements</i>	
<a href="#">parserElement::setFile()</a>	842
<a href="#">parserElement::setName()</a>	842
<a href="#">parserElement::setLineNumber()</a>	842
<i>Set starting line number</i>	
<a href="#">parserElement::getPackage()</a>	841
<a href="#">parserElement::getName()</a>	841
<a href="#">parserElement::\$name</a>	841
<i>name of this element, or include type if element is a <a href="#">parserInclude</a></i>	
<a href="#">parserElement::\$linenumber</a>	841
<i>Line number in the source on which this element appears</i>	
<a href="#">parserElement::\$path</a>	841
<i>full path location of this element (filename)</i>	
<a href="#">parserElement::getEndLineNumber()</a>	841
<a href="#">parserElement::getLineNumber()</a>	841
<a href="#">parserElement::getFile()</a>	841

<a href="#">parserDocBlock::getKeyword()</a>	823
<a href="#">parserDocBlock::getExplicitPackage()</a>	823
<i>If the DocBlock has a @package tag, then this returns true</i>	
<a href="#">parserPage::\$parserVersion</a>	805
<a href="#">parserPage::\$packageOutput</a>	804
<i>Used to limit output, contains contents of --packageoutput commandline.</i>	
<a href="#">parserPage::\$path</a>	805
<a href="#">parserPage::\$source</a>	805
<i>Tokenized source code of the file</i>	
<a href="#">parserPage::\$subpackage</a>	805
<a href="#">parserPage::\$sourceLocation</a>	805
<i>relative source location</i>	
<a href="#">parserPage::\$package</a>	804
<a href="#">parserPage::\$origName</a>	804
<i>original phpdoc-safe name (only letters, numbers and _)</i>	
<a href="#">parserPage::\$category</a>	804
<a href="#">parserPage</a>	803
<i>Contains information about a PHP file, used to group procedural elements together.</i>	
<a href="#">parserPage::\$file</a>	804
<i>filename.ext (no path)</i>	
<a href="#">parserPage::\$id</a>	804
<i>not implemented in this version, will be used to link xml output pages</i>	
<a href="#">parserPage::\$name</a>	804
<i>phpdoc-safe name (only letters, numbers and _)</i>	
<a href="#">parserPage::\$modDate</a>	804
<i>not implemented yet</i>	
<a href="#">parserPage::\$type</a>	805
<i>Type is used by many functions to skip the hassle of if</i>	
<a href="#">parserPage::getFile()</a>	806
<i>gets the file name</i>	
<a href="#">parserPage::setPath()</a>	808
<i>sets the path to the file</i>	
<a href="#">parserPage::setPackageOutput()</a>	807
<i>loads the package output array</i>	
<a href="#">parserPage::setSource()</a>	808
<i>Sets the source code of the file for highlighting.</i>	
<a href="#">parserPage::setSourceLocation()</a>	808
<i>sets the source location</i>	
<a href="#">parserStringWithInlineTags::\$type</a>	809
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'...</i>	
<a href="#">parserStringWithInlineTags</a>	809
<i>Used to represent strings that contain inline tags, so that they can be properly parsed at link time</i>	
<a href="#">parserPage::setName()</a>	807
<i>sets the name</i>	
<a href="#">parserPage::setFile()</a>	807
<i>Sets the name to display in documentation (can be an alias set with @name)</i>	
<a href="#">parserPage::getPackageOutput()</a>	806
<i>gets the package output array</i>	
<a href="#">parserPage::getName()</a>	806
<i>gets the name</i>	

<a href="#">parserPage::getParseData()</a>	. . . . .	806
<i>Not implemented in this version</i>		
<a href="#">parserPage::getPath()</a>	. . . . .	806
<i>gets the path</i>		
<a href="#">parserPage::getType()</a>	. . . . .	807
<i>gets the tag type</i>		
<a href="#">parserPage::getSourceLocation()</a>	. . . . .	806
<i>gets the source location</i>		
<a href="#">parserData::setParent()</a>	. . . . .	803
<i>sets the parent</i>		
<a href="#">parserData::setDocBlock()</a>	. . . . .	803
<i>sets the docblock</i>		
<a href="#">parserData::\$clean</a>	. . . . .	799
<i>used by <a href="#">phpDocumentor\IntermediateParser::handleDocBlock()</a> to</i>		
<a href="#">parserData::\$classelements</a>	. . . . .	799
<i>array of parsed class elements</i>		
<a href="#">parserData::\$docblock</a>	. . . . .	800
<i>DocBlock (<a href="#">parserDocBlock</a>) for this page, or false if not set</i>		
<a href="#">parserData::\$elements</a>	. . . . .	800
<i>array of parsed elements</i>		
<a href="#">parserData::\$parent</a>	. . . . .	800
<i><a href="#">parserPage</a> element that is this parserData's parent, or false if not set.</i>		
<a href="#">parserData::\$links</a>	. . . . .	800
<i>array of links descended from <a href="#">abstractLink</a></i>		
<a href="#">parserData</a>	. . . . .	799
<i>Contains an in-memory representation of all documentable elements (<a href="#">parserPage</a>, <a href="#">parserFunction</a>, <a href="#">parserDefine</a>, <a href="#">parserInclude</a>, <a href="#">parserClass</a>, <a href="#">parserMethod</a>, <a href="#">parserVar</a>) and their DocBlocks (<a href="#">parserDocBlock</a>).</i>		
<a href="#">parserBase::setValue()</a>	. . . . .	798
<i>sets the given value</i>		
<a href="#">parserBase</a>	. . . . .	797
<i>Base class for all elements</i>		
<a href="#">ParserData.inc</a>	. . . . .	797
<i>Parser Data Structures</i>		
<a href="#">parserBase::\$type</a>	. . . . .	798
<i>Type is used by many functions to skip the hassle of if</i>		
<a href="#">parserBase::\$value</a>	. . . . .	798
<i>set to different things by its descendants</i>		
<a href="#">parserBase::getValue()</a>	. . . . .	798
<i>gets the value</i>		
<a href="#">parserBase::getType()</a>	. . . . .	798
<i>gets the type</i>		
<a href="#">parserData::\$privateclasselements</a>	. . . . .	800
<i>array of parsed class elements with @access private</i>		
<a href="#">parserData::\$privateelements</a>	. . . . .	800
<i>array of parsed elements with @access private</i>		
<a href="#">parserData::getTutorial()</a>	. . . . .	802
<i>If this file has a tutorial associated with it, returns a link to the tutorial.</i>		
<a href="#">parserData::getName()</a>	. . . . .	802
<i>Get the output-safe filename (. changed to _)</i>		

<a href="#">parserData::hasClasses()</a>	802
<i>Does this package have classes?</i>	
<a href="#">parserData::hasExplicitDocBlock()</a>	802
<i>If the page-level DocBlock was present in the source, returns true</i>	
<a href="#">parserData::isClean()</a>	802
<i>checks if the element is "cleaned" already</i>	
<a href="#">parserData::hasInterfaces()</a>	802
<i>Does this package have interfaces?</i>	
<a href="#">parserData::getLink()</a>	802
<i>returns a link</i>	
<a href="#">parserData::getClasses()</a>	802
<i>returns a list of all classes declared in a file</i>	
<a href="#">parserData::\$type</a>	801
<i>Type is used by many functions to skip the hassle of if</i>	
<a href="#">parserData::\$Tutorial</a>	801
<a href="#">parserData::addElement()</a>	801
<i>add a new element to the tracking array</i>	
<a href="#">parserData::addLink()</a>	801
<i>adds a link</i>	
<a href="#">parserData::explicitDocBlock()</a>	802
<i>Tells this page that its DocBlock was not implicit</i>	
<a href="#">parserData::addTutorial()</a>	801
<i>adds a tutorial parser</i>	
<a href="#">parserStringWithInlineTags::\$value</a>	809
<i>array of strings and <a href="#">parserInlineTags</a></i>	
<a href="#">parserStringWithInlineTags::add()</a>	809
<i>equivalent to the . operator (\$a = \$b . \$c)</i>	
<a href="#">parserDocBlock::addKeyword()</a>	819
<a href="#">parserDocBlock::addFuncGlobal()</a>	818
<i>adds a function declaration of @global to the <a href="#">\$funcglobals</a> array</i>	
<a href="#">parserDocBlock::addLink()</a>	819
<i>creates a <a href="#">parserLinkTag</a> and adds it to the <a href="#">\$tags</a> array</i>	
<a href="#">parserDocBlock::addName()</a>	819
<i>Adds a @name tag to the tag list</i>	
<a href="#">parserDocBlock::addParam()</a>	820
<a href="#">parserDocBlock::addPackage()</a>	819
<a href="#">parserDocBlock::addFileSource()</a>	818
<i>Adds a new @filesource tag to the DocBlock</i>	
<a href="#">parserDocBlock::addExample()</a>	818
<i>adds an @example tag</i>	
<a href="#">parserDocBlock::\$subpackagedescrip</a>	817
<i>description of subpackage parsed from @package tag</i>	
<a href="#">parserDocBlock::\$subpackage</a>	817
<a href="#">parserDocBlock::\$tags</a>	817
<i>array of <a href="#">parserTags</a></i>	
<a href="#">parserDocBlock::\$unknowntags</a>	817
<i>array of unrecognized <a href="#">parserTags</a></i>	
<a href="#">parserDocBlock::addAccess()</a>	818
<i>add an @access tag to the <a href="#">tags</a> array</i>	
<a href="#">parserDocBlock::\$var</a>	817
<i>This is either a <a href="#">parserVarTag</a> or false if no var tag is present</i>	
<a href="#">parserDocBlock::addProperty()</a>	820
<i>Adds a @property(-read or -write) or @method magic tag to the DocBlock</i>	

<a href="#">parserDocBlock::addReturn()</a>	820
<i>creates a parserReturnTag and adds it to the \$tags array</i>	
<a href="#">parserDocBlock::changeParam()</a>	822
<a href="#">parserDocBlock::changeGlobal()</a>	822
<a href="#">parserDocBlock::changeStatic()</a>	823
<a href="#">parserDocBlock::getDesc()</a>	823
<a href="#">parserDocBlock::getExplicitCategory()</a>	823
<i>If the DocBlock has a @category tag, then this returns true</i>	
<a href="#">parserDocBlock::getEndLineNumber()</a>	823
<i>Retrieve ending line number</i>	
<a href="#">parserDocBlock::cantSource()</a>	822
<i>Tells the DocBlock it can't have a @filesource tag</i>	
<a href="#">parserDocBlock::canSource()</a>	822
<i>Tells the DocBlock it can have a @filesource tag</i>	
<a href="#">parserDocBlock::addStaticVar()</a>	821
<a href="#">parserDocBlock::addSee()</a>	820
<i>creates a parserLinkTag and adds it to the \$tags array</i>	
<a href="#">parserDocBlock::addTag()</a>	821
<i>Used to insert DocBlock Template tags into a docblock</i>	
<a href="#">parserDocBlock::addUnknownTag()</a>	821
<i>adds an unknown tag to the \$unknown_tags array for use by custom converters</i>	
<a href="#">parserDocBlock::addVar()</a>	822
<i>creates a parserVarTag and adds it to the \$tags array</i>	
<a href="#">parserDocBlock::addUses()</a>	821
<i>Add a @uses tag to the DocBlock</i>	
<a href="#">parserDocBlock::\$statics</a>	817
<i>array of static variable data.</i>	
<a href="#">parserDocBlock::\$sdesc</a>	817
<a href="#">parserDesc::hasInheritDoc()</a>	813
<a href="#">parserDesc::add()</a>	813
<a href="#">parserDesc::hasSource()</a>	813
<a href="#">parserDesc::replaceInheritDoc()</a>	813
<i>replaces {@inheritdoc} with the contents of the parent DocBlock</i>	
<a href="#">parserDocBlock::\$category</a>	814
<a href="#">parserDocBlock</a>	814
<i>Represents a docblock and its components, \$desc, \$sdesc, \$tags, and also \$params for functions</i>	
<a href="#">parserDesc::\$type</a>	813
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'</i>	
<a href="#">parserDesc</a>	812
<i>represents a short or long description in a DocBlock (parserDocBlock)</i>	
<a href="#">parserStringWithInlineTags::getString()</a>	810
<i>return the string unconverted (all inline tags are taken out - this should only be used in pre-parsing to see if any other text is in the string)</i>	
<a href="#">parserStringWithInlineTags::Convert()</a>	809
<i>Use to convert the string to a real string with all inline tags parsed and linked</i>	
<a href="#">parserStringWithInlineTags::hasInlineTag()</a>	810
<i>Determine whether the string contains any inline tags</i>	
<a href="#">parserStringWithInlineTags::setSource()</a>	810
<i>Pass source code to any {@source} tags contained within the string</i>	

<i>for later conversion.</i>	
<a href="#">ParserDocBlock.inc</a>	812
<i>DocBlock Parser Classes</i>	
<a href="#">parserStringWithInlineTags::trimmedStrlen()</a>	811
<i>equivalent to trim(strlen(\$string))</i>	
<a href="#">parserDocBlock::\$desc</a>	814
<a href="#">parserDocBlock::\$linenumber</a>	814
<i>Line number in the source on which this docblock ends</i>	
<a href="#">parserDocBlock::\$params</a>	816
<i>array of param data.</i>	
<a href="#">parserDocBlock::\$packagedescrip</a>	816
<i>description of package parsed from @package tag</i>	
<a href="#">parserDocBlock::\$processed_desc</a>	816
<a href="#">parserDocBlock::\$processed_sdesc</a>	816
<a href="#">parserDocBlock::\$return</a>	816
<i>This is either a <a href="#">parserReturnTag</a> or false if no return tag is present</i>	
<a href="#">parserDocBlock::\$properties</a>	816
<i>array of <a href="#">parserPropertyTag</a>, <a href="#">parserPropertyReadTag</a>, <a href="#">parserPropertyWriteTag</a>, <a href="#">parserMethodTag</a> magic tags</i>	
<a href="#">parserDocBlock::\$package</a>	815
<a href="#">parserDocBlock::\$linenumber</a>	815
<i>Line number in the source on which this docblock begins</i>	
<a href="#">parserDocBlock::\$explictpackage</a>	815
<i>fix for bug 591396</i>	
<a href="#">parserDocBlock::\$explictcategory</a>	814
<i>fix for bug 708559</i>	
<a href="#">parserDocBlock::\$funcglobals</a>	815
<i>array of global variable data.</i>	
<a href="#">parserDocBlock::\$hasaccess</a>	815
<i>whether this DocBlock has an @access tag</i>	
<a href="#">parserDocBlock::\$hasname</a>	815
<i>whether this DocBlock has a @name tag</i>	
<a href="#">parserInclude::\$type</a>	849
<i>Type is used by many functions to skip the hassle of if <code>phpDocumentor_get_class(\$blah) == 'parserBlah'</code></i>	
<a href="#">parserMethod</a>	849
<a href="#">parserDescParser::getParserEventName()</a>	894
<a href="#">parserDescParser::doSimpleList()</a>	893
<i>Return a simple list, if found</i>	
<a href="#">parserDescParser::parse()</a>	894
<i>Parse a long or short description for tags</i>	
<a href="#">parserDescParser::setupStates()</a>	894
<i>setup the parser tokens, and the pushEvent/popEvent arrays</i>	
<a href="#">phpDocumentorTParser::\$eventHandlers</a>	896
<a href="#">phpDocumentorTParser</a>	895
<i>Tokenizer-based parser for PHP source code</i>	
<a href="#">parserDescParser</a>	893
<i>Parses a DocBlock description to retrieve abstract representations of</i>	
<a href="#">Parser::varTagHandler()</a>	892
<i>handles @var</i>	
<a href="#">Parser::propertyTagHandler()</a>	891
<i>Handles @property(-read or -write) and @method magic tag</i>	
<a href="#">Parser::parse()</a>	890

<i>Parse a new file</i>	
<a href="#">Parser::returnTagHandler()</a>	891
<i>handles @return</i>	
<a href="#">Parser::setupStates()</a>	891
<i>setup the parser tokens, and the pushEvent/popEvent arrays</i>	
<a href="#">Parser::usesTagHandler()</a>	892
<i>handles @uses</i>	
<a href="#">Parser::staticvarTagHandler()</a>	892
<i>handles @staticvar</i>	
<a href="#">phpDocumentorTParser::\$inlineTagHandlers</a>	896
<a href="#">phpDocumentorTParser::\$source_location</a>	896
<i>relative path of the parsed file from the base parse directory</i>	
<a href="#">phpDocumentor_HighlightWordParser::getWord()</a>	900
<i>Retrieve the next token</i>	
<a href="#">phpDocumentor_HighlightWordParser::backupPos()</a>	900
<i>back the word parser to the previous token as defined by \$last_token</i>	
<a href="#">phpDocumentor_HighlightWordParser::nextToken()</a>	900
<i>Retrieve the position of the next token that will be parsed in the internal token array</i>	
<a href="#">phpDocumentor_HighlightWordParser::setup()</a>	900
<i>Initialize the parser object</i>	
<a href="#">phpDocumentor_peardoc2_XML_Beautifier::formatFile()</a>	901
<i>format a file or URL</i>	
<a href="#">phpDocumentor_peardoc2_XML_Beautifier</a>	901
<i>This is just like XML_Beautifier, but uses <a href="#">phpDocumentor XML_Beautifier Tokenizer</a></i>	
<a href="#">phpDocumentor_HighlightWordParser</a>	899
<i>Retrieve tokens from an array of tokens organized by line numbers</i>	
<a href="#">phpDocumentor_HighlightParser::setupStates()</a>	899
<i>Initialize all parser state variables</i>	
<a href="#">phpDocumentor_HighlightParser</a>	897
<i>Highlights source code using <a href="#">parse()</a></i>	
<a href="#">phpDocumentorTParser::parse()</a>	896
<i>Parse a new file</i>	
<a href="#">phpDocumentor_HighlightParser::configWordParser()</a>	897
<i>Give the word parser necessary data to begin a new parse</i>	
<a href="#">phpDocumentor_HighlightParser::newLineNum()</a>	898
<i>wraps the current line (via the converter) and resets it to empty</i>	
<a href="#">phpDocumentor_HighlightParser::setLineNum()</a>	899
<i>Start the parsing at a certain line number</i>	
<a href="#">phpDocumentor_HighlightParser::parse()</a>	898
<i>Parse a new file</i>	
<a href="#">Parser::paramTagHandler()</a>	890
<i>handles @param</i>	
<a href="#">Parser::packageTagHandler()</a>	890
<i>handles @package</i>	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_ATTRIBUTES</a>	885
<i>when tag attributes name="value" are found</i>	
<a href="#">phpDocumentorTParser.inc</a>	881
<i>tokenizer extension-based parser for PHP code</i>	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_CDATA</a>	885
<i>when &lt;![CDATA[ ]]&gt; is found</i>	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_ENTITY</a>	886
<i>when tag attributes name="value" are found</i>	

<a href="#">PHPDOCUMENTOR_PDP_EVENT_TAG</a>	886
when a DocBook <tag> is found	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_PROGRAMLISTING</a>	886
when <programlisting> is found	
<a href="#">PHPDOCUMENTOR_EVENT_VAR</a>	876
Class Variable published to IntermediateParser with this event	
<a href="#">PHPDOCUMENTOR_EVENT_TUTORIAL</a>	876
Tutorial published to IntermediateParser with this event	
<a href="#">PHPDOCUMENTOR_EVENT_NEWLINENUM</a>	875
use to inform ErrorTracker of the next line number being parsed	
<a href="#">PHPDOCUMENTOR_EVENT_NEWFILE</a>	875
use to inform ErrorTracker of a new file being parsed	
<a href="#">PHPDOCUMENTOR_EVENT_NEWSTATE</a>	875
use to inform IntermediateParser of a new element being parsed	
<a href="#">PHPDOCUMENTOR_EVENT_PACKAGEPAGE</a>	875
Package-level page published to IntermediateParser with this event	
<a href="#">PHPDOCUMENTOR_EVENT_README_INSTALL_CHANGELOG</a>	875
Contents of README/INSTALL/CHANGELOG files published to IntermediateParser with this event	
<a href="#">PHPDOCUMENTOR_EVENT_PAGE</a>	875
New File (page) published to IntermediateParser with this event	
<a href="#">PHPDOCUMENTOR_PDP_STATE_ATTRIBUTES</a>	886
when tag attributes name="value" are found	
<a href="#">PHPDOCUMENTOR_PDP_STATE_CDATA</a>	886
when <![CDATA[ ]]> is found	
<a href="#">Parser::endTag()</a>	888
Called to clean up at the end of parsing a @tag in a docblock	
<a href="#">Parser::defaultTagHandler()</a>	888
Handles all standard tags that only have a description	
<a href="#">Parser::exampleTagHandler()</a>	888
handles @example	
<a href="#">Parser::getParserEventName()</a>	889
Debugging function, takes an event number and attempts to return its name	
<a href="#">Parser::invalidTagHandler()</a>	889
Handles tags like '@filesource' that only work in PHP 4.3.0+	
<a href="#">Parser::globalTagHandler()</a>	889
handles @global	
<a href="#">Parser::configWordParser()</a>	888
tell the parser's WordParser \$wp to set up tokens to parse words by.	
<a href="#">Parser::checkEventPush()</a>	887
this function checks whether parameter \$word is a token for pushing a new event onto the Event Stack.	
<a href="#">PHPDOCUMENTOR_PDP_STATE_PROGRAMLISTING</a>	886
when <programlisting> is found	
<a href="#">PHPDOCUMENTOR_PDP_STATE_ENTITY</a>	886
when tag attributes name="value" are found	
<a href="#">PHPDOCUMENTOR_PDP_STATE_TAG</a>	886
when a DocBook <tag> is found	
<a href="#">Parser</a>	886
PHP Parser for PHP 4.2.3-	
<a href="#">Parser::checkEventPop()</a>	887
this function checks whether parameter \$word is a token for popping the current event off of the Event Stack.	

<a href="#">Parser::categoryTagHandler()</a>	handles @category	887
<a href="#">phpDocumentor_peardoc2_XML_Beautifier::formatString()</a>	format an XML string	901
<a href="#">phpDocumentor_TutorialHighlightParser</a>	Highlights source code using <a href="#">parse()</a>	902
<a href="#">parserXMLDocBookTag::hasTitle()</a>	Determine whether the docbook element has a title	921
<a href="#">parserXMLDocBookTag::getTOC()</a>	Retrieve either the table of contents index, or the location that the TOC will go	921
<a href="#">parserXMLDocBookTag::setId()</a>	If the id attribute is present, this method will set its id	921
<a href="#">parserXMLDocBookTag::setTitle()</a>	Set the title of a DocBook tag section.	922
<a href="#">parserXMLDocBookTag::startCData()</a>	Begin a new CData section	922
<a href="#">parserXMLDocBookTag::setTOC()</a>	sets the TOC value	922
<a href="#">parserXMLDocBookTag::getTitle()</a>	Retrieve Converter-specific formatting of the title of this element	921
<a href="#">parserXMLDocBookTag::getSubsection()</a>	Retrieve the contents of a subsection	921
<a href="#">parserXMLDocBookTag::addAttribute()</a>	Add an xml tag attribute name="value" pair	919
<a href="#">parserXMLDocBookTag::add()</a>	Add contents to this tag.	919
<a href="#">parserXMLDocBookTag::addCData()</a>	add a word to CData	920
<a href="#">parserXMLDocBookTag::Convert()</a>	calls the output conversion	920
<a href="#">parserXMLDocBookTag::getId()</a>	Return converter-specific formatting of ID.	920
<a href="#">parserXMLDocBookTag::endCData()</a>	Adds \$_cdata to \$value	920
<a href="#">phpDocumentorTWordParser.inc</a>	tokenizer extension-based lexer for PHP code	923
<a href="#">phpDocumentorTWordParser</a>	Like WordParser, but expects an array of tokens from the tokenizer instead of a string.	925
<a href="#">phpDocumentorTWordParser::tokenEquals()</a>	Utility function to determine whether two tokens from the tokenizer are equal	928
<a href="#">phpDocumentorTWordParser::setup()</a>	Uses <a href="http://www.php.net/token_get_all">http://www.php.net/token_get_all</a> to tokenize the source code.	928
<a href="#">ParserPDF.inc</a>	Source code	1118
<a href="#">PDFdefaultConverter.inc</a>	Source code	1126
<a href="#">phpdoc.inc</a>	Source code	1252
<a href="#">phpdoc.php</a>	Source code	1184
<a href="#">phpDocumentorTWordParser::getWord()</a>		928

<i>Retrieve a token for the phpDocumentorTParser</i>	
<a href="#">phpDocumentorTWordParser::getSource()</a>	928
<i>loads up next set of source code</i>	
<a href="#">phpDocumentorTWordParser::addSource()</a>	926
<i>Generate source token arrays organized by line number</i>	
<a href="#">phpDocumentorTWordParser::addFileSource()</a>	926
<i>Wrapper for <a href="#">addSource()</a> used to retrieve the entire source code organized by line number in setup()</i>	
<a href="#">phpDocumentorTWordParser::backupPos()</a>	926
<i>backs the parser up to the previous position</i>	
<a href="#">phpDocumentorTWordParser::concatTokens()</a>	927
<i>Utility function to convert a series of tokens into a string</i>	
<a href="#">phpDocumentorTWordParser::getFileSource()</a>	927
<i>gets the source code tokens</i>	
<a href="#">phpDocumentorTWordParser::findGlobal()</a>	927
<i>Tell the phpDocumentorTWordParser to return the entire global variable if it is found.</i>	
<a href="#">parserXMLDocBookTag::\$name</a>	919
<i>Name of the tag</i>	
<a href="#">parserXMLDocBookTag::\$attributes</a>	918
<i>Attributes from the XML tag</i>	
<a href="#">phpDocumentor XML Beautifier Tokenizer::\$eventHandlers</a>	906
<a href="#">phpDocumentor XML Beautifier Tokenizer</a>	905
<i>Highlights source code using parse()</i>	
<a href="#">phpDocumentor XML Beautifier Tokenizer::checkEventPop()</a>	906
<i>this function checks whether parameter \$word is a token for popping the current event off of the Event Stack.</i>	
<a href="#">phpDocumentor XML Beautifier Tokenizer::checkEventPush()</a>	906
<i>this function checks whether parameter \$word is a token for pushing a new event onto the Event Stack.</i>	
<a href="#">phpDocumentor XML Beautifier Tokenizer::getParserEventName()</a>	907
<a href="#">phpDocumentor XML Beautifier Tokenizer::configWordParser()</a>	907
<i>tell the parser's WordParser \$wp to set up tokens to parse words by.</i>	
<a href="#">phpDocumentor TutorialHighlightParser::setupStates()</a>	905
<i>Initialize all parser state variables</i>	
<a href="#">phpDocumentor TutorialHighlightParser::setLineNum()</a>	905
<i>Start the parsing at a certain line number</i>	
<a href="#">phpDocumentor TutorialHighlightParser::checkEventPush()</a>	903
<i>This function checks whether parameter \$word is a token for pushing a new event onto the Event Stack.</i>	
<a href="#">phpDocumentor TutorialHighlightParser::checkEventPop()</a>	902
<i>This function checks whether parameter \$word is a token for popping the current event off of the Event Stack.</i>	
<a href="#">phpDocumentor TutorialHighlightParser::configWordParser()</a>	903
<i>Tell the parser's WordParser \$wp to set up tokens to parse words by.</i>	
<a href="#">phpDocumentor TutorialHighlightParser::getParserEventName()</a>	904
<i>searches for a parser event name based on its number</i>	
<a href="#">phpDocumentor TutorialHighlightParser::parse()</a>	904
<i>Parse a new file</i>	
<a href="#">phpDocumentor TutorialHighlightParser::newLineNum()</a>	904
<i>advances output to a new line</i>	
<a href="#">phpDocumentor XML Beautifier Tokenizer::incdataHandler()</a>	907
<i>Handler for real character data</i>	

<a href="#">phpDocumentor XML Beautifier Tokenizer::parseString()</a>	907
Parse a new file	
<a href="#">parserCData</a>	916
Represents <![CDATA[ ]]> sections.	
<a href="#">PackagePageElements.inc</a>	915
Data structures used in parsing XML DocBook-based tutorials	
<a href="#">parserCData::Convert()</a>	916
calls the output conversion	
<a href="#">parserEntity</a>	917
a standard entity like &rdquo;	
<a href="#">parserXMLDocBookTag</a>	918
a standard XML DocBook Tag	
<a href="#">parserEntity::Convert()</a>	917
calls the output conversion	
<a href="#">ppageParser::setupStates()</a>	910
set up invariant Parser variables	
<a href="#">ppageParser::parse()</a>	910
Parse a new file	
<a href="#">ppageParser</a>	909
Global package page parser	
<a href="#">phpDocumentor XML Beautifier Tokenizer::setupStates()</a>	908
Initialize all parser state variables	
<a href="#">ppageParser::\$package</a>	909
<a href="#">ppageParser::\$subpackage</a>	909
<a href="#">ppageParser::handleInlineDockeyword()</a>	910
handler for INLINE_DOCKEYWORD.	
<a href="#">ppageParser::defaultHandler()</a>	909
Handles all non-inline tags	
<a href="#">PHPDOCUMENTOR_EVENT_MESSAGE</a>	875
<a href="#">PHPDOCUMENTOR_EVENT_INCLUDE</a>	875
Include (include/require/include_once/include_once) published to IntermediateParser with this event	
<a href="#">parserTutorial::isChildOf()</a>	858
Determine if this parserTutorial object is a child of another	
<a href="#">parserTutorial::getTitle()</a>	858
Retrieve the title of the tutorial, or of any subsection	
<a href="#">parserTutorial::setNext()</a>	858
<a href="#">parserTutorial::setParent()</a>	859
<a href="#">parserVar</a>	860
<a href="#">parserTutorial::setPrev()</a>	859
<a href="#">parserTutorial::getPrev()</a>	858
Retrieve converter-specific link to the previous tutorial's documentation	
<a href="#">parserTutorial::getParent()</a>	857
Retrieve converter-specific link to the parent tutorial's documentation	
<a href="#">parserTutorial::\$type</a>	856
Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'	
<a href="#">parserTutorial::\$Tutorial_type</a>	856
Either cls, pkg, or proc	
<a href="#">parserTutorial::\$_xml</a>	856
<a href="#">parserTutorial::Convert()</a>	857
<a href="#">parserTutorial::getNext()</a>	857
Retrieve converter-specific link to the next tutorial's documentation	

<a href="#">parserTutorial::getLink()</a>	857
Get a link to this tutorial, or to any subsection of this tutorial	
<a href="#">parserVar::\$class</a>	860
<a href="#">parserVar::\$type</a>	860
Type is used by many functions to skip the hassle of if <code>phpDocumentor_get_class(\$blah) == 'parserBlah'</code>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_ATTRIBUTE</a>	864
used when a <code>&lt;tag attr="attribute"&gt;</code> is found	
<a href="#">PHPDOC_BEAUTIFIER_CDATA</a>	864
do not remove, needed in plain renderer	
<a href="#">PHPDOC_XMLTOKEN_EVENT_CDATA</a>	865
used when a <code>&lt;!&gt;</code> is found	
<a href="#">PHPDOC_XMLTOKEN_EVENT_CLOSETAG</a>	865
used when a close <code>&lt;/tag&gt;</code> is found	
<a href="#">PHPDOC_XMLTOKEN_EVENT_DEF</a>	865
used when a <code>&lt;!&gt;</code> is found	
<a href="#">PHPDOC_XMLTOKEN_EVENT_COMMENT</a>	865
used when a <code>&lt;!-- comment --&gt;</code> is found	
<a href="#">parserVar::setModifiers()</a>	861
Return name of the class that contains this method	
<a href="#">parserVar::getOverridingVarsForClass()</a>	861
<a href="#">parserVar::getClass()</a>	860
Retrieve the class name	
<a href="#">parserVar::\$modifiers</a>	860
<a href="#">parserVar::getLink()</a>	861
quick way to link to this element	
<a href="#">parserVar::getModifiers()</a>	861
Return a list of access modifiers (static/private/etc.)	
<a href="#">parserVar::getOverridingVars()</a>	861
<a href="#">parserVar::getOverrides()</a>	861
<a href="#">parserTutorial::\$prev</a>	856
link to the previous tutorial in a document series, or false if none	
<a href="#">parserTutorial::\$path</a>	856
path to the tutorial page	
<a href="#">parserMethod::getIntricateFunctionCall()</a>	851
<a href="#">parserMethod::getImplements()</a>	851
<a href="#">parserMethod::getLink()</a>	852
quick way to link to this element	
<a href="#">parserMethod::getModifiers()</a>	852
Return name of the class that contains this method	
<a href="#">parserMethod::getOverridingMethods()</a>	852
<a href="#">parserMethod::getOverrides()</a>	852
<a href="#">parserMethod::getFunctionCall()</a>	851
adds "constructor" to start of function call if <code>\$isConstructor</code> is true	
<a href="#">parserMethod::getClass()</a>	851
Return name of the class that contains this method	
<a href="#">parserMethod::\$isConstructor</a>	850
<a href="#">parserMethod::\$class</a>	850
<a href="#">parserMethod::\$isDestructor</a>	850
<a href="#">parserMethod::\$type</a>	850
Type is used by many functions to skip the hassle of if <code>phpDocumentor_get_class(\$blah) == 'parserBlah'</code>	
<a href="#">parserMethod::addParam()</a>	851

<a href="#">parserMethod::\$modifiers</a>	851
<a href="#">parserMethod::getOverridingMethodsForClass()</a>	852
<a href="#">parserMethod::setConstructor()</a>	853
<i>Use this method to tell the parser that this method is the class constructor</i>	
<a href="#">parserTutorial::\$linked_element</a>	855
<i>The documentable element this tutorial is linked to</i>	
<a href="#">parserTutorial::\$ini</a>	854
<i>output from tutorialname.ext.ini</i>	
<a href="#">parserTutorial::\$name</a>	855
<i>filename minus extension of this tutorial (used for @tutorial tag)</i>	
<a href="#">parserTutorial::\$next</a>	855
<i>link to the next tutorial in a document series, or false if none</i>	
<a href="#">parserTutorial::\$parent</a>	855
<i>link to the parent tutorial in a document series, or false if none</i>	
<a href="#">parserTutorial::\$package</a>	855
<a href="#">parserTutorial::\$children</a>	854
<i>links to the child tutorials, or false if none</i>	
<a href="#">parserTutorial</a>	854
<a href="#">parserMethod::setModifiers()</a>	853
<i>Return name of the class that contains this method</i>	
<a href="#">parserMethod::setDestructor()</a>	853
<i>Use this method to tell the parser that this method is the class constructor</i>	
<a href="#">parserPackagePage</a>	853
<a href="#">parserPackagePage::\$package</a>	853
<a href="#">parserPackagePage::Convert()</a>	854
<a href="#">parserPackagePage::\$type</a>	853
<i>Type is used by many functions to skip the hassle of if phpDocumentor_get_class(\$blah) == 'parserBlah'</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_DOUBLEQUOTE</a>	865
<i>used when a &lt;!-- comment --&gt; is found</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_ENTITY</a>	865
<i>used when an &amp;entity; is found</i>	
<a href="#">PARSER_EVENT_NOEVENTS</a>	873
<i>used for the beginning of parsing, before first &lt; ? php encountered</i>	
<a href="#">PARSER_EVENT_METHOD_LOGICBLOCK</a>	873
<i>used by the HighlightParser only, when a method body is parsed</i>	
<a href="#">PARSER_EVENT_OUTPHP</a>	873
<i>used when a ? &gt; (with no space) is encountered in parsing</i>	
<a href="#">PARSER_EVENT_PHPCODE</a>	873
<i>used when php code processor instruction (&lt; ? php) is encountered in parsing</i>	
<a href="#">PARSER_EVENT_QUOTE_VAR</a>	873
<i>used by the HighlightParser only, when {\$var} is encountered in a string</i>	
<a href="#">PARSER_EVENT_QUOTE</a>	873
<i>used when double quotation mark ("') encountered in parsing</i>	
<a href="#">PARSER_EVENT_METHOD</a>	873
<i>used by the HighlightParser only, when a method starts</i>	
<a href="#">PARSER_EVENT_LOGICBLOCK</a>	873
{i encountered in parsing a function or php code}	
<a href="#">PARSER_EVENT_IMPLEMENT</a>	872
<i>used when a class implements interfaces</i>	
<a href="#">PARSER_EVENT_GLOBAL_VALUE</a>	872
<i>used when parsing the default value in a global variable declaration</i>	
<a href="#">PARSER_EVENT_INCLUDE</a>	872

<i>used when an include/require/include_once/include_once statement is encountered in parsing</i>	
<a href="#"><b>PARSER_EVENT_INCLUDE_PARAMS</b></a>	872
<i>used when an opening parenthesis of an include/require/include_once/include_once statement is encountered in parsing</i>	
<a href="#"><b>PARSER_EVENT_INLINE_DOCKEYWORD</b></a>	873
<i>used when an inline {@tag} is encountered in parsing a DocBlock</i>	
<a href="#"><b>PARSER_EVENT_INCLUDE_PARAMS_PARENTHESIS</b></a>	873
<i>used when an inner ( ) is encountered while parsing an include/require/include_once/include_once statement</i>	
<a href="#"><b>PARSER_EVENT_SINGLEQUOTE</b></a>	873
<i>used when a single quote (' ) is encountered in parsing</i>	
<a href="#"><b>PARSER_EVENT_STATIC_VAR</b></a>	873
<i>used when parsing a "static \$var1, \$var2;" declaration in a function</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_DOCBLOCK_TEMPLATE</b></a>	874
<i>used when a docblock template is encountered in the source</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_DOCBLOCK</b></a>	874
<i>DocBlock published to IntermediateParser with this event</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_END_DOCBLOCK_TEMPLATE</b></a>	874
<i>used when a docblock template is encountered in the source</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_END_PAGE</b></a>	874
<i>used to inform phpDocumentor_IntermediateParser that the current file has been completely parsed.</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_GLOBAL</b></a>	875
<i>used when a global variable definition is encountered in the source</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_FUNCTION</b></a>	875
<i>Function published to IntermediateParser with this event</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_DEFINE</b></a>	874
<i>Constant (define) published to IntermediateParser with this event</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_CONST</b></a>	874
<i>Class Constant published to IntermediateParser with this event</i>	
<a href="#"><b>PARSER_EVENT_TAGS</b></a>	874
<i>used when parsing the @tag block of a docblock</i>	
<a href="#"><b>PARSER_EVENT_STATIC_VAR_VALUE</b></a>	874
<i>used when parsing the value in a "static \$var1 = x" declaration in a function</i>	
<a href="#"><b>PARSER_EVENT_VAR</b></a>	874
<i>used when a var statement is encountered in parsing a class definition</i>	
<a href="#"><b>PARSER_EVENT_VAR_ARRAY</b></a>	874
<i>used when a variable value is an array</i>	
<a href="#"><b>PHPDOCUMENTOR_EVENT_CLASS</b></a>	874
<i>Class published to IntermediateParser with this event</i>	
<a href="#"><b>PARSER_EVENT_VAR_ARRAY_COMMENT</b></a>	874
<i>used when a comment is found in a variable array value</i>	
<a href="#"><b>PARSER_EVENT_FUNC_GLOBAL</b></a>	872
<i>used when parsing a "global \$var1, \$var2;" declaration in a function</i>	
<a href="#"><b>PARSER_EVENT_FUNCTION_PARAM_VAR</b></a>	872
<i>used when a \$param is encountered in a function definition</i>	
<a href="#"><b>PARSER_EVENT_CLASS</b></a>	870
<i>used when a class definition is encountered in parsing</i>	
<a href="#"><b>PARSER_EVENT_ARRAY</b></a>	870
<i>used when an array definition is encountered in parsing</i>	
<a href="#"><b>PARSER_EVENT_CLASS_CONSTANT</b></a>	871
<i>used when a class implements interfaces</i>	

<a href="#">PARSER_EVENT_CLASS_MEMBER</a>	871
<i>used by the HighlightParser only, when -&gt;var or -&gt;function() is encountered in a method</i>	
<a href="#">PARSER_EVENT_COMMENTBLOCK</a>	871
<i>used when long comment /x x/ where x is an asterisk is encountered in parsing</i>	
<a href="#">PARSER_EVENT_COMMENT</a>	871
<i>used when short comment // is encountered in parsing</i>	
<a href="#">PARSER_EVENT_ACCESS_MODIFIER</a>	870
<i>used when parsing an access modifier</i>	
<a href="#">Parser.inc</a>	870
<i>Base parser for all parsers</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_NOEVENTS</a>	865
<i>starting state</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_IN_CDATA</a>	865
<i>used when a &lt;![CDATA[ section is found</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_OPENTAG</a>	865
<i>used when an open &lt;tag&gt; is found</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_PI</a>	865
<i>used when a processor instruction is found</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_XML</a>	865
<i>used when a &lt;?xml is found</i>	
<a href="#">PHPDOC_XMLTOKEN_EVENT_SINGLEQUOTE</a>	865
<i>used when a &lt;!-- comment --&gt; is found</i>	
<a href="#">PARSER_EVENT_DEFINE</a>	871
<i>used when a define statement is encountered in parsing</i>	
<a href="#">PARSER_EVENT_DEFINE_GLOBAL</a>	871
<i>used when parsing a global variable declaration</i>	
<a href="#">PARSER_EVENT_END_STATEMENT</a>	872
<a href="#">PARSER_EVENT_END_DOCBLOCK_TEMPLATE</a>	872
<i>used when encountering a /**#@-*/ comment (no space) marking the end of using a docblock template</i>	
<a href="#">PARSER_EVENT_EOFQUOTE</a>	872
<i>used when a &lt;&lt;&lt; is encountered in parsing</i>	
<a href="#">PARSER_EVENT_ESCAPE</a>	872
<i>used when a backslash is encountered in parsing a string or other escapable entity</i>	
<a href="#">PARSER_EVENT_FUNCTION_PARAMS</a>	872
<i>used when a function statement opening parenthesis is encountered in parsing</i>	
<a href="#">PARSER_EVENT_FUNCTION</a>	872
<i>used when a function definition is encountered in parsing</i>	
<a href="#">PARSER_EVENT_DOCKEYWORD_EMAIL</a>	872
<i>used when a &lt;email@address&gt; is encountered in parsing an @author tag</i>	
<a href="#">PARSER_EVENT_DOCKEYWORD</a>	871
<i>used when a @tag is encountered in DocBlock parsing</i>	
<a href="#">PARSER_EVENT_DEFINE_PARAMS_PARENTHESIS</a>	871
<i>used when a define statement's opening parenthesis is encountered in parsing</i>	
<a href="#">PARSER_EVENT_DEFINE_PARAMS</a>	871
<i>used when a define statement opening parenthesis is encountered in parsing</i>	
<a href="#">PARSER_EVENT_DESC</a>	871
<i>used when parsing the desc part of a docblock</i>	
<a href="#">PARSER_EVENT_DOCBLOCK</a>	871
<i>used when a DocBlock is encountered in parsing</i>	
<a href="#">PARSER_EVENT_DOCBLOCK_TEMPLATE</a>	871
<i>used when encountering a /**#@+ comment marking a new docblock template</i>	

<a href="#">pageLink::\$type</a>	794
<a href="#">pageLink</a>	794
<i>procedural page link</i>	
<a href="#">phpDocumentorTParserTests.php</a>	660
<a href="#">phpDocumentorSetupTests.php</a>	659
<a href="#">PHPUnit_MAIN_METHOD</a>	660
<a href="#">phpDocumentor_IntermediateParser</a>	674
<i>The phpDocumentor_IntermediateParser Class</i>	
<a href="#">phpDocumentor_IntermediateParser::\$classes</a>	675
<i>used to keep track of inheritance at the smartest level possible for a</i>	
<a href="#">phpDocumentor_IntermediateParser::\$all_packages</a>	675
<i>list of all packages encountered while documenting. Used in automatic linking.</i>	
<a href="#">ParserPageTests.php</a>	658
<a href="#">ParserClassTests.php</a>	657
<a href="#">PHEDOCUMENTOR_PDP_STATE_VAR</a>	647
<i>when &lt;&gt;var&gt; is found in a desc</i>	
<a href="#">PHEDOCUMENTOR_PDP_STATE_SIMLIST</a>	647
<i>when a simple list is found in a desc</i>	
<a href="#">phpdoc.inc</a>	648
<i>startup file</i>	
<a href="#">ProceduralPages.inc</a>	649
<i>Intermediate procedural page parsing structure.</i>	
<a href="#">phpDocumentor_out()</a>	654
<i>Print parse information if quiet setting is off</i>	
<a href="#">Publisher.inc</a>	650
<i>a class for handling the publishing of data</i>	
<a href="#">phpDocumentor_IntermediateParser::\$converters</a>	675
<i>an array of template names indexed by converter name</i>	
<a href="#">phpDocumentor_IntermediateParser::\$cur_class</a>	676
<i>Name of the class currently being parsed.</i>	
<a href="#">phpDocumentor_IntermediateParser::\$pages</a>	679
<a href="#">phpDocumentor_IntermediateParser::\$package_parents</a>	678
<i>Keeps track of packages of classes that have parent classes in another package. Used in automatic linking.</i>	
<a href="#">phpDocumentor_IntermediateParser::\$sparsePrivate</a>	679
<i>set in <a href="#">Setup.inc.php</a> to the value of the parseprivate commandline</i>	
<a href="#">phpDocumentor_IntermediateParser::\$privatepages</a>	679
<i>Put away a page that has been @ignored or @access private if !\$sparsePrivate</i>	
<a href="#">phpDocumentor_IntermediateParser::\$proceduralpages</a>	680
<i>used to keep track of all elements in a procedural page. Handles name</i>	
<a href="#">phpDocumentor_IntermediateParser::\$private_class</a>	679
<i>this variable is used to prevent parsing of elements with an @ignore tag</i>	
<a href="#">phpDocumentor_IntermediateParser::\$package_pages</a>	678
<i>array of parsed package pages</i>	
<a href="#">phpDocumentor_IntermediateParser::\$packageoutput</a>	678
<i>array of packages to parser and output documentation for, if not all packages should be documented</i>	
<a href="#">phpDocumentor_IntermediateParser::\$db_template</a>	676
<a href="#">phpDocumentor_IntermediateParser::\$data</a>	676
<i>\$data contains parsed structures for the current page being parsed</i>	
<a href="#">phpDocumentor_IntermediateParser::\$event_handlers</a>	676

<i>the functions which handle output from the <a href="#">Parser</a></i>	
<a href="#">phpDocumentor_IntermediateParser::\$last</a>	677
<a href="#">phpDocumentor_IntermediateParser::\$packagecategories</a>	677
Used to determine the category for tutorials.	
<a href="#">phpDocumentor_IntermediateParser::\$lasttype</a>	677
type of the last parser Element handled	
<a href="#">PHPDOCUMENTOR_PDP_STATE_SAMP</a>	647
when <><sample>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_STATE_PRE</a>	647
when <><pre>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_ESCAPE_CODE</a>	644
when <></code>> is found in a <><code>><></code>> section	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_ESCAPE</a>	644
when the <> potential escape for tags is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_ESCAPE_PRE</a>	644
when <></pre>> is found in a <><pre>><></pre>> section	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_I</a>	644
when <><i>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_LIST</a>	645
when <><ul>>/<><ol>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_KBD</a>	644
when <><kbd>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_DOUBLECR</a>	644
when \n\n is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_CODE</a>	644
when <><code>> is found in a desc	
<a href="#">PHPDOCUMENTOR_WEBSITE</a>	637
<a href="#">PHPDOCUMENTOR_VER</a>	637
<a href="#">PHPDOCUMENTOR_WINDOWS</a>	637
<a href="#">ParserDescCleanup.inc</a>	643
All of the functions to clean up and handle the long description of a DocBlock are in this file.	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_BR</a>	644
when <> > is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_B</a>	643
when <><b>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_P</a>	645
when <><p>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_EVENT_PRE</a>	645
when <><pre>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_STATE_ESCAPE_PRE</a>	646
when <></pre>> is found in a <><pre>><></pre>> section	
<a href="#">PHPDOCUMENTOR_PDP_STATE_ESCAPE_CODE</a>	646
when <></code>> is found in a <><code>><></code>> section	
<a href="#">PHPDOCUMENTOR_PDP_STATE_I</a>	646
when <><i>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_STATE_KBD</a>	646
when <><kbd>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_STATE_P</a>	646
when <><p>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_STATE_LIST</a>	646
when <><ul>>/<><ol>> is found in a desc	
<a href="#">PHPDOCUMENTOR_PDP_STATE_ESCAPE</a>	646

<i>when the &lt;&lt; potential escape for tags is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_STATE_DOUBLECR</u></a>	646
<i>when \n\n is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_EVENT_SIMLIST</u></a>	645
<i>when a simple list is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_EVENT_SAMP</u></a>	645
<i>when &lt;&lt;samp&gt;&gt; is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_EVENT_VAR</u></a>	645
<i>when &lt;&lt;var&gt;&gt; is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_STATE_B</u></a>	645
<i>when &lt;&lt;b&gt;&gt; is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_STATE_CODE</u></a>	645
<i>when &lt;&lt;code&gt;&gt; is found in a desc</i>	
<a href="#"><u>PHPDOCUMENTOR_PDP_STATE_BR</u></a>	645
<i>when &lt;&lt;br&gt;&gt; is found in a desc</i>	
<a href="#"><u>phpDocumentor_IntermediateParser::\$quietMode</u></a>	680
<i>set in <a href="#"><u>Setup.inc.php</u></a> to the value of the quitemode commandline option.</i>	
<a href="#"><u>phpDocumentor_IntermediateParser::\$ric</u></a>	680
<i>Stores parsed CHANGELOG/INSTALL/README files</i>	
<a href="#"><u>phpDocumentor_setup::\$ignoresymlinks</u></a>	695
<i>contents of --ignoresymlinks commandline</i>	
<a href="#"><u>phpDocumentor_setup::\$hidden</u></a>	695
<i>contents of --hidden commandline</i>	
<a href="#"><u>phpDocumentor_setup::\$ignore_files</u></a>	695
<i>contents of --ignore commandline</i>	
<a href="#"><u>phpDocumentor_setup::\$packages</u></a>	695
<i>Packages to create documentation for</i>	
<a href="#"><u>phpDocumentor_setup::\$render</u></a>	695
<i>Used to organize output from the Parser before Conversion</i>	
<a href="#"><u>phpDocumentor_setup::\$parse</u></a>	695
<i>The main parser</i>	
<a href="#"><u>phpDocumentor_setup::\$files</u></a>	694
<i>contents of --filename commandline</i>	
<a href="#"><u>phpDocumentor_setup::\$dirs</u></a>	694
<i>contents of --directory commandline</i>	
<a href="#"><u>phpDocumentor_IntermediateParser::setTargetDir()</u></a>	692
<i>Sets the output directory</i>	
<a href="#"><u>phpDocumentor_IntermediateParser::setQuietMode()</u></a>	692
<i>set parsing information output mode (quiet or verbose)</i>	
<a href="#"><u>phpDocumentor_IntermediateParser::setTemplateBase()</u></a>	693
<i>Sets the template base directory</i>	
<a href="#"><u>phpDocumentor_IntermediateParser::setUndocumentedElementWarningsMode()</u></a>	693
<i>show warnings for undocumented elements</i>	
<a href="#"><u>phpDocumentor_setup</u></a>	694
<a href="#"><u>phpDocumentor_IntermediateParser::guessPackage()</u></a>	693
<i>Guess the package/subpackage based on subdirectory if the --pear option</i>	
<a href="#"><u>phpDocumentor_setup::\$setup</u></a>	696
<i>Used to parse command-line options</i>	
<a href="#"><u>phpDocumentor_setup::checkIgnoreTag()</u></a>	696
<a href="#"><u>phpDocumentor_setup::setMemoryLimit()</u></a>	697
<i>Allow a memory_limit setting in phpDocumentor.ini to override php.ini or default memory limit</i>	

<a href="#">phpDocumentor_setup::setJavadocDesc()</a>	697
<a href="#">phpDocumentor_setup::setPackageOutput()</a>	698
<a href="#">phpDocumentor_setup::setParsePrivate()</a>	698
<a href="#">phpDocumentor_setup::setTargetDir()</a>	698
<a href="#">phpDocumentor_setup::setQuietMode()</a>	698
<a href="#">phpDocumentor_setup::setIgnore()</a>	697
<a href="#">phpDocumentor_setup::setFilesToParse()</a>	697
<a href="#">phpDocumentor_setup::parseHiddenFiles()</a>	696
<a href="#">phpDocumentor_setup::createDocs()</a>	696
<a href="#">phpDocumentor_setup::parseIni()</a>	696
<i>Parse configuration file phpDocumentor.ini</i>	
<a href="#">phpDocumentor_setup::readCommandLineSettings()</a>	697
<i>Get phpDocumentor settings from command-line or web interface</i>	
<a href="#">phpDocumentor_setup::setDirectoriesToParse()</a>	697
<a href="#">phpDocumentor_setup::readConfigFile()</a>	697
<i>Get phpDocumentor settings from a user configuration file</i>	
<a href="#">phpDocumentor_IntermediateParser::setParsePrivate()</a>	692
<i>set display of elements marked with @access private</i>	
<a href="#">phpDocumentor_IntermediateParser::parsePackagePage()</a>	692
<i>Backward-compatibility only, use the new tutorials for more power</i>	
<a href="#">phpDocumentor_IntermediateParser::addPage()</a>	683
<i>Replaces the <a href="#">parserPage</a> represented by \$this-&gt;pages[\$path] with \$page</i>	
<a href="#">phpDocumentor_IntermediateParser::addPackageParent()</a>	683
<i>If the parent class of \$class is in a different package, adds it to the</i>	
<a href="#">phpDocumentor_IntermediateParser::addPageIfNecessary()</a>	684
<i>add a new <a href="#">parserPage</a> to the \$pages array if none is found</i>	
<a href="#">phpDocumentor_IntermediateParser::addPrivatePage()</a>	684
<i>Adds a <a href="#">parserPage</a> element to the <a href="#">parserData</a> element in \$this-&gt;privatelpages[\$path]</i>	
<a href="#">phpDocumentor_IntermediateParser::ClasselementCmp()</a>	685
<i>does a natural case sort on two class elements (either <a href="#">parserClass</a>, <a href="#">parserMethod</a> or <a href="#">parserVar</a></i>	
<a href="#">phpDocumentor_IntermediateParser::addUses()</a>	684
<i>Add all the @uses tags from \$element to the \$uses array so that @usedby</i>	
<a href="#">phpDocumentor_IntermediateParser::addElementToPage()</a>	683
<i>adds a processed descendant of <a href="#">parserElement</a> to the \$pages array or <a href="#">\$privatelpages</a> array</i>	
<a href="#">phpDocumentor_IntermediateParser::addConverter()</a>	682
<i>Add a converter name to use to the list of converters</i>	
<a href="#">phpDocumentor_IntermediateParser::\$templateBase</a>	681
<i>used to set the template base directory</i>	
<a href="#">phpDocumentor_IntermediateParser::\$targetDir</a>	681
<i>used to set the output directory</i>	
<a href="#">phpDocumentor_IntermediateParser::\$title</a>	681
<a href="#">phpDocumentor_IntermediateParser::\$type</a>	681
<i>type of the current parser Element being handled</i>	
<a href="#">phpDocumentor_IntermediateParser::\$uses</a>	682
<a href="#">phpDocumentor_IntermediateParser::\$undocumentedElementWarnings</a>	682
<i>set in <a href="#">Setup.inc.php</a> to the value of the undocumentedElementWarnings commandline option.</i>	
<a href="#">phpDocumentor_IntermediateParser::Convert()</a>	685
<i>Interface to the Converter</i>	

<a href="#">phpDocumentor_IntermediateParser::elementCmp()</a>	686
<i>does a natural case sort on two <a href="#">parserElement</a> descendants</i>	
<a href="#">phpDocumentor_IntermediateParser::handlePackagePage()</a>	690
<i>handles post-parsing of Package-level documentation pages.</i>	
<a href="#">phpDocumentor_IntermediateParser::handleMethod()</a>	689
<i>handles post-parsing of class methods</i>	
<a href="#">phpDocumentor_IntermediateParser::handlePage()</a>	690
<i>handles post-parsing of procedural pages</i>	
<a href="#">phpDocumentor_IntermediateParser::handleTutorial()</a>	690
<i>handle post-parsing of Tutorials.</i>	
<a href="#">phpDocumentor_IntermediateParser::Output()</a>	691
<i>call this method once parsing has completed.</i>	
<a href="#">phpDocumentor_IntermediateParser::handleVar()</a>	691
<i>handles post-parsing of class vars</i>	
<a href="#">phpDocumentor_IntermediateParser::handleInclude()</a>	689
<i>handles post-parsing of include/require/include_once/require_once</i>	
<a href="#">phpDocumentor_IntermediateParser::handleGlobal()</a>	689
<i>handles post-parsing of global variables</i>	
<a href="#">phpDocumentor_IntermediateParser::handleConst()</a>	686
<i>handles post-parsing of class constants</i>	
<a href="#">phpDocumentor_IntermediateParser::handleClass()</a>	686
<i>handles post-parsing of classes</i>	
<a href="#">phpDocumentor_IntermediateParser::handleDefine()</a>	687
<i>handles post-parsing of defines</i>	
<a href="#">phpDocumentor_IntermediateParser::handleDocBlock()</a>	687
<i>handles post-parsing of DocBlocks</i>	
<a href="#">phpDocumentor_IntermediateParser::handleFunction()</a>	688
<i>handles post-parsing of functions</i>	
<a href="#">phpDocumentor_IntermediateParser::HandleEvent()</a>	688
<i>called via <a href="#">Parser::parse()</a> and Parser's inherited method  <a href="#">Publisher::publishEvent()</a></i>	
<a href="#">phpDocumentor_parse_ini_file()</a>	637
<i>Parse an .ini file</i>	
<a href="#">phpDocumentor_get_class()</a>	636
<i>Returns a lower-cased version of get_class for PHP 5</i>	
<a href="#">PDFdefaultConverter::convertVar()</a>	266
<a href="#">PDFdefaultConverter::convertTutorial()</a>	266
<a href="#">PDFdefaultConverter::Convert_RIC()</a>	266
<i>Convert README/INSTALL/CHANGELOG file contents to output format</i>	
<a href="#">PDFdefaultConverter::generateFormattedClassTrees()</a>	266
<i>returns a template-enabled array of class trees</i>	
<a href="#">PDFdefaultConverter::getClassLink()</a>	267
<a href="#">PDFdefaultConverter::getCDATA()</a>	266
<a href="#">PDFdefaultConverter::convertParams()</a>	265
<a href="#">PDFdefaultConverter::convertPage()</a>	265
<a href="#">PDFdefaultConverter::convertFunction()</a>	264
<a href="#">PDFdefaultConverter::convertDocBlock()</a>	264
<a href="#">PDFdefaultConverter::convertGlobal()</a>	265
<a href="#">PDFdefaultConverter::convertInclude()</a>	265
<a href="#">PDFdefaultConverter::convertPackagepage()</a>	265
<a href="#">PDFdefaultConverter::convertMethod()</a>	265
<a href="#">PDFdefaultConverter::getConstLink()</a>	267
<a href="#">PDFdefaultConverter::getDefineLink()</a>	267

<a href="#">PDFdefaultConverter::getSourceAnchor()</a>	270
<i>Retrieve a Converter-specific anchor to a segment of a source code file parsed via a <code>@filesource</code> tag.</i>	
<a href="#">PDFdefaultConverter::getRootTree()</a>	270
<i>return formatted class tree for the Class Trees page</i>	
<a href="#">PDFdefaultConverter::getSourceLink()</a>	270
<a href="#">PDFdefaultConverter::getState()</a>	270
<a href="#">PDFdefaultConverter::getVarLink()</a>	271
<a href="#">PDFdefaultConverter::getTutorialId()</a>	270
<a href="#">PDFdefaultConverter::getPageName()</a>	270
<a href="#">PDFdefaultConverter::getPageLink()</a>	269
<a href="#">PDFdefaultConverter::getFileSourceName()</a>	268
<a href="#">PDFdefaultConverter::getExampleLink()</a>	268
<a href="#">PDFdefaultConverter::getFunctionLink()</a>	268
<a href="#">PDFdefaultConverter::getGlobalLink()</a>	268
<a href="#">PDFdefaultConverter::getMethodLink()</a>	269
<a href="#">PDFdefaultConverter::getGlobalValue()</a>	269
<a href="#">PDFdefaultConverter::convertDefine()</a>	264
<a href="#">PDFdefaultConverter::convertConst()</a>	264
<a href="#"><b>PHPDOCUMENTOR PDF EVENT TEXT</b></a>	260
<i>when &lt;text&gt; is found in an ezText input</i>	
<a href="#"><b>PHPDOCUMENTOR PDF EVENT PDFFUNCTION</b></a>	260
<i>when &lt;pdffunction&gt; is found in an ezText input</i>	
<a href="#"><b>PHPDOCUMENTOR PDF STATE CONTENT</b></a>	260
<i>used for parsing stuff between &lt;text&gt;</i>	
<a href="#"><b>PHPDOCUMENTOR PDF STATE FONT</b></a>	260
<i>when &lt;font&gt; is found in an ezText input</i>	
<a href="#"><b>PHPDOCUMENTOR PDF STATE PDFFUNCTION</b></a>	260
<i>when &lt;pdffunction&gt; is found in an ezText input</i>	
<a href="#"><b>PHPDOCUMENTOR PDF STATE NEWPAGE</b></a>	260
<i>when &lt;newpage/&gt; is found in an ezText input</i>	
<a href="#"><b>PHPDOCUMENTOR PDF EVENT NEWPAGE</b></a>	260
<i>when &lt;newpage/&gt; is found in an ezText input</i>	
<a href="#"><b>PHPDOCUMENTOR PDF EVENT FONT</b></a>	259
<i>when &lt;font&gt; is found in an ezText input</i>	
<a href="#">phpDocumentor Tutorials</a>	41
<a href="#">phpDocumentor Quickstart</a>	3
<a href="#">phpDocumentor tags</a>	49
<a href="#">phpDocumentor Inline tags</a>	96
<a href="#"><b>PHPDOCUMENTOR PDF EVENT CONTENT</b></a>	259
<i>used for parsing stuff between &lt;text&gt;</i>	
<a href="#">ParserPDF.inc</a>	259
<i>This class handles the XML-based CezPDF markup language created to allow templates for the PDFdefaultConverter</i>	
<a href="#"><b>PHPDOCUMENTOR PDF STATE TEXT</b></a>	260
<i>when &lt;text&gt; is found in an ezText input</i>	
<a href="#">PDFdefaultConverter.inc</a>	261
<i>Outputs documentation in PDF format</i>	
<a href="#">PDFdefaultConverter::\$ric_set</a>	263
<a href="#">PDFdefaultConverter::\$pdf</a>	263
<a href="#">PDFdefaultConverter::\$smarty_dir</a>	263
<a href="#">PDFdefaultConverter::\$sort_absolutely_everything</a>	263
<i>default PDF Converter wants elements sorted by type as well as alphabetically</i>	

<a href="#">PDFdefaultConverter::convertClass()</a>	264
<a href="#">PDFdefaultConverter::\$_sourcecode</a>	263
<i>Source files for appendix C are stored here</i>	
<a href="#">PDFdefaultConverter::\$pagepackage_pagenums</a>	263
<a href="#">PDFdefaultConverter::\$outputformat</a>	262
<a href="#">PDFdefaultConverter::\$classpackage_pagenums</a>	262
<a href="#">PDFdefaultConverter</a>	262
<i>PDF output converter.</i>	
<a href="#">PDFdefaultConverter::\$curclasspackage</a>	262
<a href="#">PDFdefaultConverter::\$curpagepackage</a>	262
<a href="#">PDFdefaultConverter::\$name</a>	262
<a href="#">PDFdefaultConverter::\$leftindex</a>	262
<a href="#">PDFdefaultConverter::mystrnatcasecmp()</a>	271
<a href="#">PDFdefaultConverter::Output()</a>	271
<i>calls <a href="#">Cezpdf::ezOutput()</a> and writes documentation.pdf to targetDir</i>	
<a href="#">phpdocpdf::validHTMLColor()</a>	278
<a href="#">phpdocpdf::textcolor()</a>	278
<a href="#">phpdocpdf:: ezText()</a>	278
<a href="#">PHPDOC_WEBROOT_DIR</a>	352
<a href="#">PDERROR_MULTIPLE_PARENT</a>	515
<i>warning triggered when inheritance could be from more than one class</i>	
<a href="#">passbyref()</a>	501
<i>passes a variable by reference</i>	
<a href="#">phpdocpdf::setupTOC()</a>	278
<a href="#">phpdocpdf::setHTMLColor()</a>	278
<a href="#">phpdocpdf::index()</a>	277
<a href="#">phpdocpdf::indent()</a>	277
<a href="#">phpdocpdf::IndexLetter()</a>	277
<a href="#">phpdocpdf::orderedBullet()</a>	278
<a href="#">phpdocpdf::setColorArray()</a>	278
<a href="#">phpdocpdf::rf()</a>	278
<a href="#">parent_b587733</a>	549
<i>inherited functions with @access private should not be shown in inherited list of child</i>	
<a href="#">priv1</a>	549
<a href="#">PHPDoc_XML_Beautifier_Renderer_Plain::serialize()</a>	617
<i>Serialize the XML tokens</i>	
<a href="#">PHPDoc_XML_Beautifier_Renderer_Plain</a>	617
<i>Basic XML Renderer for XML Beautifier</i>	
<a href="#">phpdoc.php</a>	630
<i>Original Web Interface to phpDocumentor</i>	
<a href="#">phpDocumentor_clone()</a>	632
<i>Clone an object in PHP 4</i>	
<a href="#">phpDocumentor_ConfigFileList()</a>	636
<i>used in phpdock.php and new_phpdock.php</i>	
<a href="#">PATH_DELIMITER</a>	636
<a href="#">Plain.php</a>	616
<i>XML/Beautifier/Renderer/Plain.php</i>	
<a href="#">phpDocumentorTParserGetInlineTagsTests.php</a>	567
<i>Unit Tests for the phpDocumentorTParser-&gt;getInlineTags() method</i>	
<a href="#">ParserPageGetSourceLocationTests.php</a>	561
<i>Unit Tests for the ParserPage-&gt;getSourceLocation() method</i>	
<a href="#">ParserClassGetSourceLocationTests.php</a>	559
<i>Unit Tests for the ParserClass-&gt;getSourceLocation() method</i>	

<a href="#">phpDocumentorSetupCleanConverterNamePieceTests.php</a>	563
<i>Unit Tests for the phpDocumentor_setup-&gt;cleanConverterNamePiece() method</i>	
<a href="#">phpDocumentorSetupDecideOnOrOffTests.php</a>	565
<i>Unit Tests for the phpDocumentor_setup-&gt;decideOnOrOff() method</i>	
<a href="#">PHPUnit_MAIN_METHOD</a>	565
<i>PHPUnit main() hack</i>	
<a href="#">PHPDOCUMENTOR_BASE</a>	565
<i>Base directory of code</i>	
<a href="#">phpdocpdf::getYPlusOffset()</a>	277
<a href="#">phpdocpdf::getColor()</a>	277
<i>This really should be in the parent class</i>	
<a href="#">PDFParser::getParserEventName()</a>	274
<i>Return the name of the parser event</i>	
<a href="#">PDFParser</a>	273
<a href="#">PDFParser::parse()</a>	274
<i>Parse text for PDFParser XML tags, and add the text to the PDF file</i>	
<a href="#">PDFParser::setupStates()</a>	274
<i>setup the parser tokens, and the pushEvent/popEvent arrays</i>	
<a href="#">phpdocpdf::\$converter</a>	275
<a href="#">phpdocpdf</a>	274
<a href="#">PDFdefaultConverter::writeSource()</a>	273
<a href="#">PDFdefaultConverter::writeExample()</a>	273
<a href="#">PDFdefaultConverter::returnLink()</a>	272
<a href="#">PDFdefaultConverter::postProcess()</a>	271
<a href="#">PDFdefaultConverter::returnSee()</a>	272
<i>Returns a bookmark using Cezpdf 009</i>	
<a href="#">PDFdefaultConverter::setTemplateDir()</a>	272
<a href="#">PDFdefaultConverter::unmangle()</a>	272
<a href="#">PDFdefaultConverter::TranslateTag()</a>	272
<i>Used to translate an XML DocBook tag from a tutorial by reading the options.ini file for the template.</i>	
<a href="#">phpdocpdf::\$font_dir</a>	275
<a href="#">phpdocpdf::\$indents</a>	275
<a href="#">phpdocpdf::dots()</a>	276
<a href="#">phpdocpdf::bullet()</a>	276
<a href="#">phpdocpdf::ezNewPage()</a>	276
<a href="#">phpdocpdf::ezOutput()</a>	276
<a href="#">phpdocpdf::ezText()</a>	277
<a href="#">phpdocpdf::ezProcessText()</a>	276
<a href="#">phpdocpdf::addMessage()</a>	276
<a href="#">phpdocpdf::\$_save</a>	275
<a href="#">phpdocpdf::\$listType</a>	275
<a href="#">phpdocpdf::\$indexContents</a>	275
<a href="#">phpdocpdf::\$reportContents</a>	275
<a href="#">phpdocpdf::\$set_pageNumbering</a>	275
<a href="#">phpdocpdf::\$colorStack</a>	275
<a href="#">phpDocumentor_setup::setTemplateBase()</a>	698
<a href="#">phpDocumentor_setup::setTitle()</a>	698
<a href="#">PDERRO_MULTIPLE_PARENT</a>	762
<i>warning triggered when inheritance could be from more than one class</i>	
<a href="#">PDERRO_MULTIPLE_PACKAGE_TAGS</a>	762
<i>warning triggered when there are multiple @package tags in a docblock</i>	
<a href="#">PDERRO_MULTIPLE_RETURN_TAGS</a>	762

<i>warning triggered when there are multiple @return tags in a docblock</i>	
<a href="#"><u>PDERROR_MULTIPLE_SUBPACKAGE_TAGS</u></a>	. . . . . 762
<i>warning triggered when there are multiple @subpackage tags in a docblock</i>	
<a href="#"><u>PDERROR_NAME_ALIAS_SAME_AS_TARGET</u></a>	. . . . . 762
<i>warning triggered when the alias value in an page-level docblock's @name tag</i>	
<a href="#"><u>PDERROR_MULTIPLE_VAR_TAGS</u></a>	. . . . . 762
<i>warning triggered when there are multiple @var tags in a docblock</i>	
<a href="#"><u>PDERROR_MULTIPLE_NAME_TAGS</u></a>	. . . . . 761
<i>warning triggered when there are multiple @name tags in a docblock</i>	
<a href="#"><u>PDERROR_MULTIPLE_GLOBAL_TAGS</u></a>	. . . . . 761
<i>warning triggered when there are multiple @name tags in a docblock</i>	
<a href="#"><u>PDERROR_MALFORMED_GLOBAL_TAG</u></a>	. . . . . 761
<i>warning triggered when there are multiple @name tags in a docblock</i>	
<a href="#"><u>PDERROR_LOOP_RECURSION_LIMIT_REACHED</u></a>	. . . . . 761
<i>warning triggered when the a loop recursion tripwire has been tripped</i>	
<a href="#"><u>PDERROR_MALFORMED_TAG</u></a>	. . . . . 761
<i>warning triggered by an empty tag</i>	
<a href="#"><u>PDERROR_MISSING_PROPERTY_TAG_NAME</u></a>	. . . . . 761
<i>warning triggered when any of {@property}, {@property-read},</i>	
<a href="#"><u>PDERROR_MULTIPLE_CATEGORY_TAGS</u></a>	. . . . . 761
<i>warning triggered by more than 1 @category tag</i>	
<a href="#"><u>PDERROR_MULTIPLE_ACCESS_TAGS</u></a>	. . . . . 761
<i>warning triggered when there are multiple @access tags in a docblock</i>	
<a href="#"><u>PDERROR_NEED_WHITESPACE</u></a>	. . . . . 762
<i>warning triggered when an entire page is ignored because of @access private</i>	
<a href="#"><u>PDERROR_NESTED_INTERNAL</u></a>	. . . . . 762
<i>warning triggered when {@internal} is nested inside another {@internal}</i>	
<a href="#"><u>PDERROR_PACKAGEOUTPUT_DELETES_PARENT_FILE</u></a>	. . . . . 763
<i>warning triggered when there are multiple @name tags in a docblock</i>	
<a href="#"><u>PDERROR_PACKAGECAT_SET</u></a>	. . . . . 763
<i>warning triggered when a package is already associated with a category, and</i>	
<a href="#"><u>PDERROR_PARENT_NOT_FOUND</u></a>	. . . . . 763
<i>warning triggered when parent class doesn't exist</i>	
<a href="#"><u>PDERROR_PARSEPRIVATE</u></a>	. . . . . 764
<i>warning triggered when an entire page is ignored because of @access private</i>	
<a href="#"><u>PDERROR_PDF_METHOD_DOESNT_EXIST</u></a>	. . . . . 764
<i>warning triggered when a &lt;pdffunction:funcname /&gt; tag is used in the PDF</i>	
<a href="#"><u>PDERROR_PDFFUNCTION_NO_FUNC</u></a>	. . . . . 764
<i>warning triggered when a &lt;pdffunction:funcname /&gt; tag is used in the PDF</i>	
<a href="#"><u>PDERROR_OVERRIDDEN_SUBPACKAGE_TAGS</u></a>	. . . . . 763
<i>warning triggered when there a @subpackage tag is used in a function,</i>	
<a href="#"><u>PDERROR_OVERRIDDEN_PACKAGE_TAGS</u></a>	. . . . . 763
<i>warning triggered when there a @package tag is used in a function,</i>	
<a href="#"><u>PDERROR_NO_CONVERTERS</u></a>	. . . . . 762
<i>warning triggered when a converter is passed to</i>	
<a href="#"><u>PDERROR_NOTODO_INCLUDE</u></a>	. . . . . 762
<i>warning triggered when @todo is used on an include element</i>	
<a href="#"><u>PDERROR_NO_CONVERTER_HANDLER</u></a>	. . . . . 763
<i>warning triggered when an example's path from @example /path/to/example.php</i>	
<a href="#"><u>PDERROR_NO_DOCBOOK_ID</u></a>	. . . . . 763
<i>warning triggered in tutorial parsing if there is a missing {@id} inline tag</i>	
<a href="#"><u>PDERROR_NO_PAGE_LEVELDOCBLOCK</u></a>	. . . . . 763
<i>warning triggered when a file does not contain a page-level docblock</i>	

<a href="#">PDERRO</a>	<a href="#">NO PACKAGE TAG</a>	763
	warning triggered when no @package tag is used in a page-level	
<a href="#">PDERRO</a>	<a href="#">INVALID VALUES</a>	761
	warning triggered if a command line option does not have a valid value passed in	
<a href="#">PDERRO</a>	<a href="#">INTERNAL NOT CLOSED</a>	760
	warning triggered when an {@internal} tag is not closed	
<a href="#">PDERRO</a>	<a href="#">CLASS EXISTS</a>	758
	warning triggered when inheritance could be from more than one class	
<a href="#">PDERRO</a>	<a href="#">CLASS CONFLICT</a>	758
	warning triggered when classes in the same package have the same name	
<a href="#">PDERRO</a>	<a href="#">CLASS NOT IN PACKAGE</a>	758
	warning triggered when a getClassByPackage is called and can't find the class	
<a href="#">PDERRO</a>	<a href="#">CLASS PARENT NOT FOUND</a>	758
	warning triggered when an entire page is ignored because of @access private	
<a href="#">PDERRO</a>	<a href="#">CONVERTER OVR GFCT</a>	758
	warning triggered when a child converter doesn't override	
<a href="#">PDERRO</a>	<a href="#">CONVERTER NOT FOUND</a>	758
	warning triggered when a converter is passed to	
<a href="#">PDERRO</a>	<a href="#">CHILD TUTORIAL NOT FOUND</a>	758
	warning triggered when a tutorial's child in the .ini file doesn't exist in the	
<a href="#">PDERRO</a>	<a href="#">CANT NEST IN B</a>	758
	warning triggered when another tag is nested in <b>;	
<a href="#">parserVarTag::\$returnType</a>		749
	the type a var has	
<a href="#">parserVarTag::\$keyword</a>		749
	always 'var'	
<a href="#">PDERRO</a>	<a href="#">ACCESS WRONG PARAM</a>	757
	warning triggered when the arguments to @access are neither public nor private	
<a href="#">PDERRO</a>	<a href="#">BEAUTIFYING FAILED</a>	757
	warning triggered by an unterminated entity in a tutorial	
<a href="#">PDERRO</a>	<a href="#">CANT HAVE INLINE IN TAGNAME</a>	758
	warning triggered when an inline tag is found inside an xml tag name	
<a href="#">PDERRO</a>	<a href="#">CANNOT EXTEND SELF</a>	757
	warning triggered if someone brilliant tries "class X extends X {"	
<a href="#">PDERRO</a>	<a href="#">DANGEROUS PHP BUG EXISTS</a>	759
	warning triggered when the PHP version being used has dangerous bug/behavior	
<a href="#">PDERRO</a>	<a href="#">DB TEMPLATE UNTERMINATED</a>	759
	warning triggered when a docblock template is never turned off	
<a href="#">PDERRO</a>	<a href="#">IGNORE TAG IGNORED</a>	760
	warning triggered when an @ignore tag is used in a DocBlock preceding	
<a href="#">PDERRO</a>	<a href="#">ID MUST BE INLINE</a>	760
	warning triggered when an id attribute in a tutorial docbook tag is not	
<a href="#">PDERRO</a>	<a href="#">ILLEGAL PACKAGENAME</a>	760
	warning triggered when the package or subpackage name is illegal	
<a href="#">PDERRO</a>	<a href="#">INHERITANCE CONFLICT</a>	760
	warning triggered when inheritance could be from more than one class	
<a href="#">PDERRO</a>	<a href="#">INLINETAG IN SEE</a>	760
	warning triggered when an example's path from @example /path/to/example.php	
<a href="#">PDERRO</a>	<a href="#">INHERITDOC DONT WORK HERE</a>	760
	warning triggered by {@inheritdoc} in a non-inheritable situation	
<a href="#">PDERRO</a>	<a href="#">GLOBAL NOT FOUND</a>	760
	warning triggered when there are multiple @name tags in a docblock	
<a href="#">PDERRO</a>	<a href="#">FUNCTION HAS NONAME</a>	759

<i>warning triggered by a function with no name</i>	
<a href="#">PDERROR_DOCBLOCK_GOES_CLASS</a>	. . . . . 759
<i>warning triggered when the first docblock in a file with a @package tag precedes a class. In this case, the class gets the docblock.</i>	
<a href="#">PDERROR_DOCBLOCK_CONFLICT</a>	. . . . . 759
<i>warning triggered by a page-level docblock preceding a source element</i>	
<a href="#">PDERROR_DUMBUSES</a>	. . . . . 759
<i>warning triggered by improper "@uses        blah"</i>	
<a href="#">PDERRORELEMENT_IGNORED</a>	. . . . . 759
<i>warning triggered when a duplicate element is encountered that will be</i>	
<a href="#">PDERROREXAMPLE_NOT_FOUND</a>	. . . . . 759
<i>warning triggered when an example's path from @example /path/to/example.php</i>	
<a href="#">PDERROREMPTY_EXAMPLE_TITLE</a>	. . . . . 759
<i>warning triggered by @example path/to/example with no title</i>	
<a href="#">PDERROPDF_TEMPVAR_DOESNT_EXIST</a>	. . . . . 764
<i>warning triggered when a &lt;pdffunction:funcname arg=\$tempvar/&gt; tag</i>	
<a href="#">PDERROPRIVATE_ASSUMED</a>	. . . . . 764
<i>warning triggered when no @access private tag is used in a</i>	
<a href="#">parserLinkInlineTag::\$linktext</a>	. . . . . 780
<i>text to display in the link, can be different from the link for standard</i>	
<a href="#">parserLinkInlineTag</a>	. . . . . 780
<i>represents inline links</i>	
<a href="#">parserLinkInlineTag::Convert()</a>	. . . . . 781
<i>calls the output conversion</i>	
<a href="#">parserLinkInlineTag::ConvertPart()</a>	. . . . . 781
<i>convert part of the tag</i>	
<a href="#">parserSourceInlineTag::\$end</a>	. . . . . 782
<i>Last line to display</i>	
<a href="#">parserSourceInlineTag</a>	. . . . . 781
<i>represents inline source tag, used for function/method source</i>	
<a href="#">parserInlineTag::Strlen()</a>	. . . . . 779
<i>get length of the tag</i>	
<a href="#">parserInlineTag::getString()</a>	. . . . . 779
<i>always gets an empty string</i>	
<a href="#">parserInheritdocInlineTag::\$inlinetype</a>	. . . . . 778
<i>always 'inheritdoc'</i>	
<a href="#">parserInheritdocInlineTag</a>	. . . . . 777
<i>Represents the inheritdoc inline tag, used by classes/methods/vars to inherit documentation from the parent class if possible</i>	
<a href="#">parserInheritdocInlineTag::Convert()</a>	. . . . . 778
<i>only sets a warning and returns empty</i>	
<a href="#">parserInlineTag</a>	. . . . . 778
<i>Use this element to represent an {@inline tag} like {@link}</i>	
<a href="#">parserInlineTag::\$type</a>	. . . . . 779
<i>Element type</i>	
<a href="#">parserInlineTag::\$inlinetype</a>	. . . . . 779
<i>the name of the inline tag (like link)</i>	
<a href="#">parserSourceInlineTag::\$inlinetype</a>	. . . . . 782
<i>always 'source'</i>	
<a href="#">parserSourceInlineTag::\$source</a>	. . . . . 782
<i>tokenized source organized by line numbers for php 4.3.0+, the old</i>	
<a href="#">parserToclineTag::Convert()</a>	. . . . . 785
<i>converter method</i>	

<a href="#">parserTocInlineTag::\$inlinetype</a>	785
always 'toc'	
<a href="#">parserTocInlineTag::setPath()</a>	786
set the path	
<a href="#">parserTocInlineTag::setTOC()</a>	786
set the TOC	
<a href="#">parserTutorialInlineTag::Convert()</a>	787
convert part of the tag	
<a href="#">parserTutorialInlineTag</a>	786
Represents inline links to external tutorial documentation	
<a href="#">parserTocInlineTag</a>	785
Represents {@toc} for table of contents generation in tutorials	
<a href="#">parserSourceInlineTag::Strlen()</a>	784
only used to determine blank lines. {@source} will not be blank, probably	
<a href="#">parserSourceInlineTag::arrayConvert()</a>	783
converter helper used in PHP 4.3.0+	
<a href="#">parserSourceInlineTag::\$start</a>	782
First line of source code to display	
<a href="#">parserSourceInlineTag::Convert()</a>	783
convert the tag	
<a href="#">parserSourceInlineTag::getString()</a>	784
gets the source string	
<a href="#">parserSourceInlineTag::stringConvert()</a>	784
converter helper used in PHP 4.2.3-	
<a href="#">parserSourceInlineTag::setSource()</a>	784
sets the source tag's value	
<a href="#">parserIdInlineTag::Convert()</a>	777
converter	
<a href="#">parserIdInlineTag::\$tutorial</a>	776
full name of the tutorial	
<a href="#">PDERROR_TUTORIAL_SUBSECTION_NOT_FOUND</a>	765
warning triggered when a subsection's title is asked for, but the subsection	
<a href="#">PDERROR_TUTORIAL_NOT_FOUND</a>	765
warning triggered when a tutorial is referenced	
<a href="#">PDERROR_UL_IN_UL</a>	765
warning triggered if <<ul>> is nested inside <<ul>> and not	
<<li>>	
<a href="#">PDERROR_UNCLOSED_TAG</a>	765
warning triggered when a DocBlock html tag is unclosed	
<a href="#">PDERROR_UNKNOWN_COMMANDLINE</a>	765
warning triggered when an entire page is ignored because of @access private	
<a href="#">PDERROR_UNDOCUMENTED_ELEMENT</a>	765
warning triggered when a class or method hasn't got docblock	
<a href="#">PDERROR_TUTORIAL_IS_OWN_GRANDPA</a>	765
warning triggered when a tutorial's child lists the parent tutorial	
<a href="#">PDERROR_TUTORIAL_IS_OWN_CHILD</a>	765
warning triggered when a tutorial lists itself as a child tutorial	
<a href="#">PDERROR_SOURCE_TAG_FUNCTION_NOT_FOUND</a>	764
warning triggered when a { @source } inline tag is used in a docblock not	
<a href="#">PDERROR_SOURCECODE_IGNORED</a>	764
warning triggered by sourcecode="on", if PHP < 4.3.0	
<a href="#">PDERROR_SOURCE_TAG_IGNORED</a>	764
warning triggered when an {@source} tag is found in a short description	

<a href="#"><u>PDERROR_TAG_NOT_HANDLED</u></a>	. . . . .	764
warning triggered by @filesource, if PHP < 4.3.0		
<a href="#"><u>PDERROR_TEXT_OUTSIDE_LIST</u></a>	. . . . .	765
warning triggered when text in a docblock list is not contained in		
<a href="#"><u>PDERROR_TEMPLATEDIR_DOESNT_EXIST</u></a>	. . . . .	764
warning triggered by non-existent template directory		
<a href="#"><u>PDERROR_UNKNOWN_TAG</u></a>	. . . . .	766
warning triggered when classes in the same package have the same name		
<a href="#"><u>PDERROR_UNMATCHED_LIST_TAG</u></a>	. . . . .	766
warning triggered when a docblock has an unmatched &lt;ol&gt; or &lt;ul&gt;		
<a href="#"><u>parserIdInlineTag::\$category</u></a>	. . . . .	776
category of the {@id}		
<a href="#"><u>parserIdInlineTag</u></a>	. . . . .	775
Represents the inline {@id} tag for tutorials		
<a href="#"><u>parserIdInlineTag::\$id</u></a>	. . . . .	776
section/subsection name		
<a href="#"><u>parserIdInlineTag::\$inlinetype</u></a>	. . . . .	776
always 'id'		
<a href="#"><u>parserIdInlineTag::\$subpackage</u></a>	. . . . .	776
subpackage of the {@id}		
<a href="#"><u>parserIdInlineTag::\$package</u></a>	. . . . .	776
package of the {@id}		
<a href="#"><u>parserExampleInlineTag::setSource()</u></a>	. . . . .	775
sets the source		
<a href="#"><u>parserExampleInlineTag::getProgramListing()</u></a>	. . . . .	775
Return the source for the example file, enclosed in a <programlisting> tag to use in a tutorial		
<a href="#"><u>PDERROR_UNTERMINATED_ATTRIB</u></a>	. . . . .	766
warning triggered when a subsection's title is asked for, but the subsection		
<a href="#"><u>PDERROR_UNMATCHED_TUTORIAL_TAG</u></a>	. . . . .	766
warning triggered when a docbook tag is not properly matched		
<a href="#"><u>PDERROR_UNTERMINATED_ENTITY</u></a>	. . . . .	766
warning triggered by an unterminated entity in a tutorial		
<a href="#"><u>PDERROR_UNTERMINATED_INLINE_TAG</u></a>	. . . . .	766
warning triggered when an {@inline tag} is not terminated		
<a href="#"><u>parserExampleInlineTag::arrayConvert()</u></a>	. . . . .	774
converter helper for PHP 4.3.0+		
<a href="#"><u>parserExampleInlineTag</u></a>	. . . . .	773
Represents the example inline tag, used to display an example file inside a docblock or tutorial		
<a href="#"><u>parserVarTag</u></a>	. . . . .	748
represents the "@var" tag		
<a href="#"><u>parserUsesTag::getDescription()</u></a>	. . . . .	748
Get the description of how the element used is being used.		
<a href="#"><u>Publisher::subscribe()</u></a>	. . . . .	709
Adds a subscriber to the \$subscriber array().		
<a href="#"><u>Publisher::publishEvent()</u></a>	. . . . .	709
Publish an event		
<a href="#"><u>parserB</u></a>	. . . . .	716
Used for <>b</b></> in a description		
<a href="#"><u>parserB::Convert()</u></a>	. . . . .	716
performs the conversion of bold tags		

<a href="#">parserBr::Convert()</a>	. . . . .	717
	performs the conversion of linebreak tags	
<a href="#">parserBr</a>	. . . . .	717
	Used for <>br></> in a description	
<a href="#">Publisher::\$tokens</a>	. . . . .	709
<a href="#">Publisher::\$subscriber</a>	. . . . .	709
	Array of references objects that have Subscribed to this publisher	
<a href="#">ProceduralPages::setupPagePackages()</a>	. . . . .	707
	if there is one class package in a file, the parent path inherits the package if its package is default.	
<a href="#">ProceduralPages::setParseBase()</a>	. . . . .	707
	sets the parser base	
<a href="#">ProceduralPages::setupPages()</a>	. . . . .	708
	Adjusts packages of all pages and removes name conflicts within a package	
<a href="#">Publisher</a>	. . . . .	708
	a class for handling the publishing of data	
<a href="#">Publisher::\$pushEvent</a>	. . . . .	709
<a href="#">Publisher::\$popEvent</a>	. . . . .	709
<a href="#">parserCode</a>	. . . . .	718
	Used for <>code></> in a description	
<a href="#">parserCode::Convert()</a>	. . . . .	718
	performs the conversion of code tags	
<a href="#">parserList::addItem()</a>	. . . . .	722
	add an item to a list	
<a href="#">parserList::\$numbered</a>	. . . . .	722
<a href="#">parserList::addList()</a>	. . . . .	722
	add a list	
<a href="#">parserList::Convert()</a>	. . . . .	722
	performs the conversion of list tags	
<a href="#">parserPre::Convert()</a>	. . . . .	723
	performs the conversion of code tags	
<a href="#">parserPre</a>	. . . . .	723
	Used for <>pre></> in a description	
<a href="#">parserList::\$items</a>	. . . . .	721
<a href="#">parserList</a>	. . . . .	721
	Used for lists <>ol></> and <>ul></>	
<a href="#">parserDescVar::Convert()</a>	. . . . .	719
	performs the conversion of variable tags	
<a href="#">parserDescVar</a>	. . . . .	718
	Used for <>var></> in a description	
<a href="#">parserI</a>	. . . . .	719
	Used for <>i></> in a description	
<a href="#">parserI::Convert()</a>	. . . . .	720
	performs the conversion of italic tags	
<a href="#">parserKbd::Convert()</a>	. . . . .	721
	performs the conversion of keyboard tags	
<a href="#">parserKbd</a>	. . . . .	720
	Used for <>kbd></> in a description	
<a href="#">ProceduralPages::setName()</a>	. . . . .	707
	Change a page's name from its file to alias \$name	
<a href="#">ProceduralPages::replaceElement()</a>	. . . . .	707
	Used to align an element with the package of its parent page prior to Conversion.	

<a href="#">ProceduralPages::\$globalconflicts</a>	. . . . .	701
<i>Namespace conflicts within all documented packages of functions</i>		
<a href="#">ProceduralPages::\$functionsbynamefile</a>	. . . . .	701
<i>array of file names organized by functions that are in the file.</i>		
<a href="#">ProceduralPages::\$globalsbyfile</a>	. . . . .	702
<i>array of parsed global variables organized by the full path         of the file that contains the global variable definition.</i>		
<a href="#">ProceduralPages::\$globalsbynamefile</a>	. . . . .	702
<i>array of file names organized by global variables that are in the file.</i>		
<a href="#">ProceduralPages::\$includesbyfile</a>	. . . . .	702
<i>array of parsed includes organized by the full path         of the file that contains the include.</i>		
<a href="#">ProceduralPages::\$ignorepages</a>	. . . . .	702
<i>array of all procedural pages ordered by name</i>		
<a href="#">ProceduralPages::\$functionsbyfile</a>	. . . . .	701
<i>array of parsed functions organized by the full path         of the file that contains the function.</i>		
<a href="#">ProceduralPages::\$functionconflicts</a>	. . . . .	700
<i>Namespace conflicts within all documented packages of functions</i>		
<a href="#">ProceduralPages</a>	. . . . .	699
<i>Intermediate procedural page parsing structure.</i>		
<a href="#">phpDocumentor_setup::setUndocumentedElementWarnings()</a>	. . . . .	698
<a href="#">ProceduralPages::\$curfile</a>	. . . . .	699
<i>file being parsed, used in every add function         to match up elements with the file that contains them</i>		
<a href="#">ProceduralPages::\$defineconflicts</a>	. . . . .	699
<i>Namespace conflicts within all documented packages of functions</i>		
<a href="#">ProceduralPages::\$definesbynamefile</a>	. . . . .	700
<i>array of file names organized by defines that are in the file.</i>		
<a href="#">ProceduralPages::\$definesbyfile</a>	. . . . .	700
<i>array of parsed defines organized by the full path         of the file that contains the define.</i>		
<a href="#">ProceduralPages::\$pageclasspackages</a>	. . . . .	703
<i>array of packages assigned to classes in a file, ordered by fullname</i>		
<a href="#">ProceduralPages::\$pageconflicts</a>	. . . . .	703
<i>Namespace conflicts within all documented pages</i>		
<a href="#">ProceduralPages::addPagePackage()</a>	. . . . .	705
<i>Changes the package of the page represented by \$path</i>		
<a href="#">ProceduralPages::addPage()</a>	. . . . .	705
<i>sets up the \$pages array</i>		
<a href="#">ProceduralPages::getPathInfo()</a>	. . . . .	706
<i>gathers path-related info about a given element</i>		
<a href="#">ProceduralPages::getRealPath()</a>	. . . . .	706
<i>Ensures the path to the file is an absolute path</i>		
<a href="#">ProceduralPages::pathMatchesParsedFile()</a>	. . . . .	707
<i>checks to see if the parsed file matches the given path</i>		
<a href="#">ProceduralPages::ignorePage()</a>	. . . . .	706
<i>moves a page from the \$pages array to the \$ignorepages array</i>		
<a href="#">ProceduralPages::addInclude()</a>	. . . . .	705
<i>sets up the \$includesbyfile array using \$curfile</i>		
<a href="#">ProceduralPages::addGlobal()</a>	. . . . .	705
<i>sets up the \$globalsbyfile array using \$curfile</i>		
<a href="#">ProceduralPages::\$pages</a>	. . . . .	704

<i>array of all procedural pages ordered by name</i>	
<a href="#">ProceduralPages::\$pagepackages</a>	703
<i>array of packages ordered by full path</i>	
<a href="#">ProceduralPages::\$pathpages</a>	704
<i>array of all procedural page names ordered by full path to the file</i>	
<a href="#">ProceduralPages::addClassPackageToFile()</a>	704
<i>adds a package from a class to the current file</i>	
<a href="#">ProceduralPages::addFunction()</a>	705
<i>sets up the \$functionsbyfile array using \$curfile</i>	
<a href="#">ProceduralPages::addDefine()</a>	704
<i>sets up the \$definesbyfile array using \$curfile</i>	
<a href="#">parserSamp</a>	724
<i>Used for &lt;&gt; in a description</i>	
<a href="#">parserSamp::Convert()</a>	724
<i>performs the conversion of sample tags</i>	
<a href="#">parserReturnTag::Convert()</a>	740
<i>process this tag through the given output converter (sets up the \$converted_returnType)</i>	
<a href="#">parserReturnTag::\$returnType</a>	739
<i>the type a function returns</i>	
<a href="#">parserSeeTag</a>	740
<i>represents the "@see" tag</i>	
<a href="#">parserSeeTag::\$keyword</a>	741
<i>always 'see'</i>	
<a href="#">parserStaticvarTag</a>	741
<i>represents the "@staticvar" tag</i>	
<a href="#">parserSeeTag::Convert()</a>	741
<i>process this tag through the given output converter</i>	
<a href="#">parserReturnTag::\$keyword</a>	739
<i>always 'return'</i>	
<a href="#">parserReturnTag::\$converted_returnType</a>	739
<i>contains a link to the documentation for a class passed as a type in @return, @var or @param</i>	
<a href="#">parserPropertyTag::\$keyword</a>	737
<i>always 'property'</i>	
<a href="#">parserPropertyTag</a>	737
<i>represents the "@property" tag</i>	
<a href="#">parserPropertyTag::\$returnType</a>	737
<i>the type a property has</i>	
<a href="#">parserPropertyWriteTag</a>	738
<i>represents the "@property-write" tag</i>	
<a href="#">parserReturnTag</a>	738
<i>represents the "@return" tag</i>	
<a href="#">parserPropertyWriteTag::\$keyword</a>	738
<i>always 'property-write'</i>	
<a href="#">parserStaticvarTag::\$keyword</a>	742
<i>always 'staticvar'</i>	
<a href="#">parserTag</a>	742
<i>used to represent standard tags like @access, etc.</i>	
<a href="#">parserUsedByTag::\$keyword</a>	746
<i>Always "usedby"</i>	
<a href="#">parserUsedByTag</a>	746
<i>This is a virtual tag, it is created by @uses to cross-reference the used element</i>	

<a href="#">parserUsedByTag::Convert()</a>	746
<i>process this tag through the given output converter</i>	
<a href="#">parserUsesTag</a>	747
<i>represents the "@uses" tag</i>	
<a href="#">parserUsesTag::Convert()</a>	748
<i>Return a link to documentation for other element, and description of how it is used</i>	
<a href="#">parserUsesTag::\$keyword</a>	747
<i>Always "uses"</i>	
<a href="#">parserTutorialTag::Convert()</a>	745
<i>process this tag through the given output converter</i>	
<a href="#">parserTutorialTag::\$keyword</a>	745
<i>Always "tutorial"</i>	
<a href="#">parserTag::\$type</a>	743
<i>Type is used by many functions to skip the hassle of</i>	
<a href="#">parserTag::\$keyword</a>	743
<i>tag name (see, access, etc.)</i>	
<a href="#">parserTag::Convert()</a>	743
<i>Perform the output conversion on this <a href="#">parserTag</a> using the <a href="#">output converter</a> that is passed in</i>	
<a href="#">parserTag::getString()</a>	744
<i>Returns the text minus any inline tags</i>	
<a href="#">parserTutorialTag</a>	744
<i>represents "@tutorial"</i>	
<a href="#">parserTag::HandleEvent()</a>	744
<i>Called by the <a href="#">parserDescParser</a> when processing a description.</i>	
<a href="#">parserPropertyReadTag::\$keyword</a>	736
<i>always 'property-read'</i>	
<a href="#">parserPropertyReadTag</a>	736
<i>represents the "@property-read" tag</i>	
<a href="#">parserFileSourceTag</a>	729
<i>represents "@filesource"</i>	
<a href="#">parserExampleTag::getSourceLink()</a>	729
<i>Retrieve a converter-specific link to the example</i>	
<a href="#">parserFileSourceTag::\$keyword</a>	729
<i>Always "filesource"</i>	
<a href="#">parserFileSourceTag::\$path</a>	730
<a href="#">parserFileSourceTag::Convert()</a>	730
<i>Return a link to the highlighted source and generate the source</i>	
<a href="#">parserFileSourceTag::\$source</a>	730
<a href="#">parserExampleTag::ConvertSource()</a>	728
<i>convert the source code</i>	
<a href="#">parserExampleTag::\$keyword</a>	728
<i>always "example"</i>	
<a href="#">parserAccessTag::\$isValid</a>	726
<i>set to true if the returned tag has a value type of private, protected</i>	
<a href="#">parserAccessTag</a>	726
<i>This class represents the @access tag</i>	
<a href="#">parserAccessTag::\$keyword</a>	726
<i>tag name</i>	
<a href="#">parserAccessTag::Convert()</a>	727
<i>process this tag through the given output converter</i>	
<a href="#">parserExampleTag</a>	727

<i>represents "@example"</i>	
<a href="#">parserAccessTag::getString()</a>	727
<i>No inline tags are possible, returns 'public', 'protected' or 'private'</i>	
<a href="#">parserFileSourceTag::ConvertSource()</a>	730
<i>convert the source code</i>	
<a href="#">parserFileSourceTag::getSourceLink()</a>	731
<i>gets path to the sourcecode file</i>	
<a href="#">parserNameTag</a>	734
<i>This class represents the @name tag</i>	
<a href="#">parserMethodTag::\$returnType</a>	734
<i>the return type a method has</i>	
<a href="#">parserNameTag::\$keyword</a>	735
<i>tag name</i>	
<a href="#">parserNameTag::Convert()</a>	735
<i>process this tag through the given output converter</i>	
<a href="#">parserParamTag::\$keyword</a>	736
<i>always 'param'</i>	
<a href="#">parserParamTag</a>	735
<i>represents the "@param" tag</i>	
<a href="#">parserMethodTag::\$keyword</a>	734
<i>always 'method'</i>	
<a href="#">parserMethodTag</a>	733
<i>represents the "@method" tag</i>	
<a href="#">parserLicenseTag</a>	732
<i>represents the "@see" tag</i>	
<a href="#">parserFileSourceTag::writeSource()</a>	731
<i>have the output converter write the source code</i>	
<a href="#">parserLicenseTag::\$keyword</a>	732
<i>always 'license'</i>	
<a href="#">parserLinkTag</a>	732
<i>represents the "@link" tag</i>	
<a href="#">parserLinkTag::\$keyword</a>	733
<i>always 'link'</i>	
<a href="#">phpDocumentor Guide to Creating Fantastic Documentation</a>	1

## R

<a href="#">RecordWarning::\$linenum</a>	771
<i>line number of the file this error occurred in</i>	
<a href="#">RecordWarning::\$file</a>	771
<i>file this error occurred in</i>	
<a href="#">RecordWarning::\$num</a>	771
<i>error number</i>	
<a href="#">RecordWarning::\$type</a>	771
<i>name of global variable that descriptors for this warning/error is kept</i>	
<a href="#">README</a>	949
<a href="#">RecordWarning::output()</a>	772
<i>prints the warning</i>	
<a href="#">RecordWarning::\$data</a>	770
<i>error string</i>	
<a href="#">RecordWarning</a>	770
<i>encapsulates warning information</i>	

<a href="#">recurseDir()</a>	626
<a href="#">RecordWarning</a>	550
<a href="#">RecordError</a>	769
<i>encapsulates error information</i>	
<a href="#">RecordError::\$type</a>	770
<i>name of global variable that descriptors for this warning/error is kept</i>	
<a href="#">RecordError::output()</a>	770
<i>prints the error</i>	
<a href="#">returnsme()</a>	501
<i>returns a reference</i>	

## S

<a href="#">smarty_modifier_htmlentities()</a>	467
<i>Smarty plugin</i>	
<a href="#">smarty_modifier_escape()</a>	466
<i>Smarty escape modifier plugin</i>	
<a href="#">smarty_modifier_default()</a>	465
<i>Smarty default modifier plugin</i>	
<a href="#">smarty_modifier_indent()</a>	468
<i>Smarty indent modifier plugin</i>	
<a href="#">smarty_modifier_lower()</a>	469
<i>Smarty lower modifier plugin</i>	
<a href="#">smarty_modifier_regex_replace()</a>	472
<i>Smarty regex_replace modifier plugin</i>	
<a href="#">smarty_modifier_rawurlencode()</a>	471
<i>Smarty plugin</i>	
<a href="#">smarty_modifier_nl2br()</a>	470
<i>Smarty plugin</i>	
<a href="#">smarty_modifier_debug_print_var()</a>	464
<i>Smarty debug_print_var modifier plugin</i>	
<a href="#">smarty_modifier_date_format()</a>	463
<i>Smarty date_format modifier plugin</i>	
<a href="#">smarty_modifier_cat()</a>	458
<i>Smarty cat modifier plugin</i>	
<a href="#">smarty_modifier_capitalize()</a>	457
<i>Smarty capitalize modifier plugin</i>	
<a href="#">smarty_function_var_dump()</a>	456
<i>Smarty plugin</i>	
<a href="#">smarty_modifier_count_characters()</a>	459
<i>Smarty count_characters modifier plugin</i>	
<a href="#">smarty_modifier_count_paragraphs()</a>	460
<i>Smarty count_paragraphs modifier plugin</i>	
<a href="#">smarty_modifier_count_words()</a>	462
<i>Smarty count_words modifier plugin</i>	
<a href="#">smarty_modifier_count_sentences()</a>	461
<i>Smarty count_sentences modifier plugin</i>	
<a href="#">smarty_modifier_replace()</a>	473
<i>Smarty replace modifier plugin</i>	
<a href="#">smarty_modifier_spacify()</a>	474
<i>Smarty spacify modifier plugin</i>	
<a href="#">smarty_make_timestamp()</a>	484

<i>Function: smarty_make_timestamp&lt;br&gt;</i>	
<i>Purpose: used by other smarty functions to make a timestamp from a string.</i>	
<a href="#">shared.make_timestamp.php</a>	484
<i>Smarty shared plugin</i>	
<a href="#">smarty_function_escape_special_chars()</a>	483
<i>escape_special_chars common function</i>	
<a href="#">SendEMail()</a>	491
<i>This is a Test</i>	
<a href="#">SMART_PATH_DELIMITER</a>	523
<i>does not parse correctly because of function in definition</i>	
<a href="#">summary_form::blah()</a>	550
<a href="#">summary_form::\$dp</a>	550
<a href="#">summary_form</a>	550
<a href="#">shared.escape_special_chars.php</a>	483
<i>Smarty shared plugin</i>	
<a href="#">smarty_outputfilter_trimwhitespace_replace()</a>	481
<a href="#">smarty_modifier_strip_tags()</a>	477
<i>Smarty strip_tags modifier plugin</i>	
<a href="#">smarty_modifier_strip()</a>	476
<i>Smarty strip modifier plugin</i>	
<a href="#">smarty_modifier_string_format()</a>	475
<i>Smarty string_format modifier plugin</i>	
<a href="#">smarty_modifier_truncate()</a>	478
<i>Smarty truncate modifier plugin</i>	
<a href="#">smarty_modifier_upper()</a>	479
<i>Smarty upper modifier plugin</i>	
<a href="#">smarty_outputfilter_trimwhitespace()</a>	481
<i>Smarty trim whitespace outputfilter plugin</i>	
<a href="#">smarty_modifier_wordwrap()</a>	480
<i>Smarty wordwrap modifier plugin</i>	
<a href="#">smarty_function_popup_init()</a>	455
<i>Smarty {popup_init} function plugin</i>	
<a href="#">smarty_function_popup()</a>	454
<i>Smarty {popup} function plugin</i>	
<a href="#">smarty_block_strip()</a>	427
<i>Smarty {strip}/{/strip} block plugin</i>	
<a href="#">smarty_core_write_file()</a>	426
<i>write out a file to disk</i>	
<a href="#">smarty_core_write_compiled_resource()</a>	425
<i>write the compiled resource</i>	
<a href="#">smarty_block_textformat()</a>	428
<i>Smarty {textformat}/{/textformat} block plugin</i>	
<a href="#">smarty_function_assign()</a>	429
<i>Smarty {assign} function plugin</i>	
<a href="#">smarty_function_counter()</a>	432
<i>Smarty {counter} function plugin</i>	
<a href="#">smarty_function_config_load()</a>	431
<i>Smarty {config_load} function plugin</i>	
<a href="#">smarty_function_assign_debug_info()</a>	430
<i>Smarty {assign_debug_info} function plugin</i>	
<a href="#">smarty_core_write_compiled_include()</a>	424
<i>Extract non-cacheable parts out of compiled template and write it</i>	

<a href="#">smarty_core_write_cache_file()</a>	423
Prepend the cache information to the cache file and write it	
<a href="#">smarty_core_read_cache_file()</a>	418
read a cache file, determine if it needs to be regenerated or not	
<a href="#">smarty_core_process_compiled_include()</a>	417
Replace nocache-tags by results of the corresponding non-cacheable functions and return it	
<a href="#">smarty_core_process_cached_inserts()</a>	416
Replace cached inserts with the actual results	
<a href="#">smarty_core_rmdir()</a>	419
delete a dir recursively (level=0 -> keep root) WARNING: no tests, it will try to remove what you tell it!	
<a href="#">smarty_core_rm_auto()</a>	420
delete an automatically created file by name and id	
<a href="#">smarty_core_smarty_include_php()</a>	422
called for included php files within templates	
<a href="#">smarty_core_run_insert_handler()</a>	421
Handle insert tags	
<a href="#">smarty_function_cycle()</a>	433
Smarty {cycle} function plugin	
<a href="#">smarty_function_debug()</a>	435
Smarty {debug} function plugin	
<a href="#">smarty_function_html_select_time()</a>	448
Smarty {html_select_time} function plugin	
<a href="#">smarty_function_html_select_date()</a>	446
Smarty {html_select_date} plugin	
<a href="#">smarty_function_html_radios_output()</a>	445
<a href="#">smarty_function_html_table()</a>	449
Smarty {html_table} function plugin	
<a href="#">smarty_function_html_table_cycle()</a>	450
<a href="#">smarty_function_math()</a>	453
Smarty {math} function plugin	
<a href="#">smarty_function_mailto()</a>	451
Smarty {mailto} function plugin	
<a href="#">smarty_function_html_radios()</a>	444
Smarty {html_radios} function plugin	
<a href="#">smarty_function_html_options_optoutput()</a>	443
<a href="#">smarty_function_html_checkboxes()</a>	438
Smarty {html_checkboxes} function plugin	
<a href="#">smarty_function_fetch()</a>	437
Smarty {fetch} plugin	
<a href="#">smarty_function_eval()</a>	436
Smarty {eval} function plugin	
<a href="#">smarty_function_html_checkboxes_output()</a>	439
<a href="#">smarty_function_html_image()</a>	440
Smarty {html_image} function plugin	
<a href="#">smarty_function_html_options_optgroup()</a>	442
<a href="#">smarty_function_html_options()</a>	442
Smarty {html_options} function plugin	
<a href="#">summary_form::get_header2()</a>	550
<a href="#">showImage()</a>	626

<i>Allows png's with alpha transparency to be displayed in IE 6</i>	
<u>STATE_LOGICBLOCK</u>	878
<i>currently parsing a {} block</i>	
<u>STATE_INLINE_DOCKEYWORD</u>	878
<i>currently parsing an inline tag like { @link} in a DocBlock</i>	
<u>STATE_INCLUDE_PARAMS_PARENTHESIS</u>	878
<i>currently parsing an inner parenthetical statement of an include/includeonce/require/requireonce( )</i>	
<u>STATE_METHOD</u>	878
<i>currently parsing a method using the HighlightParser</i>	
<u>STATE_METHOD_LOGICBLOCK</u>	878
<i>currently parsing the method body using the HighlightParser</i>	
<u>STATE_PHPCODE</u>	879
<i>currently parsing php code</i>	
<u>STATE_OUTPHP</u>	878
<i>currently out of php code</i>	
<u>STATE_NOEVENTS</u>	878
<i>out of &lt; ?php tag</i>	
<u>STATE_INCLUDE_PARAMS</u>	878
<i>currently parsing the stuff in ( ) of a define statement</i>	
<u>STATE_INCLUDE</u>	878
<i>currently parsing an include/require/include_once/include_once</i>	
<u>STATE_FUNCTION_PARAM_VAR</u>	877
<i>currently parsing a \$param in a function definition</i>	
<u>STATE_FUNCTION_PARAMS</u>	877
<i>currently parsing the stuff in ( ) of a function definition</i>	
<u>STATE_FUNCTION</u>	877
<i>currently parsing a Function or Method</i>	
<u>STATE_FUNC_GLOBAL</u>	878
<i>currently parsing a "global \$var1, \$var2;" declaration in a function</i>	
<u>STATE_GLOBAL</u>	878
<i>currently parsing a global variable declaration</i>	
<u>STATE_IMPLEMENTS</u>	878
<i>currently parsing an implements clause</i>	
<u>STATE_GLOBAL_VALUE</u>	878
<i>currently parsing the default value in a global variable declaration</i>	
<u>STATE_QUOTE</u>	879
<i>currently parsing a quote</i>	
<u>STATE_QUOTE_VAR</u>	879
<i>currently parsing a {\$encapsulated_var} using the HighlightParser</i>	
<u>STATE_TUTORIAL_ENTITY</u>	883
<i>currently parsing an &amp;entity;</i>	
<u>STATE_TUTORIAL_DOUBLEQUOTE</u>	883
<i>currently parsing a &lt;!-- comment --&gt;</i>	
<u>STATE_TUTORIAL_COMMENT</u>	883
<i>currently parsing a &lt;!-- comment --&gt;</i>	
<u>STATE_TUTORIAL_ITAG</u>	883
<i>currently parsing an {@inline tag}</i>	
<u>STATE_TUTORIAL_NOEVENTS</u>	883
<i>currently in starting state</i>	
<u>STATE_TUTORIAL_SINGLEQUOTE</u>	883
<i>currently parsing a &lt;!-- comment --&gt;</i>	
<u>STATE_TUTORIAL_OPENTAG</u>	883

<i>currently parsing an open &lt;tag&gt;</i>	
<a href="#">STATE TUTORIAL_CLOSETAG</a>	883
<i>currently parsing a close &lt;/tag&gt;</i>	
<a href="#">STATE TUTORIAL_ATTRIBUTE</a>	883
<i>currently parsing an open &lt;tag&gt;</i>	
<a href="#">STATE STATIC_VAR_VALUE</a>	879
<i>currently parsing the value in a "static \$var1 = x" declaration in a function</i>	
<a href="#">STATE STATIC_VAR</a>	879
<i>currently parsing a "static \$var1, \$var2;" declaration in a function</i>	
<a href="#">STATE_SINGLEQUOTE</a>	879
<i>currently parsing a string enclosed in single quotes ('')</i>	
<a href="#">STATE_TAGS</a>	879
<i>currently parsing the @tag block of a docblock</i>	
<a href="#">STATE_VAR</a>	879
<i>currently parsing a Class variable</i>	
<a href="#">STATE_VAR_ARRAY_COMMENT</a>	879
<i>currently parsing a comment in a variable array value</i>	
<a href="#">STATE_VAR_ARRAY</a>	879
<i>currently parsing a variable value is an array</i>	
<a href="#">STATE_ESCAPE</a>	877
<i>used when a backslash is encountered in parsing a string or other escapable entity</i>	
<a href="#">STATE_EOFQUOTE</a>	877
<i>currently parsing a string defined using Perl &lt;&lt;&lt;</i>	
<a href="#">STATE_XMLTOKEN_ENTITY</a>	866
<i>currently parsing an &amp;entity;</i>	
<a href="#">STATE_XMLTOKEN_DOUBLEQUOTE</a>	866
<i>currently parsing a &lt;!-- comment --&gt;</i>	
<a href="#">STATE_XMLTOKEN_DEF</a>	866
<i>currently parsing a &lt;!</i>	
<a href="#">STATE_XMLTOKEN_IN_CDATA</a>	866
<i>currently parsing a &lt;![CDATA[ ]]&gt;</i>	
<a href="#">STATE_XMLTOKEN_NOEVENTS</a>	866
<i>currently in starting state</i>	
<a href="#">STATE_XMLTOKEN_SINGLEQUOTE</a>	866
<i>currently parsing a &lt;!-- comment --&gt;</i>	
<a href="#">STATE_XMLTOKEN_PI</a>	866
<i>currently in processor instruction</i>	
<a href="#">STATE_XMLTOKEN_OPENTAG</a>	866
<i>currently parsing an open &lt;tag&gt;</i>	
<a href="#">STATE_XMLTOKEN_COMMENT</a>	866
<i>currently parsing a &lt;!-- comment --&gt;</i>	
<a href="#">STATE_XMLTOKEN_CLOSETAG</a>	866
<i>currently parsing a close &lt;/tag&gt;</i>	
<a href="#">setup_dirs()</a>	641
<i>Recursively move contents of \$struc into associative array</i>	
<a href="#">SMART_PATH_DELIMITER</a>	637
<a href="#">switchDirTree()</a>	626
<a href="#">set_dir()</a>	641
<i>Recursively add all the subdirectories of \$contents to \$dir without erasing anything in Setup.inc.php</i>	
<i>This was all in <a href="#">phpdoc.inc</a>, and now encapsulates the complexity</i>	
<a href="#">STATE_XMLTOKEN_CDATA</a>	866
<i>currently parsing a &lt;!</i>	

<a href="#"><u>STATE XMLTOKEN ATTRIBUTE</u></a>	. . . . .	866
<i>currently parsing an open &lt;tag&gt;</i>		
<a href="#"><u>STATE XMLTOKEN XML</u></a>	. . . . .	867
<i>currently parsing a &lt;?xml</i>		
<a href="#"><u>STATE ACCESS MODIFIER</u></a>	. . . . .	876
<i>currently parsing an access modifier</i>		
<a href="#"><u>STATE DOCBLOCK TEMPLATE</u></a>	. . . . .	877
<i>currently parsing the value in a "static \$var1 = x" declaration in a function</i>		
<a href="#"><u>STATE DOCBLOCK</u></a>	. . . . .	877
<i>currently parsing a DocBlock</i>		
<a href="#"><u>STATE DESC</u></a>	. . . . .	877
<i>currently parsing the desc part of a docblock</i>		
<a href="#"><u>STATE DOCKEYWORD</u></a>	. . . . .	877
<i>currently parsing a @tag in a DocBlock</i>		
<a href="#"><u>STATE DOCKEYWORD EMAIL</u></a>	. . . . .	877
<i>currently parsing an email in brackets in an @author tag of a DocBlock</i>		
<a href="#"><u>STATE END_DOCBLOCK TEMPLATE</u></a>	. . . . .	877
<i>currently parsing the value in a "static \$var1 = x" declaration in a function</i>		
<a href="#"><u>STATE END CLASS</u></a>	. . . . .	877
<i>used to tell Render that a class has been completely parsed, and to flush buffers</i>		
<a href="#"><u>STATE DEFINE PARAMS PARENTHESIS</u></a>	. . . . .	876
<i>currently parsing an inner parenthetical statement of a define()</i>		
<a href="#"><u>STATE DEFINE PARAMS</u></a>	. . . . .	876
<i>currently parsing the stuff in () of a define statement</i>		
<a href="#"><u>STATE CLASS CONSTANT</u></a>	. . . . .	876
<i>currently parsing a class constant</i>		
<a href="#"><u>STATE CLASS</u></a>	. . . . .	876
<i>currently parsing a class definition</i>		
<a href="#"><u>STATE ARRAY</u></a>	. . . . .	876
<i>currently parsing an array</i>		
<a href="#"><u>STATE CLASS MEMBER</u></a>	. . . . .	876
<i>currently parsing a class member using the HighlightParser</i>		
<a href="#"><u>STATE COMMENT</u></a>	. . . . .	876
<i>currently parsing a short comment //</i>		
<a href="#"><u>STATE DEFINE</u></a>	. . . . .	876
<i>currently parsing a define statement</i>		
<a href="#"><u>STATE COMMENTBLOCK</u></a>	. . . . .	876
<i>currently parsing a long comment /x x/ where x is an asterisk</i>		
<a href="#"><u>smarty_core_load_resource_plugin()</u></a>	. . . . .	415
<i>load a resource plugin</i>		
<a href="#"><u>smarty_core_load_plugins()</u></a>	. . . . .	414
<i>Load requested plugins</i>		
<a href="#"><u>Smarty::append()</u></a>	. . . . .	383
<i>appends values to template variables</i>		
<a href="#"><u>Smarty::\$use_sub_dirs</u></a>	. . . . .	382
<i>This tells Smarty whether or not to use sub dirs in the cache/ and templates_c/ directories. sub directories better organized, but may not work well with PHP safe mode enabled.</i>		
<a href="#"><u>Smarty::\$undefined</u></a>	. . . . .	382
<i>The value of "undefined". Leave it alone :-)</i>		
<a href="#"><u>Smarty::append_by_ref()</u></a>	. . . . .	383
<i>appends values to template variables by reference</i>		
<a href="#"><u>Smarty::assign()</u></a>	. . . . .	383

<i>assigns values to template variables</i>	
<a href="#">Smarty::clear_all_cache()</a>	384
<i>clear the entire contents of cache (all templates)</i>	
<a href="#">Smarty::clear_all_assign()</a>	384
<i>clear all the assigned template variables.</i>	
<a href="#">Smarty::assign_by_ref()</a>	383
<i>assigns values to template variables by reference</i>	
<a href="#">Smarty::\$trusted_dir</a>	382
<i>This is an array of directories where trusted php scripts reside.</i>	
<a href="#">Smarty::\$template_dir</a>	382
<i>The name of the directory where templates are located.</i>	
<a href="#">Smarty::\$request_vars_order</a>	381
<i>The order in which request variables are registered, similar to variables_order in php.ini E = Environment, G = GET, P = POST, C = Cookies, S = Server</i>	
<a href="#">Smarty::\$request_use_auto_globals</a>	380
<i>Indicates whether \$HTTP_*_VARS[] (request_use_auto_globals=false)</i>	
<a href="#">Smarty::\$plugins_dir</a>	380
<i>An array of directories searched for plugins.</i>	
<a href="#">Smarty::\$right_delimiter</a>	381
<i>The right delimiter used for the template tags.</i>	
<a href="#">Smarty::\$secure_dir</a>	381
<i>This is the list of template directories that are considered secure. This is used only if \$security is enabled. One directory per array element. \$template_dir is in this list implicitly.</i>	
<a href="#">Smarty::\$security_settings</a>	381
<i>These are the security settings for Smarty. They are used only when \$security is enabled.</i>	
<a href="#">Smarty::\$security</a>	381
<i>This enables template security. When enabled, many things are restricted</i>	
<a href="#">Smarty::clear_assign()</a>	384
<i>clear the given assigned template variable.</i>	
<a href="#">Smarty::clear_cache()</a>	384
<i>clear cached content for the given template and cache id</i>	
<a href="#">Smarty::register_compiler_function()</a>	387
<i>Registers compiler function</i>	
<a href="#">Smarty::register_block()</a>	387
<i>Registers block function to be used in templates</i>	
<a href="#">Smarty::load_filter()</a>	386
<i>load a filter of specified type and name</i>	
<a href="#">Smarty::register_function()</a>	387
<i>Registers custom function to be used in templates</i>	
<a href="#">Smarty::register_modifier()</a>	388
<i>Registers modifier to be used in templates</i>	
<a href="#">Smarty::register_postfilter()</a>	388
<i>Registers a postfilter function to apply to a compiled template after compilation</i>	
<a href="#">Smarty::register_outputfilter()</a>	388
<i>Registers an output filter function to apply to a template output</i>	
<a href="#">Smarty::register_object()</a>	388
<i>Registers object to be used in templates</i>	
<a href="#">Smarty::is_cached()</a>	386

<i>test to see if valid cache exists for this template</i>	
<a href="#">Smarty::get_template_vars()</a>	386
<i>Returns an array containing template variables</i>	
<a href="#">Smarty::config_load()</a>	385
<i>load configuration values</i>	
<a href="#">Smarty::clear_config()</a>	385
<i>clear configuration values</i>	
<a href="#">Smarty::clear_compiled_tpl()</a>	384
<i>clears compiled version of specified template resource,     or all compiled template files if one is not specified.</i>	
<a href="#">Smarty::display()</a>	385
<i>executes &amp; displays the template results</i>	
<a href="#">Smarty::fetch()</a>	385
<i>executes &amp; returns or displays the template results</i>	
<a href="#">Smarty::get_registered_object()</a>	386
<i>return a reference to a registered object</i>	
<a href="#">Smarty::get_config_vars()</a>	386
<i>Returns an array containing config variables</i>	
<a href="#">Smarty::\$php_handling</a>	380
<i>This determines how Smarty handles "&lt;?php ... ?&gt;" tags in templates.</i>	
<a href="#">Smarty::\$left_delimiter</a>	380
<i>The left delimiter used for the template tags.</i>	
<a href="#">Smarty::\$cache_dir</a>	375
<i>The name of the directory for cache files.</i>	
<a href="#">Smarty::\$autoload_filters</a>	374
<i>This indicates which filters are automatically loaded into Smarty.</i>	
<a href="#">Smarty</a>	374
<a href="#">Smarty::\$cache_handler_func</a>	375
<i>The function used for cache file handling. If not set, built-in caching is used.</i>	
<a href="#">Smarty::\$cache_lifetime</a>	375
<i>This is the number of seconds cached content will persist.</i>	
<a href="#">Smarty::\$compiler_class</a>	376
<i>The class used for compiling templates.</i>	
<a href="#">Smarty::\$caching</a>	375
<i>This enables template caching.</i>	
<a href="#">Smarty::\$cache_modified_check</a>	375
<i>Only used when \$caching is enabled. If true, then If-Modified-Since headers     are respected with cached content, and appropriate HTTP headers are sent.</i>	
<a href="#">Smarty_Compiler.class.php</a>	370
<i>Project: Smarty: the PHP compiling template engine     File: Smarty_Compiler.class.php</i>	
<a href="#">SMARTY_PHP_REMOVE</a>	369
<a href="#">Smarty.class.php</a>	368
<i>Project: Smarty: the PHP compiling template engine     File: Smarty.class.php</i>	
<a href="#">Source code for sample3.php</a>	15
<a href="#">Source code for sample2.php</a>	12
<a href="#">SMARTY_DIR</a>	368
<i>set SMARTY_DIR to absolute path to Smarty library files.</i>	
<a href="#">SMARTY_PHP_ALLOW</a>	369
<a href="#">SMARTY_PHP_QUOTE</a>	369
<a href="#">SMARTY_PHP_PASSTHRU</a>	369
<a href="#">Smarty::\$compiler_file</a>	376

<i>The file that contains the compiler class. This can a full pathname, or relative to the php_include path.</i>	
<a href="#">Smarty::\$compile_check</a>	376
<i>This tells Smarty whether to check for recompiling or not. Recompiling does not need to happen unless a template or config file is changed.</i>	
<a href="#">Smarty::\$default_modifiers</a>	378
<i>This is a list of the modifiers to apply to all template variables.</i>	
<a href="#">Smarty::\$debug_tpl</a>	378
<i>This is the path to the debug console template. If not set, the default one will be used.</i>	
<a href="#">Smarty::\$debugging_ctrl</a>	378
<i>This determines if debugging is enable-able from the browser.</i>	
<a href="#">Smarty::\$default_resource_type</a>	379
<i>This is the resource type to be used when not specified</i>	
<a href="#">Smarty::\$default_template_handler_func</a>	379
<i>If a template cannot be found, this PHP function will be executed.</i>	
<a href="#">Smarty::\$global_assign</a>	379
<i>These are the variables from the globals array that are assigned to all templates automatically. This isn't really necessary any more, you can use the \$smarty var to access them directly.</i>	
<a href="#">Smarty::\$force_compile</a>	379
<i>This forces templates to compile every time. Useful for development or debugging.</i>	
<a href="#">Smarty::\$debugging</a>	378
<i>If debugging is enabled, a debug console window will display when the page loads (make sure your browser allows unrequested popup windows)</i>	
<a href="#">Smarty::\$config_read_hidden</a>	377
<i>This tells whether hidden sections [foobar] are readable from the templates or not. Normally you would never allow this since that is the point behind hidden sections: the application can access them, but the templates cannot.</i>	
<a href="#">Smarty::\$config_booleanize</a>	377
<i>This tells whether or not to automatically booleanize config file variables.</i>	
<a href="#">Smarty::\$compile_id</a>	376
<i>Set this if you want different sets of compiled files for the same templates. This is useful for things like different languages.</i>	
<a href="#">Smarty::\$compile_dir</a>	376
<i>The directory where compiled templates are located.</i>	
<a href="#">Smarty::\$config_class</a>	377
<i>The class used to load config vars.</i>	
<a href="#">Smarty::\$config_dir</a>	377
<i>The directory where config files are located.</i>	
<a href="#">Smarty::\$config_overwrite</a>	377
<i>This tells if config file vars of the same name overwrite each other or not.</i>	
<a href="#">Smarty::\$config_fix_newlines</a>	377
<i>This tells whether or not automatically fix newlines in config files.</i>	
<a href="#">Smarty::register_prefilter()</a>	388
<i>Registers a prefilter function to apply to a template before compiling</i>	
<a href="#">Smarty::register_resource()</a>	389
<i>Registers a resource to fetch a template</i>	

<a href="#">Smarty Compiler:: expand quoted text()</a>	401
<i>expand quoted text with embedded variables</i>	
<a href="#">Smarty Compiler:: compile_tag()</a>	400
<i>Compile a template tag</i>	
<a href="#">Smarty Compiler:: compile_smarty_ref()</a>	400
<i>Compiles references of type \$smarty.foo</i>	
<a href="#">Smarty Compiler:: load_filters()</a>	401
<i>load pre- and post-filters</i>	
<a href="#">Smarty Compiler:: parse_attrs()</a>	401
<i>Parse attribute string</i>	
<a href="#">Smarty Compiler:: parse_modifiers()</a>	402
<i>parse modifier chain into PHP code</i>	
<a href="#">Smarty Compiler:: parse_is_expr()</a>	401
<i>Parse is expression</i>	
<a href="#">Smarty Compiler:: parse_conf_var()</a>	401
<i>parse configuration variable expression into PHP code</i>	
<a href="#">Smarty Compiler:: compile_section_start()</a>	400
<i>Compile {section ...} tag</i>	
<a href="#">Smarty Compiler:: compile_registered_object_tag()</a>	400
<i>compile a registered object tag</i>	
<a href="#">Smarty Compiler:: compile_if_tag()</a>	399
<i>Compile {if ...} tag</i>	
<a href="#">Smarty Compiler:: compile_foreach_start()</a>	398
<i>Compile {foreach ...} tag.</i>	
<a href="#">Smarty Compiler:: compile_file()</a>	398
<i>compile a resource</i>	
<a href="#">Smarty Compiler:: compile_include_php_tag()</a>	399
<i>Compile {include ...} tag</i>	
<a href="#">Smarty Compiler:: compile_include_tag()</a>	399
<i>Compile {include ...} tag</i>	
<a href="#">Smarty Compiler:: compile_plugin_call()</a>	399
<i>compiles call to plugin of type \$type with name \$name</i>	
<a href="#">Smarty Compiler:: compile_insert_tag()</a>	399
<i>Compile {insert ...} tag</i>	
<a href="#">Smarty Compiler:: parse_parenth_args()</a>	402
<i>parse arguments in function call parenthesis</i>	
<a href="#">Smarty Compiler:: parse_section_prop()</a>	402
<i>parse section property expression into PHP code</i>	
<a href="#">smarty_core_get_include_path()</a>	409
<i>Get path to file from include_path</i>	
<a href="#">smarty_core_display_debug_console()</a>	408
<i>Smarty debug_console function plugin</i>	
<a href="#">smarty_core_create_dir_structure()</a>	407
<i>create full directory structure</i>	
<a href="#">smarty_core_get_microtime()</a>	410
<i>Get seconds and microseconds</i>	
<a href="#">smarty_core_get_php_resource()</a>	411
<i>Retrieves PHP script resource</i>	
<a href="#">smarty_core_is_trusted()</a>	413
<i>determines if a resource is trusted or not</i>	
<a href="#">smarty_core_is_secure()</a>	412
<i>determines if a resource is secure or not.</i>	
<a href="#">smarty_core_assign_smarty_interface()</a>	406

<i>Smarty assign_smarty_interface core plugin</i>	
<a href="#"><u>smarty_core_assemble_plugin_filepath()</u></a>	405 <i>assemble filepath of requested plugin</i>
<a href="#"><u>Smarty_Compiler::parse_var_props()</u></a>	403 <i>compile single variable and section properties token into PHP code</i>
<a href="#"><u>Smarty_Compiler::parse_vars_props()</u></a>	403 <i>compile multiple variables and section properties tokens into PHP code</i>
<a href="#"><u>Smarty_Compiler::parse_var()</u></a>	402 <i>parse variable expression into PHP code</i>
<a href="#"><u>Smarty_Compiler::pop_cacheable_state()</u></a>	403 <i>check if the compilation changes from non-cacheable to cacheable state with the end of the current plugin return php-code to reflect the transition.</i>
<a href="#"><u>Smarty_Compiler::push_cacheable_state()</u></a>	403 <i>check if the compilation changes from cacheable to non-cacheable state with the beginning of the current plugin. return php-code to reflect the transition.</i>
<a href="#"><u>Smarty_Compiler::syntax_error()</u></a>	404 <i>display Smarty syntax error</i>
<a href="#"><u>Smarty_Compiler::quote_replace()</u></a>	403 <i>Quote subpattern references</i>
<a href="#"><u>Smarty_Compiler::compile_custom_tag()</u></a>	398 <i>compile custom function tag</i>
<a href="#"><u>Smarty_Compiler::compile_compiler_tag()</u></a>	398 <i>compile the custom compiler tag</i>
<a href="#"><u>Smarty::compile_resource()</u></a>	391 <i>compile the template</i>
<a href="#"><u>Smarty::unregister_resource()</u></a>	391 <i>Unregisters a resource</i>
<a href="#"><u>Smarty::unregister_prefilter()</u></a>	391 <i>Unregisters a prefilter function</i>
<a href="#"><u>Smarty::compile_source()</u></a>	391 <i>compile the given source</i>
<a href="#"><u>Smarty::dequote()</u></a>	392 <i>Remove starting and ending quotes from the string</i>
<a href="#"><u>Smarty::get_auto_filename()</u></a>	393 <i>get a concrete filename for automagically created content</i>
<a href="#"><u>Smarty::fetch_resource_info()</u></a>	392 <i>fetch the template info. Gets timestamp, and source if get_source is true</i>
<a href="#"><u>Smarty::eval()</u></a>	392 <i>wrapper for eval() retaining \$this</i>
<a href="#"><u>Smarty::unregister_postfilter()</u></a>	391 <i>Unregisters a postfilter function</i>
<a href="#"><u>Smarty::unregister_outputfilter()</u></a>	390 <i>UnRegisters an outputfilter function</i>
<a href="#"><u>Smarty::unregister_block()</u></a>	389 <i>UnRegisters block function</i>
<a href="#"><u>Smarty::trigger_error()</u></a>	389 <i>trigger Smarty error</i>
<a href="#"><u>Smarty::template_exists()</u></a>	389

<i>Checks whether requested template exists.</i>	
<a href="#">Smarty::unregister_compiler_function()</a>	390
<i>Unregisters compiler function</i>	
<a href="#">Smarty::unregister_function()</a>	390
<i>UnRegisters custom function</i>	
<a href="#">Smarty::unregister_object()</a>	390
<i>UnRegisters object</i>	
<a href="#">Smarty::unregister_modifier()</a>	390
<i>UnRegisters modifier</i>	
<a href="#">Smarty::get_auto_id()</a>	393
<i>returns an auto_id for auto-file-functions</i>	
<a href="#">Smarty::get_compile_path()</a>	393
<i>Get the compile path for this resource</i>	
<a href="#">Smarty_Compiler</a>	396
<i>Template compiling class</i>	
<a href="#">Smarty::unlink()</a>	396
<i>unlink a file, possibly using expiration time</i>	
<a href="#">Smarty::trigger_fatal_error()</a>	396
<i>trigger Smarty plugin error</i>	
<a href="#">Smarty_Compiler::add_plugin()</a>	397
<i>add plugin</i>	
<a href="#">Smarty_Compiler::compile_arg_list()</a>	397
<a href="#">Smarty_Compiler::compile_capture_tag()</a>	397
<i>Compile {capture}..{/capture} tags</i>	
<a href="#">Smarty_Compiler::compile_block_tag()</a>	397
<i>compile block function tag</i>	
<a href="#">Smarty::smarty_include()</a>	395
<i>called for included templates</i>	
<a href="#">Smarty::smarty_cache_attrs()</a>	395
<i>get or set an array of cached attributes for function that is</i>	
<a href="#">Smarty::is_compiled()</a>	394
<i>test if resource needs compiling</i>	
<a href="#">Smarty::include()</a>	394
<i>wrapper for include() retaining \$this</i>	
<a href="#">Smarty::get_plugin_filepath()</a>	393
<i>get filepath of requested plugin</i>	
<a href="#">Smarty::parse_resource_name()</a>	394
<i>parse out the type and name from the resource</i>	
<a href="#">Smarty::process_compiled_include_callback()</a>	394
<i>callback function for preg_replace, to call a non-cacheable block</i>	
<a href="#">Smarty::run_mod_handler()</a>	395
<i>Handle modifiers</i>	
<a href="#">Smarty::read_file()</a>	395
<i>read in a file from line \$start for \$lines.</i>	
<a href="#">Source code for sample1.php</a>	10

## T

<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyNoC()</a>	604
<i>Shows correct behavior for handling the fuzzy "NO" value</i>	
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyNoB()</a>	603
<i>Shows correct behavior for handling the fuzzy "No" value</i>	

<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOffA()</a>	Shows correct behavior for handling the fuzzy "Off" value	604
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOffB()</a>	Shows correct behavior for handling the fuzzy "OFF" value	604
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOnB()</a>	Shows correct behavior for handling the fuzzy "ON" value	605
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOnA()</a>	Shows correct behavior for handling the fuzzy "On" value	604
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyNoA()</a>	Shows correct behavior for handling the fuzzy "no" value	603
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyFalseC()</a>	Shows correct behavior for handling the fuzzy "FALSE" value	603
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testBasicOff()</a>	Shows correct behavior for handling the perfect expected "off" value	602
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::tearDown()</a>	Tears down the fixture, for example, close a network connection.	601
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testBasicOn()</a>	Shows correct behavior for handling the perfect expected "on" value	602
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyEmpty()</a>	Shows correct behavior for handling the fuzzy "" value	602
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyFalseB()</a>	Shows correct behavior for handling the fuzzy "False" value	603
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyFalseA()</a>	Shows correct behavior for handling the fuzzy "false" value	602
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyOne()</a>	Shows correct behavior for handling the fuzzy "1" value	605
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyTrueA()</a>	Shows correct behavior for handling the fuzzy "true" value	605
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedGreatLiterature()</a>	Shows correct behavior for handling an odd "ash nazg durbatuluk" value	607
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyZero()</a>	Shows correct behavior for handling the fuzzy "0" value	607
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedLargeNumber()</a>	Shows correct behavior for handling an odd "10" value	607
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedNegative()</a>	Shows correct behavior for handling an odd "-1" value	608
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedSpaces()</a>	Shows correct behavior for handling an odd " " value	608
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testUnexpectedNull()</a>	Shows correct behavior for handling an odd NULL value	608
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesE()</a>	Shows correct behavior for handling the fuzzy "YES" value	607
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesD()</a>	Shows correct behavior for handling the fuzzy "Yes" value	606
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyTrueC()</a>	Shows correct behavior for handling the fuzzy "TRUE" value	605
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyTrueB()</a>	Shows correct behavior for handling the fuzzy "True" value	605
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesA()</a>	Shows correct behavior for handling the fuzzy "y" value	606
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesB()</a>	Shows correct behavior for handling the fuzzy "Y" value	606
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::testFuzzyYesC()</a>	Shows correct behavior for handling the fuzzy "Yes" value	606

<i>Shows correct behavior for handling the fuzzy "yes" value</i>	
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::setUp()</a>	601
<i>Sets up the fixture, for example, open a network connection.</i>	
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests::main()</a>	601
<i>Runs the test methods of this class.</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryHands()</a>	595
<i>Shows correct behavior for handling the perfect expected "HandS" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryEarthli()</a>	594
<i>Shows correct behavior for handling the perfect expected "earthli" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryL0l33t()</a>	595
<i>Shows correct behavior for handling the perfect expected "l0l33t" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPear()</a>	595
<i>Shows correct behavior for handling the perfect expected "PEAR" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPhpdocde()</a>	596
<i>Shows correct behavior for handling the perfect expected "phpdoc.de" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPhp()</a>	595
<i>Shows correct behavior for handling the perfect expected "PHP" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomPhptmllib()</a>	594
<i>Shows correct behavior for handling the perfect expected "DOM/phptmllib" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomPhpdocde()</a>	594
<i>Shows correct behavior for handling the perfect expected "DOM/phpdoc.de" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondarySmarty()</a>	592
<i>Shows correct behavior for handling the perfect expected "Smarty" secondary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondaryFrames()</a>	592
<i>Shows correct behavior for handling the perfect expected "frames" secondary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDefault()</a>	592
<i>Shows correct behavior for handling the perfect expected "default" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomDefault()</a>	593
<i>Shows correct behavior for handling the perfect expected "DOM/default" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomL0l33t()</a>	593
<i>Shows correct behavior for handling the perfect expected "DOM/l0l33t" tertiary value</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryDomEarthli()</a>	593
<i>Shows correct behavior for handling the perfect expected "DOM/earthli" tertiary value</i>	

<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalTertiaryPhphmlib()</a>	596
Shows correct behavior for handling the perfect expected "phphmlib" tertiary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryCHM()</a>	596
Shows correct behavior for handling the perfect expected "CHM" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnPrimaryWithTwoArgs()</a>	599
Verify no up-to-parent pathing is allowed...	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnPrimaryWithOneArg()</a>	599
Verify no up-to-parent pathing is allowed...	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnSecondary()</a>	600
Verify no up-to-parent pathing is allowed...	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testPreventUpToParentPathingOnTertiary()</a>	600
Verify no up-to-parent pathing is allowed...	
<a href="#">tests_phpDocumentorSetupDecideOnOrOffTests</a>	601
Unit Testing of the phpDocumentor_setup's decideOnOrOff() method	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testUserDefinedTertiaryValue()</a>	600
Shows correct behavior for handling the perfect expected "b2evo.v-1-10" tertiary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryXML()</a>	599
Shows correct behavior for handling the perfect expected "XML" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryPDF()</a>	598
Shows correct behavior for handling the perfect expected "PDF" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryPDF()</a>	597
Shows correct behavior for handling the perfect expected "PDF" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryHTML()</a>	597
Shows correct behavior for handling the perfect expected "HTML" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithOneArgPrimaryXML()</a>	597
Shows correct behavior for handling the perfect expected "XML" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryCHM()</a>	598
Shows correct behavior for handling the perfect expected "CHM" primary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalWithTwoArgsPrimaryHTML()</a>	598
Shows correct behavior for handling the perfect expected "HTML" primary value	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests</a>	608
Unit Testing of the phpDocumentorTParser's getInlineTags() method	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::main()</a>	609
Runs the test methods of this class.	
<a href="#">T_DOC_COMMENT</a>	879

<a href="#">T CONST</a>	879
<a href="#">T FINAL</a>	879
<a href="#">T IMPLEMENTS</a>	879
<a href="#">T ML COMMENT</a>	879
<a href="#">T INTERFACE</a>	879
<a href="#">T ABSTRACT</a>	879
<a href="#">Tokenizer.php</a>	864
<a href="#">XML/Beautifier.php</a>	
<a href="#">tutorialLink</a>	794
<a href="#">tutorial link</a>	
<a href="#">tests_phpDocumentorTParserTests::suite()</a>	713
<a href="#">tutorialLink::\$section</a>	795
<a href="#">tutorialLink::\$title</a>	795
<a href="#">tutorialLink::addLink()</a>	795
<a href="#">sets up the link</a>	
<a href="#">tutorialLink::\$type</a>	795
<a href="#">T PRIVATE</a>	879
<a href="#">T PROTECTED</a>	879
<a href="#">TUTORIAL_EVENT_NOEVENTS</a>	884
<a href="#">starting state</a>	
<a href="#">TUTORIAL_EVENT_ITAG</a>	884
<a href="#">used when an {@inline tag} is found</a>	
<a href="#">TUTORIAL_EVENT_OPENTAG</a>	884
<a href="#">used when an open &lt;tag&gt; is found</a>	
<a href="#">TUTORIAL_EVENT_SINGLEQUOTE</a>	884
<a href="#">used when a &lt;!-- comment --&gt; is found</a>	
<a href="#">Tokenizer.php</a>	1313
<a href="#">Source code</a>	
<a href="#">top.php</a>	1181
<a href="#">Source code</a>	
<a href="#">TUTORIAL_EVENT_ENTITY</a>	884
<a href="#">used when an &amp;entity; is found</a>	
<a href="#">TUTORIAL_EVENT_DOUBLEQUOTE</a>	884
<a href="#">used when a &lt;!-- comment --&gt; is found</a>	
<a href="#">TutorialHighlightParser.inc</a>	882
<a href="#">Source Code Highlighting</a>	
<a href="#">T PUBLIC</a>	880
<a href="#">TUTORIAL_EVENT_ATTRIBUTE</a>	883
<a href="#">used when a &lt;tag attr="attribute"&gt; is found</a>	
<a href="#">TUTORIAL_EVENT_CLOSETAG</a>	883
<a href="#">used when a close &lt;/tag&gt; is found</a>	
<a href="#">TUTORIAL_EVENT_COMMENT</a>	883
<a href="#">used when a &lt;!-- comment --&gt; is found</a>	
<a href="#">tests_phpDocumentorTParserTests::main()</a>	713
<a href="#">tests_phpDocumentorTParserTests</a>	713
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineSourceWithParsePrivateOn()</a>	612
<a href="#">Shows correct behavior for handling an inline source tag e.g. {at-source}</a>	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineSourceWithParsePrivateOff()</a>	611
<a href="#">Shows correct behavior for handling an inline source tag e.g. {at-source}</a>	

<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineTutorialWhenParsePrivateOff()</a>	612
Shows correct behavior for handling an <i>inline tutorial</i> tag	
e.g. {at-tutorial path-to-pkgfile}	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineTutorialWhenParsePrivateOn()</a>	612
Shows correct behavior for handling an <i>inline tutorial</i> tag	
e.g. {at-tutorial path-to-pkgfile}	
<a href="#">tokenizer_ext</a>	637
<a href="#">top.php</a>	628
phpDocumentor :: docBuilder Web Interface	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineLinkWhenParsePrivateOn()</a>	611
Shows correct behavior for handling an <i>inline link</i> tag	
e.g. {at-link URL link-text}	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineLinkWhenParsePrivateOff()</a>	611
Shows correct behavior for handling an <i>inline link</i> tag	
e.g. {at-link URL link-text}	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::tearDown()</a>	609
Tears down the fixture, for example, close a network connection.	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::setUp()</a>	609
Sets up the fixture, for example, open a network connection.	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineExampleWhenParsePrivateOff()</a>	610
Shows correct behavior for handling an <i>inline example</i> tag	
e.g. {at-example path-to-file}	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineExampleWhenParsePrivateOn()</a>	610
Shows correct behavior for handling an <i>inline example</i> tag	
e.g. {at-example path-to-file}	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineInternalWhenParsePrivateOn()</a>	611
Shows correct behavior for handling an <i>inline internal</i> tag	
e.g. {at-internal blah-blah-blah}	
<a href="#">tests_phpDocumentorTParserGetInlineTagsTests::testShowCorrectBehaviorInlineInternalWhenParsePrivateOff()</a>	610
Shows correct behavior for handling an <i>inline internal</i> tag	
e.g. {at-internal blah-blah-blah}	
<a href="#">tests_HighlightParserTests</a>	710
<a href="#">tests_HighlightParserTests::main()</a>	710
<a href="#">tests_ParserPageTests::main()</a>	712
<a href="#">tests_ParserPageTests</a>	712
<a href="#">tests_ParserPageTests::suite()</a>	712
<a href="#">tests_phpDocumentorSetupTests</a>	712
<a href="#">tests_phpDocumentorSetupTests::suite()</a>	713
<a href="#">tests_phpDocumentorSetupTests::main()</a>	713
<a href="#">tests_ParserClassTests::suite()</a>	712
<a href="#">tests_ParserClassTests::main()</a>	711
<a href="#">tests_IntermediateParserTests</a>	711
<a href="#">tests_HighlightParserTests::suite()</a>	710
<a href="#">tests_IntermediateParserTests::main()</a>	711
<a href="#">tests_IntermediateParserTests::suite()</a>	711

<a href="#">tests_ParserClassTests</a>	711
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondaryDocbookPeardoc2()</a>	591
Shows correct behavior for handling the perfect expected "DocBook/peardoc2" secondary value	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testNormalSecondaryDefault()</a>	591
Shows correct behavior for handling the perfect expected "default" secondary value	
<a href="#">testClass</a>	552
<a href="#">testarraybug::\$myarrayName1</a>	552
<i>test with no default, should be myarrayName1</i>	
<a href="#">testme</a>	552
<a href="#">testme::\$me</a>	553
<a href="#">tests_HighlightParserGetInlineTagsTests</a>	568
Unit Testing of the HighlightParser's getInlineTags() method	
<a href="#">test_541886</a>	554
<i>Test for bug #541886</i>	
<a href="#">testarraybug::\$myarrayName</a>	552
<a href="#">testarraybug::\$arrayType1</a>	552
<i>test with no default, should be arrayType1</i>	
<a href="#">testContantBlah5</a>	529
<i>Short Desc 2.0 Test This is still the short desc Comment and so is this</i>	
<a href="#">testContantBlah4</a>	529
<i>Short Desc 2.0 Test This is still the short desc Comment</i>	
<a href="#">test</a>	551
<i>I'm a odd test case</i>	
<a href="#">test2</a>	551
<i>tags demonstration, but this @version tag is ignored</i>	
<a href="#">testarraybug::\$arrayType</a>	552
<a href="#">testarraybug</a>	551
<i>tests variable names with the word 'array' in them</i>	
<a href="#">tests_HighlightParserGetInlineTagsTests::main()</a>	568
<i>Runs the test methods of this class.</i>	
<a href="#">tests_HighlightParserGetInlineTagsTests::setUp()</a>	568
<i>Sets up the fixture, for example, open a network connection.</i>	
<a href="#">tests_IntermediateParserAddPrivatePageTests::testShowCorrectBehaviorWhenPrivatePageArrayIsEmpty()</a>	571
<i>Show correct behavior for adding a private page object,</i>	
<a href="#">tests_IntermediateParserAddPrivatePageTests::tearDown()</a>	571
<i>Tears down the fixture, for example, close a network connection.</i>	
<a href="#">tests_IntermediateParserAddPrivatePageTests::testShowCorrectBehaviorWhenPrivatePageArrayIsNotAlreadyEmpty()</a>	571
<i>Show correct behavior for adding a private page object,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests</a>	572
Unit Testing of the ParserClass's getSourceLocation() method	
<a href="#">tests_ParserClassGetSourceLocationTests::setUp()</a>	572
<i>Sets up the fixture, for example, open a network connection.</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::main()</a>	572
<i>Runs the test methods of this class.</i>	
<a href="#">tests_IntermediateParserAddPrivatePageTests::setUp()</a>	570
<i>Sets up the fixture, for example, open a network connection.</i>	

<a href="#">tests_IntermediateParserAddPrivatePageTests::main()</a>	570
<i>Runs the test methods of this class.</i>	
<a href="#">tests_HighlightParserGetInlineTagsTests::testShowCorrectBehaviorWhenGivenOneEmptyArg()</a>	569
<i>Shows correct behavior when called with no actual value</i>	
<a href="#">tests_HighlightParserGetInlineTagsTests::tearDown()</a>	569
<i>Tears down the fixture, for example, close a network connection.</i>	
<a href="#">tests_HighlightParserGetInlineTagsTests::testShowCorrectBehaviorWhenGivenOneEmptyArgAndFalse()</a>	569
<i>Shows correct behavior when called with no actual value</i>	
<a href="#">tests_HighlightParserGetInlineTagsTests::testShowCorrectBehaviorWhenGivenOneEmptyArgAndTrue()</a>	569
<i>Shows correct behavior when called with no actual value</i>	
<a href="#">tests_IntermediateParserAddPrivatePageTests</a>	570
<i>Unit Testing of the IntermediateParser's addPrivatePage() method</i>	
<a href="#">testContantBlah3</a>	529
<i>Short Desc 2.0 Test. This is the Extended Comment</i>	
<a href="#">testContantBlah2</a>	529
<i>Short Desc 2.0 Test. This is the Extended Comment</i>	
<a href="#">test_4412893()</a>	489
<i>I am using just the first line as my short desc</i>	
<a href="#">test_4412892()</a>	489
<i>I started my short desc on the first line.</i>	
<a href="#">test_4412894()</a>	490
<i>I am using a blank line to end my short desc</i>	
<a href="#">test_4412895()</a>	490
<i>This is a test to see if the desc is fooled by 2.4 the short desc ends here.</i>	
<a href="#">test_bug_567455_2()</a>	490
<i>This example is also really tricky</i>	
<a href="#">test_bug_567455()</a>	490
<i>This example is really tricky</i>	
<a href="#">test_441289()</a>	489
<i>This is a long comment which needs more than one line.</i>	
<a href="#">test8()</a>	489
<i>This is a test case where i think things break</i>	
<a href="#">test_441278()</a>	487
<i>Test description.</i>	
<a href="#">test_441275()</a>	486
<i>This makes sure the basic element of bug 44127 is fixed</i>	
<a href="#">test_441287()</a>	488
<i>sdesc</i>	
<a href="#">test_4412872()</a>	488
<i>Check to see if we are stripping whitespace before the *</i>	
<a href="#">test7()</a>	489
<i>This desc tries to fool the non period short desc systems</i>	
<a href="#">test6()</a>	489
<i>This desc</i>	
<a href="#">test_441433()</a>	491
<i>a second test.</i>	

<a href="#">test_443153()</a>	492
<i>test function</i>	
<a href="#">test_escape()</a>	525
<i>tests</i>	
<a href="#">test_eofquotes()</a>	524
<i>tests</i>	
<a href="#">test_escape2()</a>	525
<a href="#">test()</a>	526
<i>test2 was lost, isn't any more</i>	
<a href="#">testContantBlah</a>	529
<i>Short Desc Test. This is the Extended Comment</i>	
<a href="#">test2()</a>	526
<a href="#">test_authoremail()</a>	522
<i>test function</i>	
<a href="#">test_558031()</a>	503
<a href="#">test_445305()</a>	494
<a href="#">test_445298()</a>	493
<i>test function</i>	
<a href="#">test_445820()</a>	495
<i>test function</i>	
<a href="#">testie</a>	497
<i>testie</i>	
<a href="#">thisisdumbbutworks</a>	498
<i>!</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::tearDown()</a>	572
<i>Tears down the fixture, for example, close a network connection.</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenLocationNotSetAndPearizeFalse()</a>	573
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeFalse()</a>	585
<i>Include a ".." in an absolute, PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetAndPearizeTrue()</a>	585
<i>Show correct behavior when</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeNull()</a>	585
<i>Include a ".." in an absolute, PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeTrue()</a>	585
<i>Include a ".." in an absolute, PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeNull()</a>	586
<i>Include a ".." in a relative, PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeFalse()</a>	586
<i>Include a ".." in a relative, PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetAndPearizeNull()</a>	584
<i>Show correct behavior when</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearLocationSetAndPearizeFalse()</a>	584
<i>Show correct behavior when</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeNull()</a>	582
<i>Include a ".." in an absolute, non-PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeFalse()</a>	582

<a href="#">PearizeFalse()</a>	582
<i>Include a ".." in an absolute, non-PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeTrue()</a>	583
<i>Include a ".." in an absolute, non-PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeFalse()</a>	583
<i>Include a ".." in a relative, non-PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeTrue()</a>	584
<i>Include a ".." in a relative, non-PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeNull()</a>	583
<i>Include a ".." in a relative, non-PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeTrue()</a>	586
<i>Include a ".." in a relative, PEAR path,</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests</a>	587
<i>Unit Testing of the phpDocumentor_setup's cleanConverterNamePiece() method</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleAndInvalidTertiaryA()</a>	590
<i>Extreme example of messy input...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleAndInvalidSecondary()</a>	589
<i>Extreme example of messy input...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleAndInvalidTertiaryB()</a>	590
<i>Extreme example of messy input...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleButValidPrimaryWithOneArg()</a>	590
<i>Extreme example of messy input...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleButValidSecondary()</a>	591
<i>Extreme example of messy input...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testExtremeExampleButValidPrimaryWithTwoArgs()</a>	591
<i>Extreme example of messy input...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnTertiary()</a>	589
<i>Verify no up-to-parent pathing is allowed...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnSecondary()</a>	589
<i>Verify no up-to-parent pathing is allowed...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::setUp()</a>	587
<i>Sets up the fixture, for example, open a network connection.</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::main()</a>	587
<i>Runs the test methods of this class.</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::tearDown()</a>	588
<i>Tears down the fixture, for example, close a network connection.</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingOnPrimaryWithOneArg()</a>	588
<i>Verify no up-to-parent pathing is allowed...</i>	
<a href="#">tests_phpDocumentorSetupCleanConverterNamePieceTests::testDoNotAllowTruePathingO</a>	

<a href="#">nPrimaryWithTwoArgs()</a>	588
<i>Verify no up-to-parent pathing is allowed...</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeTrue()</a>	582
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeNull()</a>	581
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeFalse()</a>	575
<i>Include a ".." in a relative, non-PEAR path,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeTrue()</a>	575
<i>Include a ".." in an absolute, non-PEAR path,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeNull()</a>	576
<i>Include a ".." in a relative, non-PEAR path,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearRelativeLocationSetAndPearizeTrue()</a>	576
<i>Include a ".." in a relative, non-PEAR path,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearLocationSetAndPearizeNull()</a>	577
<i>Show correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearLocationSetAndPearizeFalse()</a>	576
<i>Show correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeNull()</a>	575
<i>Include a ".." in an absolute, non-PEAR path,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetIncludingDotsAndPearizeFalse()</a>	575
<i>Include a ".." in an absolute, non-PEAR path,</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenLocationNotSetAndPearizeTrue()</a>	573
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenLocationNotSetAndPearizeNull()</a>	573
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeFalse()</a>	574
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeNull()</a>	574
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeTrue()</a>	574
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearLocationSetAndPearizeTrue()</a>	577
<i>Show correct behavior when</i>	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeFalse()</a>	577
<i>Include a ".." in an absolute, PEAR path,</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::tearDown()</a>	580
<i>Tears down the fixture, for example, close a network connection.</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::setUp()</a>	580
<i>Sets up the fixture, for example, open a network connection.</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenLocationNotSetAndPearizeFalse()</a>	580
<i>Shows correct behavior when</i>	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenLocationNotSetAndPearizeNull()</a>	580
<i>Shows correct behavior when</i>	

<a href="#">tests_ParserPageGetSourceLocationTests::testWhenNonPearLocationSetAndPearizeFalse()</a>	581
Shows correct behavior when	
<a href="#">tests_ParserPageGetSourceLocationTests::testWhenLocationNotSetAndPearizeTrue()</a>	581
Shows correct behavior when	
<a href="#">tests_ParserPageGetSourceLocationTests::main()</a>	579
Runs the test methods of this class.	
<a href="#">tests_ParserPageGetSourceLocationTests</a>	579
Unit Testing of the ParserPage's getSourceLocation() method	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeTrue()</a>	578
Include a ".." in an absolute, PEAR path,	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearLocationSetIncludingDotsAndPearizeNull()</a>	578
Include a ".." in an absolute, PEAR path,	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeFalse()</a>	578
Include a ".." in a relative, PEAR path,	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeNull()</a>	578
Include a ".." in a relative, PEAR path,	
<a href="#">tests_ParserClassGetSourceLocationTests::testWhenPearRelativeLocationSetAndPearizeTrue()</a>	579
Include a ".." in a relative, PEAR path,	
<a href="#">test2_441275()</a>	486
This tests some more advanced cases to make sure to handle them	

## U

<a href="#">utilities.php</a>	625
phpDocumentor :: docBuilder Web Interface	
<a href="#">Using the PDFParser XML templating language</a>	131

## V

<a href="#">varLink::\$type</a>	796
<a href="#">varLink</a>	795
class variable link	
<a href="#">vdump_par()</a>	626

## W

<a href="#">WordParser::getSource()</a>	929
Retrieve source code for the last function/method	
<a href="#">WordParser::getPos()</a>	929
Returns the current pointer position, or 1 character after the end of the word	
<a href="#">WordParser::getBlock()</a>	929
Unused	
<a href="#">WordParser::getWord()</a>	930
Retrieve a token from the token list	

<a href="#">WordParser::setPos()</a>	. . . . .	930
<i>Set the internal cursor within the source code</i>		
<a href="#">WordParser::setWhitespace()</a>	. . . . .	931
<i>set parser to return or strip whitespace</i>		
<a href="#">WordParser::setup()</a>	. . . . .	931
<i>Initialize the WordParser</i>		
<a href="#">WordParser::setSeparator()</a>	. . . . .	930
<i>Sets the list of possible separator tokens</i>		
<a href="#">WordParser::backupPos()</a>	. . . . .	929
<i>Backup to the previous token so that it can be retrieved again in a new context.</i>		
<a href="#">WordParser</a>	. . . . .	928
<i>Retrieves tokens from source code for use by the Parser</i>		
<a href="#">Writing templates for the CHMdefault Converter</a>	. . . . .	134
<a href="#">Writing templates for the PDFdefault Converter</a>	. . . . .	133
<a href="#">Writing a Converter, Methods</a>	. . . . .	122
<a href="#">Writing templates for the XMLDocBook Converter</a>	. . . . .	135
<a href="#">Writing templates for the HTMLSmarty Converter</a>	. . . . .	136
<a href="#">WordParser.inc</a>	. . . . .	924
<i>a generic lexer</i>		
<a href="#">Writing templates for the HTMLframes Converter</a>	. . . . .	137
<a href="#">Writing a New Converter</a>	. . . . .	113

## X

<a href="#">XMLDocBookpeardoc2Converter::convertVar()</a>	. . . . .	307
<i>Does nothing in this converter</i>		
<a href="#">XMLDocBookpeardoc2Converter::endClass()</a>	. . . . .	307
<i>Writes out the template file of <u>\$class_data</u> and unsets the template to save memory</i>		
<a href="#">XMLDocBookpeardoc2Converter::exampleProgramExample()</a>	. . . . .	307
<i>Used to convert the {@example} inline tag in a docblock.</i>		
<a href="#">XMLDocBookpeardoc2Converter::flushPackageXml()</a>	. . . . .	308
<a href="#">XMLDocBookpeardoc2Converter::convertTutorial()</a>	. . . . .	306
<i>Convert tutorials for output</i>		
<a href="#">XMLDocBookpeardoc2Converter::convertPage()</a>	. . . . .	306
<i>converts procedural pages for template output</i>		
<a href="#">XMLDocBookpeardoc2Converter::convertInclude()</a>	. . . . .	305
<i>Converts include elements for template output</i>		
<a href="#">XMLDocBookpeardoc2Converter::convertMethod()</a>	. . . . .	305
<i>Converts method for template output</i>		
<a href="#">XMLDocBookpeardoc2Converter::convertPackageName()</a>	. . . . .	306
<i>Does nothing - use tutorials for DocBook</i>		
<a href="#">XMLDocBookpeardoc2Converter::formatIndex()</a>	. . . . .	308
<i>Does nothing</i>		
<a href="#">XMLDocBookpeardoc2Converter::formatLeftIndex()</a>	. . . . .	308
<i>Creates package/lang/categoryname/packagename.xml for each package</i>		
<a href="#">XMLDocBookpeardoc2Converter::generateFormattedInterfaceTrees()</a>	. . . . .	309
<i>returns a template-enabled array of interface inheritance trees</i>		
<a href="#">XMLDocBookpeardoc2Converter::generatePkgElementIndex()</a>	. . . . .	310
<i>Generate alphabetical index of all elements by package and subpackage</i>		
<a href="#">XMLDocBookpeardoc2Converter::generatePkgElementIndexes()</a>	. . . . .	310
<a href="#">XMLDocBookpeardoc2Converter::getCDATA()</a>	. . . . .	310

<a href="#">XMLDocBookpeardoc2Converter::generateFormattedClassTrees()</a>	309
<i>returns a template-enabled array of class trees</i>	
<a href="#">XMLDocBookpeardoc2Converter::generateFormattedClassTree()</a>	309
<i>returns an array containing the class inheritance tree from the root object to the class</i>	
<a href="#">XMLDocBookpeardoc2Converter::formatPkgIndex()</a>	308
<i>Does nothing</i>	
<a href="#">XMLDocBookpeardoc2Converter::generateChildClassList()</a>	308
<i>returns a list of child classes</i>	
<a href="#">XMLDocBookpeardoc2Converter::generateElementIndex()</a>	308
<i>does nothing</i>	
<a href="#">XMLDocBookpeardoc2Converter::convertGlobal()</a>	305
<i>Converts global variables for template output</i>	
<a href="#">XMLDocBookpeardoc2Converter::convertFunction()</a>	305
<i>Converts function for template output - does nothing in peardoc2!</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$path</a>	301
<i>path of current page being converted</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$processSpecialRoots</a>	302
<i>This converter knows about the new root tree processing</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$sort_absolutely_everything</a>	302
<i>Pass elements by package, simplifies generation of package.xml/category.xml</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$sort_page_contents_by_type</a>	302
<i>XMLDocBookConverter wants elements sorted by type as well as alphabetically</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$page_dir</a>	301
<i>output directory for the current procedural page being processed</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$page_data</a>	301
<i>template for the procedural page currently being processed</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$packagexml</a>	301
<i>Contents of the packagename.xml file are stored in this template variable</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$package_pages</a>	301
<i>array of converted package page names.</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$page</a>	301
<i>name of current page being converted</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$sourceloc</a>	302
<a href="#">XMLDocBookpeardoc2Converter::\$template_options</a>	302
<i>template options. Currently only 1 recognized option usepear</i>	
<a href="#">XMLDocBookpeardoc2Converter::Br()</a>	304
<a href="#">XMLDocBookpeardoc2Converter::convertClass()</a>	304
<i>Converts class for template output</i>	
<a href="#">XMLDocBookpeardoc2Converter::convertDefine()</a>	304
<i>Converts defines for template output</i>	
<a href="#">XMLDocBookpeardoc2Converter::ConvertErrorLog()</a>	304
<i>Create errors.html template file output</i>	
<a href="#">XMLDocBookpeardoc2Converter::addSummaryToPackageXml()</a>	303
<a href="#">XMLDocBookpeardoc2Converter::\$write_globals_xml</a>	303
<a href="#">XMLDocBookpeardoc2Converter::\$peardoc2_constants</a>	303
<i>Constants, used for constants.tpl</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$peardoc2_globals</a>	303
<i>Global Variables, used for globals.tpl</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$write_constants_xml</a>	303
<a href="#">XMLDocBookpeardoc2Converter::getClassLink()</a>	310
<a href="#">XMLDocBookpeardoc2Converter::getDefineLink()</a>	311
<a href="#">XMLDocBookpeardoc2Converter::writeFile()</a>	317
<i>Writes a file to target dir, beautify any .xml files first</i>	

<a href="#">XMLDocBookpeardoc2Converter::writeNewPPage()</a>	318
<i>Does nothing</i>	
<a href="#">XMLDocBookpeardoc2Converter::writeSource()</a>	318
<i>Does nothing</i>	
<a href="#">XMLpackagePageParser.inc</a>	885
<i>Parser for XML DocBook-based phpDocumentor tutorials</i>	
<a href="#">XMLDocBookpeardoc2Converter::writeExample()</a>	317
<a href="#">XMLDocBookpeardoc2Converter::wordwrap()</a>	317
<a href="#">XMLDocBookpeardoc2Converter::setTemplateDir()</a>	316
<a href="#">XMLDocBookpeardoc2Converter::type_adjust()</a>	317
<a href="#">XMLDocBookpeardoc2Converter::unmangle()</a>	317
<a href="#">XMLPackagePageParser</a>	910
<i>Used to parse XML DocBook-based tutorials</i>	
<a href="#">XMLPackagePageParser::\$context</a>	911
<a href="#">XMLPackagePageParser::getParserEventName()</a>	912
<i>debugging function</i>	
<a href="#">XMLPackagePageParser::parse()</a>	913
<i>Parse a new file</i>	
<a href="#">XMLPackagePageParser::setupStates()</a>	913
<i>setup the parser tokens, and the pushEvent/popEvent arrays</i>	
<a href="#">XMLDocBookpeardoc2Converter.inc</a>	1140
<i>Source code</i>	
<a href="#">XMLPackagePageParser::\$refsect3id</a>	911
<a href="#">XMLPackagePageParser::\$refsect2id</a>	911
<a href="#">XMLPackagePageParser::\$eventHandlers</a>	911
<a href="#">XMLPackagePageParser::\$pars</a>	911
<a href="#">XMLPackagePageParser::\$refsect1id</a>	911
<a href="#">XMLDocBookpeardoc2Converter::returnSee()</a>	316
<i>This function takes an <a href="#">abstractLink</a> descendant and returns an html link</i>	
<a href="#">XMLDocBookpeardoc2Converter::returnLink()</a>	316
<a href="#">XMLDocBookpeardoc2Converter::getMethodLink()</a>	313
<a href="#">XMLDocBookpeardoc2Converter::getPageLink()</a>	313
<a href="#">XMLDocBookpeardoc2Converter::getPageName()</a>	313
<a href="#">XMLDocBookpeardoc2Converter::getRootTree()</a>	313
<i>return formatted class tree for the Class Trees page</i>	
<a href="#">XMLDocBookpeardoc2Converter::getLink()</a>	312
<i>do that stuff in \$template_options</i>	
<a href="#">XMLDocBookpeardoc2Converter::getId()</a>	312
<i>Get the id value needed to allow linking</i>	
<a href="#">XMLDocBookpeardoc2Converter::getExampleLink()</a>	311
<a href="#">XMLDocBookpeardoc2Converter::getFunctionLink()</a>	311
<a href="#">XMLDocBookpeardoc2Converter::getGlobalLink()</a>	312
<a href="#">XMLDocBookpeardoc2Converter::getSourceAnchor()</a>	314
<i>Retrieve a Converter-specific anchor to a segment of a source code file parsed via a <a href="#">@filesource</a> tag.</i>	
<a href="#">XMLDocBookpeardoc2Converter::getTutorialId()</a>	314
<a href="#">XMLDocBookpeardoc2Converter::ProgramExample()</a>	315
<a href="#">XMLDocBookpeardoc2Converter::rcNatCmp()</a>	316
<i>does a nat case sort on the specified second level value of the array</i>	
<a href="#">XMLDocBookpeardoc2Converter::rcNatCmp1()</a>	316
<i>does a nat case sort on the specified second level value of the array.</i>	
<a href="#">XMLDocBookpeardoc2Converter::prepareDocBlock()</a>	315
<a href="#">XMLDocBookpeardoc2Converter::postProcess()</a>	315

<a href="#">XMLDocBookpear</a>	<a href="#">doc2Converter::getVarLink()</a>	314
<a href="#">XMLDocBookpear</a>	<a href="#">doc2Converter::makeLeft()</a>	315
<a href="#">XMLDocBookpear</a>	<a href="#">doc2Converter::Output()</a>	315
	<i>Generate the constants.xml, packagename.xml, and globals.xml files</i>	
<a href="#">XMLDocBookpear</a>	<a href="#">doc2Converter::\$outputformat</a>	301
<a href="#">XMLDocBookpear</a>	<a href="#">doc2Converter::\$name</a>	301
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertFunction()</a>	286
	<i>Converts function for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertGlobal()</a>	287
	<i>Converts global variables for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertInclude()</a>	287
	<i>Converts include elements for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertMethod()</a>	287
	<i>Converts method for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::ConvertErrorLog()</a>	286
	<i>Create errors.html template file output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertDefine()</a>	286
	<i>Converts defines for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::\$sourceloc</a>	284
<a href="#">XMLDocBookConverter</a>	<a href="#">::\$template_options</a>	284
	<i>template options. Currently only 1 recognized option usepear</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertClass()</a>	285
	<i>Converts class for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertPackagePage()</a>	288
	<i>Converts package page and sets its package as used in \$package_pages</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertPage()</a>	288
	<i>converts procedural pages for template output</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::formatLeftIndex()</a>	289
	<i>Generate indexes for li_package.html and classtree output files</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::formatPkgIndex()</a>	290
	<i>HTMLdefaultConverter chooses to format both package indexes and the complete index here</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::generateElementIndex()</a>	290
	<i>Generate alphabetical index of all elements</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::generateFormattedClassTree()</a>	290
	<i>returns an array containing the class inheritance tree from the root object to the class</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::formatIndex()</a>	289
	<i>HTMLdefaultConverter uses this function to format template index.html and packages.html</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::endPage()</a>	289
	<i>Writes out the template file of \$page_data and unsets the template to save memory</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertTutorial()</a>	288
<a href="#">XMLDocBookConverter</a>	<a href="#">::convertVar()</a>	288
	<i>Converts class variables for template output.</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::endClass()</a>	289
	<i>Writes out the template file of \$class_data and unsets the template to save memory</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::\$sort_page_contents_by_type</a>	284
	<i>XMLDocBookConverter wants elements sorted by type as well as alphabetically</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::\$path</a>	284
	<i>path of current page being converted</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::\$class_data</a>	282
	<i>template for the class currently being processed</i>	
<a href="#">XMLDocBookConverter</a>	<a href="#">::\$class_dir</a>	282
	<i>output directory for the current class being processed</i>	

<a href="#">XMLDocBookConverter::\$current</a>	282
<i>contains all of the template procedural page element loop data needed for the current template</i>	
<a href="#">XMLDocBookConverter::\$currentclass</a>	282
<i>contains all of the template class element loop data needed for the current template</i>	
<a href="#">XMLDocBookConverter::\$class</a>	282
<i>name of current class being converted</i>	
<a href="#">XMLDocBookConverter::\$category</a>	282
<i>peardoc2 Category</i>	
<a href="#">XMLDocBookConverter.inc</a>	281
<i>XML output converter for DocBook.</i>	
<a href="#">XMLDocBookConverter</a>	281
<i>XML DocBook converter.</i>	
<a href="#">XMLDocBookConverter::\$base_dir</a>	281
<i>target directory passed on the command-line.</i>	
<a href="#">XMLDocBookConverter::\$function_data</a>	282
<a href="#">XMLDocBookConverter::\$juststarted</a>	283
<i>controls formatting of parser informative output</i>	
<a href="#">XMLDocBookConverter::\$package_pages</a>	283
<i>array of converted package page names.</i>	
<a href="#">XMLDocBookConverter::\$page</a>	284
<i>name of current page being converted</i>	
<a href="#">XMLDocBookConverter::\$page_data</a>	284
<i>template for the procedural page currently being processed</i>	
<a href="#">XMLDocBookConverter::\$page_dir</a>	284
<i>output directory for the current procedural page being processed</i>	
<a href="#">XMLDocBookConverter::\$outputformat</a>	283
<a href="#">XMLDocBookConverter::\$name</a>	283
<a href="#">XMLDocBookConverter::\$leftindex</a>	283
<i>indexes of elements by package that need to be generated</i>	
<a href="#">XMLDocBookConverter::\$local</a>	283
<i>whether a @see is going to be in the \$base_dir, or in a package/subpackage subdirectory of \$base_dir</i>	
<a href="#">XMLDocBookConverter::\$method_data</a>	283
<a href="#">XMLDocBookConverter::generateFormattedClassTrees()</a>	291
<i>returns a template-enabled array of class trees</i>	
<a href="#">XMLDocBookConverter::generatePkgElementIndex()</a>	291
<i>Generate alphabetical index of all elements by package and subpackage</i>	
<a href="#">XMLDocBookConverter::unmangle()</a>	298
<a href="#">XMLDocBookConverter::writeNewPPage()</a>	298
<a href="#">XMLDocBookpeardoc2Converter</a>	298
<i>XML DocBook converter.</i>	
<a href="#">XMLDocBookpeardoc2Converter::\$base_dir</a>	299
<i>target directory passed on the command-line.</i>	
<a href="#">XMLDocBookConverter::type_adjust()</a>	298
<a href="#">XMLDocBookConverter::SmartyInit()</a>	298
<a href="#">XMLDocBookConverter::returnSee()</a>	297
<i>This function takes an <a href="#">abstractLink</a> descendant and returns an html link</i>	
<a href="#">XMLDocBookConverter::setTargetDir()</a>	297
<i>calls the converter setTargetDir, and then copies any template images and the stylesheet if they haven't been copied</i>	
<a href="#">XMLDocBookConverter::setTemplateDir()</a>	297
<a href="#">XMLDocBookpeardoc2Converter::\$category</a>	299

<i>pear</i> <i>doc2 Category</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$class</i>	299
<i>name of current class being converted</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$juststarted</i>	300
<i>controls formatting of parser informative output</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$leftindex</i>	300
<i>indexes of elements by package that need to be generated</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$local</i>	300
<i>whether a @see is going to be in the \$base_dir, or in a package/subpackage subdirectory of \$base_dir</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$method_data</i>	300
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$function_data</i>	300
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$currentclass</i>	300
<i>contains all of the template class element loop data needed for the current template</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$class_data</i>	299
<i>template for the class currently being processed</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$class_dir</i>	299
<i>output directory for the current class being processed</i>	
<a href="#">XMLDocBookpear</a> <i>doc2Converter::\$current</i>	300
<i>contains all of the template procedural page element loop data needed for the current template</i>	
<a href="#">XMLDocBookConverter::returnLink()</a>	297
<a href="#">XMLDocBookConverter::rcNatCmp1()</a>	296
<i>does a nat case sort on the specified second level value of the array.</i>	
<a href="#">XMLDocBookConverter::getGlobalLink()</a>	293
<a href="#">XMLDocBookConverter::getId()</a>	293
<i>Get the id value needed to allow linking</i>	
<a href="#">XMLDocBookConverter::getLink()</a>	294
<i>do that stuff in \$template_options</i>	
<a href="#">XMLDocBookConverter::getMethodLink()</a>	294
<a href="#">XMLDocBookConverter::getFunctionLink()</a>	293
<a href="#">XMLDocBookConverter::getDefineLink()</a>	292
<a href="#">XMLDocBookConverter::generatePkgElementIndexes()</a>	292
<a href="#">XMLDocBookConverter::getCData()</a>	292
<a href="#">XMLDocBookConverter::getClassLink()</a>	292
<a href="#">XMLDocBookConverter::getPageLink()</a>	294
<a href="#">XMLDocBookConverter::getPageName()</a>	295
<a href="#">XMLDocBookConverter::postProcess()</a>	296
<a href="#">XMLDocBookConverter::prepareDocBlock()</a>	296
<a href="#">XMLDocBookConverter::rcNatCmp()</a>	296
<i>does a nat case sort on the specified second level value of the array</i>	
<a href="#">XMLDocBookConverter::Output()</a>	296
<i>This function is not used by HTMLdefaultConverter, but is required by Converter</i>	
<a href="#">XMLDocBookConverter::makeLeft()</a>	296
<a href="#">XMLDocBookConverter::getRootTree()</a>	295
<i>return formatted class tree for the Class Trees page</i>	
<a href="#">XMLDocBookConverter::getTutorialId()</a>	295
<a href="#">XMLDocBookConverter::getVarLink()</a>	295
<a href="#">XMLDocBookpear</a> <i>doc2Converter.inc</i>	280
<i>Outputs documentation in XML DocBook format, in the version expected by pear.php.net's documentation team</i>	

