# White Boy's Framework Project

# Contents

# Package default Procedural Elements

## app_config.php

- **Package** default
- **Filesource** Source Code for this file

# Package default Classes

## Class AppConfig

*[line 9]*

**AppConfig is a static class that managees configuration variables**

- **Package** default
- **Author** TQ White II

*void* function AppConfig::getServerVar($name) *[line 32]*

***Function Parameters:***

- **$name**

- **Static**
- **Access** public

*void* function AppConfig::initialize() *[line 11]*

- **Static**
- **Access** public

# Package phpDocumentor Procedural Elements

## host_ftp_config.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# init_environment.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

*void* function dispatchFirst($pathString) *[line 1]*
***Function Parameters:***

- **$pathString**

*void* function getTools() *[line 1]*
require_once **"app_config.php"** *[line 1]*

require_once **"host_ftp_config.php"** *[line 1]*

# ftp.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# transfer_directory.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# camelCaseToUnderscore.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

*void* function camelCaseToUnderscore($inString) *[line 4]*

***Function Parameters:***

- **$inString**

# contains.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

*void* function contains($pattern, $inString, [$noCase = 'true']) *[line 2]*

***Function Parameters:***

- **$pattern**
- **$inString**
- **$noCase**

# dump.php

**Write out human readable and PHP includable data. Keep the data types the same, so null and false and 0 and '' are all different.**

Something like an improved var_dump() or a neater var_export().

Unlike the built-in functions, this does write out null, 0, '', and false appropriately. Think of it as a data pretty-printer or a data formatter for writing include files. It even detects and handles circular references.

Of course, it would be nice if it could dump the recursive variables so that running the code would set the pointers up, but until I think of an extra-clever way ....

function dump($data, $name = false, $html = true, $echo_out = true) $data = What you want to dump $name = The name to display, great for naming the dumped data $html = Escape for HTML, turn off to make an include file $echo_out = Write to output, turn off if you want the content returned

Also includes dump_callstack() to show where you are.

- **Package** phpDocumentor
- **Filesource** Source Code for this file

*void* function dump($data, [$name = false], [$html = null], [$echo_out = true]) *[line 65]*

***Function Parameters:***

- **$data**
- **$name**
- **$html**
- **$echo_out**

**Dumps data - Calls the __dump_anything() function for the real work**

*void* function dump_callstack([$return = false]) *[line 718]*

***Function Parameters:***

- **$return**

**Dump the call stack to the current point.**

$return = false (wrap in HTML and echo), or true (return text)

*void* function __dump_anything(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 139]*

***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Determines how to write the data - calls __dump_* helper functions**

*void* function __dump_array(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 154]*

***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out an array, avoids recursion.**

*void* function __dump_boolean(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 402]*

***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out a boolean value**

*void* function __dump_collapse_end($html, $noFormat) *[line 240]*
***Function Parameters:***

- **$html**
- **$noFormat**

### Closes a toggle

*void* function __dump_collapse_start($html, $noFormat, $count) *[line 209]*
***Function Parameters:***

- **$html**
- **$noFormat**
- **$count**

### Starts a toggle so the user can show/hide large amounts of data

*void* function __dump_default(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 515]*
***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

### This is the default catch-all that should never get used.

*void* function __dump_double(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 474]*
***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out a number.  No special formatting needed.**

*void* function __dump_index(&$data, $html) *[line 254]*
***Function Parameters:***

- **&$data**
- **$html**

**Write out an index to an array or object.**
Only called from __dump_array, __dump_object, and __dump_recursive.  Only handles known data types (try to keep it to strings and integers).

*void* function __dump_integer(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 467]*
***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out an integer by calling __dump_double**

*void* function __dump_NULL(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 485]*
***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out a null value**

*void* function __dump_object(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 263]*
***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out an object, avoids recursion.**

*void* function __dump_recursive(&$uniqStack, $html, $noFormat) *[line 355]*

***Function Parameters:***

- **&$uniqStack**
- **$html**
- **$noFormat**

**Writes out a recursion message.**

Called by __dump_array and __dump_object when recursion is detected.

*void* function __dump_resource(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 494]*

***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out a resource**

If it is a stream resource, additional attributes are also shown.

*void* function __dump_sql($data) *[line 540]*

***Function Parameters:***

- **$data**

**Write out JS to show the select statement in a new window.**

Do not pass $data by reference for this function.

*void* function __dump_sql_html(&$sql) *[line 669]*

***Function Parameters:***

- **&$sql**

*void* function __dump_sql_test(&$data) *[line 522]*

***Function Parameters:***

- **&$data**

**Determine if we should allow the display of this statement as SQL**

*void* function __dump_sql_token($token, $indent) *[line 609]*

***Function Parameters:***

- **$token**
- **$indent**

*void* function __dump_string(&$data, $html, $indentation, $uniqId, $uniqStack, $noFormat) *[line 424]*

***Function Parameters:***

- **&$data**
- **$html**
- **$indentation**
- **$uniqId**
- **$uniqStack**
- **$noFormat**

**Writes out an escaped string.**

To avoid additional overhead if the string is parsed, single quotes are  preferred.

# processTemplateArray.php

- **Package** phpDocumentor
- **Filesource** Source Code for this file

*void* function processSubstitutions($template, $inputArray, [$ctlArray = ''], [$parmArray = ''], [$evalArrays = ''], [$dataArrays = '']) *[line 5]*

***Function Parameters:***

- **$template**
- **$inputArray**
- **$ctlArray**
- **$parmArray**
- **$evalArrays**
- **$dataArrays**

*void* function processTemplateArray($template, $inputArray, [$inCtlArray = ''], [$inParmArray = ''], [$inEvalArrays = ''], [$inDataArrays = '']) *[line 338]*

***Function Parameters:***

- **$template**
- **$inputArray**
- **$inCtlArray**
- **$inParmArray**
- **$inEvalArrays**
- **$inDataArrays**

*void* function processText($inString, [$autoParagraphs = 'on'], [$processOutline = 'off'], [$activeURLs = 'off'], [$keepLineBreaksFlag = 'false'], [$skipURLdecode = 'true']) *[line 106]*

***Function Parameters:***

- **$inString**
- **$autoParagraphs**
- **$processOutline**
- **$activeURLs**
- **$keepLineBreaksFlag**
- **$skipURLdecode**

*void* function replaceArray($inString, $inArray) *[line 90]*

***Function Parameters:***

- **$inString**
- **$inArray**

*void* function replaceCompoundTags($inString, $dataArray2, [$parmArray = "noDBtbl"], [$debug = ""]) *[line 211]*

***Function Parameters:***

- **$inString**
- **$dataArray2**
- **$parmArray**
- **$debug**

# index.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# page2.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# environment_info.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# index.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# install_notes.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# send_files_to_host.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# index.ctp

- **Package** phpDocumentor
- **Filesource** Source Code for this file

# Package phpDocumentor Classes

## Class Ftp

*[line 13]*

**FTP - access to an FTP server.**

- **Package** phpDocumentor
- **Author** David Grudl
- **Version** 1.0
- **Copyright** Copyright (c) 2008 David Grudl
- **Link** http://phpfashion.com/
- **License** New

**Ftp::ASCII**

= FTP_ASCII *[line 15]*

**Ftp::AUTORESUME**

= FTP_AUTORESUME *[line 21]*

**Ftp::AUTOSEEK**

= FTP_AUTOSEEK *[line 20]*

**Ftp::BINARY**

= FTP_BINARY *[line 17]*

**Ftp::FAILED**

= FTP_FAILED *[line 22]*

**Ftp::FINISHED**

= FTP_FINISHED *[line 23]*

**Ftp::IMAGE**

= FTP_IMAGE *[line 18]*

**Ftp::MOREDATA**

= FTP_MOREDATA *[line 24]*

**Ftp::TEXT**

= FTP_TEXT *[line 16]*

**Ftp::TIMEOUT_SEC**

= FTP_TIMEOUT_SEC *[line 19]*

*bool* function Ftp::fileExists($file) *[line 134]*

***Function Parameters:***

- *string* **$file**

## Checks if file or directory exists.

- **Access** public

*bool* function Ftp::isDir($dir) *[line 146]*

***Function Parameters:***

- *string* **$dir**

## Checks if directory exists.

- **Access** public

*void* function Ftp::mkDirRecursive($dir) *[line 164]*

***Function Parameters:***

- *string* **$dir**

**Recursive creates directory.**

- **Access** public

*void* function Ftp::reconnect() *[line 119]*

**Reconnects to FTP server.**

- **Access** public

*void* function Ftp::_errorHandler($code, $message) *[line 108]*

***Function Parameters:***

- **$code**
- **$message**

**Internal error handler. Do not call directly.**

- **Access** public

*void* function Ftp::__call($name, $args) *[line 51]*

***Function Parameters:***

- **$name**

- **$args**

**Magic method (do not call directly).**

- **Access** public

# Class FtpException

*[line 185]*

- **Package** phpDocumentor

# Class HostFtpConfig

*[line 3]*

- **Package** phpDocumentor

**HostFtpConfig::$baseUrl**

*mixed* = 'http://tqwhite.org/' *[line 8]*

- **Static**
- **Access** public

**HostFtpConfig::$docRootDir**

*mixed* = '/public_html/' *[line 9]*

- **Static**
- **Access** public

**HostFtpConfig::$ftpHost**

*mixed* = 'tqwhite.org' *[line 13]*

- **Static**
- **Access** public

**HostFtpConfig::$ftpPass**

*mixed* = 'tq3141' *[line 12]*

- **Static**
- **Access** public

**HostFtpConfig::$ftpUser**

*mixed* = 'tq.org' *[line 11]*

- **Static**
- **Access** public

**HostFtpConfig::$local**

*mixed* = ROOT *[line 5]*

- **Static**
- **Access** public

**HostFtpConfig::$remote**

*mixed* = '/' *[line 6]*

- **Static**
- **Access** public

*void* function HostFtpConfig::copyIntoTransferDirectoryObject($transferDirectory) *[line 15]*

***Function Parameters:***

- **$transferDirectory**

- **Access** public

*void* function HostFtpConfig::test($transferDirectory) *[line 35]*

***Function Parameters:***

- **$transferDirectory**

- **Access** public

# Class qLoad
*[line 1]*

- **Package** phpDocumentor

*void* function qLoad::qAutoload($className) *[line 1]*
***Function Parameters:***

- **$className**

# Class transferDirectory
*[line 2]*

- **Package** phpDocumentor

**transferDirectory::$allFilesArray**

*mixed* = array() *[line 25]*

- **Access** public

**transferDirectory::$arrayItemWrapperTemplate**

*mixed* = " <!message!> " *[line 39]*

- **Access** public

**transferDirectory::$deletedFilesArray**

*mixed* = array() *[line 24]*

- **Access** public

**transferDirectory::$deleteFileTemplate**

*mixed* = *[line 41]*

- **Access** public

**transferDirectory::$deleteReportArray**

*mixed* = array() *[line 33]*

- **Access** public

**transferDirectory::$dryRunFlag**

*mixed* = false *[line 8]*

- **Access** public

**transferDirectory::$errorReportArray**

*mixed* = array() *[line 34]*

- **Access** public

**transferDirectory::$excludedFilesArray**

*mixed* = array() *[line 26]*

- **Access** public

**transferDirectory::$exclusionStringArray**

*mixed* = array() *[line 19]*

- **Access** public

**transferDirectory::$ftpHost**

*mixed* = *[line 6]*

- **Access** public

**transferDirectory::$ftpPass**

*mixed* = *[line 5]*

- **Access** public

**transferDirectory::$ftpUser**

*mixed* = *[line 4]*

- **Access** public

**transferDirectory::$helpArray**

*mixed* = array() *[line 31]*

- **Access** public

**transferDirectory::$initDatabase**

*mixed* = false *[line 9]*

- **Access** public

**transferDirectory::$linkTemplate**

*mixed* = '(<A href=<!url!> target=_new>here</A>)' *[line 42]*

- **Access** public

**transferDirectory::$needsUploadArray**

*mixed* = array() *[line 22]*

- **Access** public

**transferDirectory::$newFileArray**

*mixed* = array() *[line 23]*

- **Access** public

**transferDirectory::$pathTranslationArray**

*mixed* = *[line 36]*

- **Access** public

**transferDirectory::$processStatusArray**

*mixed* = array() *[line 29]*

- **Access** public

**transferDirectory::$unchangedArray**

*mixed* = array() *[line 21]*

- **Access** public

**transferDirectory::$uploadedFileCount**

*mixed* = 0 *[line 27]*

- **Access** public

**transferDirectory::$uploadFileTemplate**

*mixed* = *[line 40]*

- **Access** public

**transferDirectory::$uploadReportArray**

*mixed* = array() *[line 32]*

- **Access** public

**transferDirectory::$urlTranslationArray**

*mixed* = *[line 37]*

- **Access** public

Constructor *void* function transferDirectory::__construct($baseDirPath) *[line 45]*

***Function Parameters:***

- **$baseDirPath**

- **Access** public

*void* function transferDirectory::addExclusionString($item) *[line 449]*

***Function Parameters:***

- **$item**

- **Access** public

*void* function transferDirectory::analyzeFiles() *[line 195]*

- **Access** public

*void* function transferDirectory::clearExclusionString() *[line 443]*

- **Access** public

*void* function transferDirectory::deleteFile($localPath) *[line 356]*

***Function Parameters:***

- **$localPath**

- **Access** public

*void* function transferDirectory::executeFTP() *[line 401]*

- **Access** public

*void* function transferDirectory::getControlFilePath() *[line 119]*

- **Access** public

*void* function transferDirectory::saveThisObject() *[line 147]*

- **Access** public

*void* function transferDirectory::sendFile($localPath) *[line 278]*
***Function Parameters:***

- **$localPath**

- **Access** public

*void* function transferDirectory::showArrayProperty($property) *[line 99]*
***Function Parameters:***

- **$property**

- **Access** public

*void* function transferDirectory::showFileArray($fileArray) *[line 426]*
***Function Parameters:***

- **$fileArray**

- **Access** public

*void* function transferDirectory::showHelp() *[line 123]*

- **Access** public

# Package WhiteBoysFramework Procedural Elements

## base_class.php

**Base Class for White Boys' Framework**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

# base_controller.php

**Base Controller**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

# open_source.php

**OpenSourceController contains actions for pages accessed by a URL of the form http://domain.com/openSource/action**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

# site_utilities.php

**SiteUtilities contains actions for pages accessed by a URL of the form http://domain.com/siteUtilities/action**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

# start.php

**SiteUtilities contains actions for pages accessed by a URL of the form http://domain.com/start/action or http://domain.com**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

# base_view.php

**Base mechanism for display of pages accessed by a URL of the form http://domain.com/openSource/action**

- **Package** WhiteBoysFramework
- **Filesource** Source Code for this file

# Package WhiteBoysFramework Classes

## Class BaseClass

*[line 10]*

**Base Class for White Boys' Framework**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

## Class BaseController

*[line 10]*

**Base Controller**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

**BaseController::$entryClassName**

*mixed = [line 12]*

- **Access** protected

**BaseController::$unknownActionHandlerName**

*mixed* = 'index' *[line 14]*

- **Access** protected

Constructor *void* function BaseController::__construct() *[line 16]*

- **Access** public

*void* function BaseController::getUnknownActionHandlerName() *[line 24]*

- **Access** public

*void* function BaseController::setUnknownActionHandlerName($value) *[line 20]*
***Function Parameters:***

- **$value**

- **Access** public

*void* function BaseController::_getControllerNameForUrl() *[line 43]*

- **Access** protected

*void* function BaseController::_getTemplatePath($templateName) *[line 28]*
***Function Parameters:***

- **$templateName**

- **Access** protected

*void* function BaseController::_getUrlPath([$templateName = '']) *[line 47]*
***Function Parameters:***

- **$templateName**

- **Access** protected

# Class BaseView

*[line 9]*

**Base mechanism for display of pages accessed by a URL of the form http://domain.com/openSource/action**

- **Package** WhiteBoysFramework
- **Filesource** Source Code for this file

Constructor *none* function BaseView::__construct([$templatePathName = '']) *[line 25]*

***Function Parameters:***

- *$templatePathName* **$templatePathName** (optional)

### Generates a view object and, optionally, sets the template

- **Author** TQ White II
- **Access** public

*void* function BaseView::render() *[line 58]*

- **Access** public

*none* function BaseView::SetViewScopedValue($varName, $value) *[line 37]*

***Function Parameters:***

- *$varName:* **$varName** name by which template will refer to this value
- *$value:* **$value** value of the template variable

### Set View-scoped Value makes a value available to a display template.

- **Author** TQ White II
- **Access** public

# Class OpenSource

*[line 11]*

**OpenSourceController contains actions for pages accessed by a URL of the form http://domain.com/openSource/action**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

Constructor *none* function OpenSource::__construct(0) *[line 22]*

***Function Parameters:***

- *none* **0**

**Calculates default page info, renders appropriate template -presently it's demo pages**

- **Author** TQ White II
- **Access** public

*void* function OpenSource::index([$pageNum = '']) *[line 28]*

***Function Parameters:***

- **$pageNum**

- **Access** public

# Class SiteUtilities

*[line 11]*

**SiteUtilities contains actions for pages accessed by a URL of the form http://domain.com/siteUtilities/action**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

Constructor *void* function SiteUtilities::__construct() *[line 14]*

- **Access** public

*none* function SiteUtilities::dispatchSpecial(0) *[line 31]*

***Function Parameters:***

- *none* **0**

**NOT FINISHED -a list of links**

- **Author** TQ White II
- **Access** public
- **Example** example not found

*none* function SiteUtilities::environmentInfo(0) *[line 172]*

***Function Parameters:***

- *none* **0**

**Displays a variety of system info for developers -phpInfo(), $_SERVER**

- **Author** TQ White II
- **Access** public

*none* function SiteUtilities::index(0) *[line 54]*

***Function Parameters:***

- *none* **0**

**Calculates default page info, renders appropriate template -a list of links**

- **Author** TQ White II
- **Access** public

*none* function SiteUtilities::sendFilesToHost(0) *[line 99]*

***Function Parameters:***

- *none* **0**

**Uploads changed development files to web host using FTP -based on configs/host_ftp_config.php**

- **Author** TQ White II
- **Access** public

# Class Start

*[line 12]*

**SiteUtilities contains actions for pages accessed by a URL of the form http://domain.com/start/action or http://domain.com**

- **Package** WhiteBoysFramework
- **Author** TQ White II < tq@justkidding.com>
- **Filesource** Source Code for this file

Constructor *none* function Start::__construct(0) *[line 23]*

***Function Parameters:***

- *none* **0**

**Initialize object -sets $this->entryClassName**

- **Author** TQ White II
- **Access** public

*void* function Start::fileDemo() *[line 62]*

- **Access** public

*none* function Start::index(0) *[line 38]*

***Function Parameters:***

- *none* **0**

**Calculates default page info, renders appropriate template  -a list of links**

- **Author** TQ White II
- **Access** public

# Appendices

# Appendix A - Class Trees

## Package default

### AppConfig

- AppConfig

## Package phpDocumentor

### Ftp

- Ftp

### FtpException

- Exception
  - FtpException

### HostFtpConfig

- HostFtpConfig

### qLoad

- qLoad

### transferDirectory

- transferDirectory

# Package WhiteBoysFramework

## BaseClass

- BaseClass

## BaseController

- \
  - BaseController

## BaseView

- \
  - BaseView

## OpenSource

- \
  - OpenSource

## SiteUtilities

- \
  - SiteUtilities

## Start

- \
  - Start

# Appendix C - Source Code

# Package WhiteBoysFramework

# File Source for app_config.php

*Documentation for this file is available at app_config.php*

```
1     <?php
2     namespace configs;
3     /**
4      * AppConfig is a static class that managees configuration variables
5      *
6      * @package default
7      * @author TQ White II
8      ***/
9     class AppConfig {
10
11        public static function initialize(){
12            define('DS', DIRECTORY_SEPARATOR);
13            define('ROOT', dirname(dirname(dirname(__FILE__))).'/');
14
15            define('FRAMEWORKROOT', ROOT.'framework/');
16            define('MVCROOT', ROOT.'framework/mvc/');
17            define('CONFIGROOT', ROOT.'framework/configs/');
18            define('LIBROOT', ROOT.'framework/library/');
19            define('HTMLROOT', ROOT.'public_html/');
20
21
22            //initialize system stuff
23
24            if (!isset($_SERVER['SYSTEM_TYPE'])){
25                define('SYSTEM_TYPE', 'production');
26            }
27            else{
28                define('SYSTEM_TYPE', $_SERVER['SYSTEM_TYPE']); //usually this is 'development' from
httpd.conf
29            }
30        }
31
32        public static function getServerVar($name){
33
34            if (isset($_SERVER[$name])){
35                return $_SERVER[$name];
36            }
37            else{
38                return '';
39            }
40        }
41
42    } //end of class
```

Package WhiteBoysFramework

# File Source for host_ftp_config.php

*Documentation for this file is available at host_ftp_config.php*

```
1     <?php
2
3     class HostFtpConfig {
4
5     public static $local=ROOT;
6     public static $remote='/';
7
8     public static $baseUrl='http://tqwhite.org/';
9     public static $docRootDir='/public_html/';
10
11    public static $ftpUser='tq.org';
12    public static $ftpPass='tq3141';
13    public static $ftpHost='tqwhite.org';
14
15    public function copyIntoTransferDirectoryObject($transferDirectory){
16
17        $itemArray=array();
18
19            $itemArray['local']=self::$local;
20            $itemArray['remote']=self::$remote;
21            $transferDirectory->   pathTranslationArray[]=$itemArray;
22
23        $itemArray=array();
24
25            $itemArray['baseUrl']=self::$baseUrl;
26            $itemArray['docRootDir']=self::$docRootDir;
27            $transferDirectory->   urlTranslationArray=$itemArray;
28
29        $transferDirectory->   ftpUser=self::$ftpUser;
30        $transferDirectory->   ftpPass=self::$ftpPass;
31        $transferDirectory->   ftpHost=self::$ftpHost;
32
33    }
34
35    public function test($transferDirectory){
36    $transferDirectory->   testVar='goodbye';
37    }
38
39    } //end of class
```

# File Source for init_environment.php

*Documentation for this file is available at init_environment.php*

```
1     <?phpnamespace
configs;$ds=DIRECTORY_SEPARATOR;require_once("app_config.php"          );AppConfig::initialize();
//define constants, etcfunction getTools(){    $ds=DIRECTORY_SEPARATOR;
$prefix=LIBROOT."     tools{$ds}"     ;    require_once("     {$prefix}dump.php"     );
require_once("     {$prefix}contains.php"     );
require_once("     {$prefix}processTemplateArray.php"     );
require_once("     {$prefix}camelCaseToUnderscore.php"     );}function dispatchFirst($pathString){
$ds=DS;        $pathArray=explode($ds, $pathString);        $controller=array_shift($pathArray);
$provisionalParameterArray=$pathArray;    $action=array_shift($pathArray);    $parameterArray=$pathArray;
//whatever is left is parameters    $className="     \\mvc\\controllers\\$controller"     ;
$controllerObj=new $className;        if (empty($action)){$action='index';}        if
(method_exists($controllerObj, $action)){
call_user_func_array(array($controllerObj,$action),$parameterArray);    }    else{        if
(method_exists($controllerObj, 'getUnknownActionHandlerName')){            $action=$controllerObj-
>   getUnknownActionHandlerName();
call_user_func_array(array($controllerObj,$action),$provisionalParameterArray);        }        else{
       exit("make it so that unknown action goes to the front page or something"          );        }
}}class qLoad{    function qAutoload($className){            try{            $ds=DIRECTORY_SEPARATOR;
              $fileName=str_replace('\\', '/', $className); //convert namespace to filepath
  $fileName=camelCaseToUnderscore($fileName);
$fileName=FRAMEWORKROOT."     $fileName.php"     ;                        if (!file_exists($fileName)){
            throw new \Exception("fileErr"          );            }
require_once($fileName);            return true;        }               catch (\Exception $e){
 if ($e->   getMessage()=='fileErr'){                $message="                          FATAL ERROR:
Class $className cannot be found at<BR>                                $fileName ("     .__FUNCTION__.")
                "     ;                exit($message);          }           else{
$message="                    FATAL ERROR: Class $className had an unknown problem at<BR>
             $fileName ("     .__FUNCTION__.")                          "          ;
exit($message);                    }      }            }}//end of class$qLoadObj=new
qLoad;spl_autoload_register(array($qLoadObj, 'qAutoload'));getTools();if
(SYSTEM_TYPE=='development'){require_once("host_ftp_config.php"          );}
```

# File Source for ftp.php

*Documentation for this file is available at ftp.php*

```php
<?php

/**
 * FTP - access to an FTP server.
 *
 * @author     David Grudl
 * @copyright  Copyright (c) 2008 David Grudl
 * @license    New BSD License
 * @link       http://phpfashion.com/
 * @version    1.0
 * @download    http://code.google.com/p/ftp-php/
 */
class Ftp
{
    const ASCII = FTP_ASCII;
    const TEXT = FTP_TEXT;
    const BINARY = FTP_BINARY;
    const IMAGE = FTP_IMAGE;
    const TIMEOUT_SEC = FTP_TIMEOUT_SEC;
    const AUTOSEEK = FTP_AUTOSEEK;
    const AUTORESUME = FTP_AUTORESUME;
    const FAILED = FTP_FAILED;
    const FINISHED = FTP_FINISHED;
    const MOREDATA = FTP_MOREDATA;

    private static $aliases = array(
        'sslconnect' =>    'ssl_connect',
        'getoption' =>    'get_option',
        'setoption' =>    'set_option',
        'nbcontinue' =>    'nb_continue',
        'nbfget' =>    'nb_fget',
        'nbfput' =>    'nb_fput',
        'nbget' =>    'nb_get',
        'nbput' =>    'nb_put',
    );

    /** @var resource */
    private $resource;

    /** @var array */
    private $state;

    /** @var string */
    private $errorMsg;



    /**
     * Magic method (do not call directly).
     */
    public function __call($name, $args)
    {
        $name = strtolower($name);
        $silent = strncmp($name, 'try', 3) === 0;
        $func = $silent ? substr($name, 3) : $name;
        $func = 'ftp_' . (isset(self::$aliases[$func]) ? self::$aliases[$func] : $func);

        if (!function_exists($func)) {
            throw new Exception("     Call to undefined method Ftp::$name()."     );
        }

        $this->   errorMsg = NULL;
        set_error_handler(array($this, '_errorHandler'));

        if ($func === 'ftp_connect' || $func === 'ftp_ssl_connect') {
            $this->   state = array($name =>    $args);
            $this->   resource = call_user_func_array($func, $args);
```

```
68                $res = NULL;
69
70            } elseif (!is_resource($this->   resource)) {
71                restore_error_handler();
72                throw new FtpException("Not connected to FTP server. Call connect() or
ssl_connect() first."     );
73
74            } else {
75                if ($func === 'ftp_login') {
76                    $this->   state['login'] = $args;
77                }
78
79                array_unshift($args, $this->   resource);
80                $res = call_user_func_array($func, $args);
81
82                if ($func === 'ftp_chdir' || $func === 'ftp_cdup') {
83                    $this->   state['chdir'] = array(ftp_pwd($this->   resource));
84                }
85            }
86
87            restore_error_handler();
88            if (!$silent &&          $this->   errorMsg !== NULL) {
89                if (ini_get('html_errors')) {
90                    $this->   errorMsg = html_entity_decode(strip_tags($this->   errorMsg));
91                }
92
93                if (($a = strpos($this->   errorMsg, ': ')) !== FALSE) {
94                    $this->   errorMsg = substr($this->   errorMsg, $a + 2);
95                }
96
97                throw new FtpException($this->   errorMsg);
98            }
99
100           return $res;
101       }
102
103
104
105       /**
106        * Internal error handler. Do not call directly.
107        */
108       public function _errorHandler($code, $message)
109       {
110           $this->   errorMsg = $message;
111       }
112
113
114
115       /**
116        * Reconnects to FTP server.
117        * @return void
118        */
119       public function reconnect()
120       {
121           @ftp_close($this->   resource);
122           foreach ($this->   state as $name =>    $args) {
123               call_user_func_array(array($this, $name), $args);
124           }
125       }
126
127
128
129       /**
130        * Checks if file or directory exists.
131        * @param  string
132        * @return bool
133        */
134       public function fileExists($file)
135       {
136           return is_array($this->   nlist($file));
137       }
138
139
140
141       /**
142        * Checks if directory exists.
143        * @param  string
144        * @return bool
145        */
146       public function isDir($dir)
```

```
147         {
148             $current = $this->   pwd();
149             try {
150                 $this->   chdir($dir);
151             } catch (FtpException $e) {
152             }
153             $this->   chdir($current);
154             return empty($e);
155         }
156 
157 
158 
159         /**
160          * Recursive creates directory.
161          * @param  string
162          * @return void
163          */
164         public function mkDirRecursive($dir)
165         {
166             $parts = explode('/', $dir);
167             $path = '';
168             while (!empty($parts)) {
169                 $path .= array_shift($parts);
170                 try {
171                     if ($path !== '') $this->   mkdir($path);
172                 } catch (FtpException $e) {
173                     if (!$this->   isDir($path)) {
174                         throw new FtpException("      Cannot create directory '$path'."      );
175                     }
176                 }
177                 $path .= '/';
178             }
179         }
180 
181     }
182 
183 
184 
185     class FtpException extends Exception
186     {
187     }
```

# File Source for transfer_directory.php

*Documentation for this file is available at transfer_directory.php*

```
1     <?
2     class transferDirectory{
3
4         public $ftpUser;
5         public $ftpPass;
6         public $ftpHost;
7
8         public $dryRunFlag=false;
9         public $initDatabase=false;
10
11        private $baseDirPath;
12        private $controlFileName;
13        private $controlFilePath;
14        private $referenceFileArray;
15        private $globArray;
16
17
18        private $mandatoryExcludes=array('nocopy','/system/' );
19        public $exclusionStringArray=array();
20
21        public $unchangedArray=array();
22        public $needsUploadArray=array();
23        public $newFileArray=array();
24        public $deletedFilesArray=array();
25        public $allFilesArray=array();
26        public $excludedFilesArray=array();
27        public $uploadedFileCount=0;
28
29        public    $processStatusArray=array();
30
31        public $helpArray=array();
32        public $uploadReportArray=array();
33        public $deleteReportArray=array();
34        public $errorReportArray=array();        //eg, justkidding.com
35
36        public $pathTranslationArray;
37        public $urlTranslationArray;
38
39        public $arrayItemWrapperTemplate="<div style=font-family:sans-serif;font-
size:8pt;color:green;><!message!></div>"                    ;
40        public $uploadFileTemplate;
41        public $deleteFileTemplate;
42        public $linkTemplate='(<A href=<!url!> target=_new>here</A>)'                ;
43
44
45    public function __construct($baseDirPath){
46
47
48            $this->   baseDirPath=$baseDirPath;
49            $this->   controlFileName='.ftpControl';
50            $this->   controlFilePath=$this->   baseDirPath.DIRECTORY_SEPARATOR.$this-
>   controlFileName;
51
52
53            $this->   exclusionStringArray=$this->   mandatoryExcludes;
54            $this->   addExclusionString($this->   controlFileName);
55
56
57
58    @        $tmp=$this->   getThisObject($this->   controlFilePath);
59
60            if (is_object($tmp)){
61                $tmpArray=(array) $tmp; //thanks Tyler
62                foreach ($tmpArray as $label=>   $data){
63                    $labelSplit = explode("\0"          , $label);
64                    if (is_array($labelSplit)){
65                    $labelName=array_pop($labelSplit);;
```

```
66                  $this->   $labelName=$data;
67                  }
68
69              unset($tmp);
70              }
71
72              $this->   processStatusArray=array();
73              $this->   processStatusArray[]="     found control file: {$this->   controlFilePath}";
74              $this->   processStatusArray[]="restored class from {$this->   controlFileName}";
75
76          }
77
78      $this->   uploadedFileCount=0;
79      $this->   helpArray=array();
80      $this->   uploadReportArray=array();
81      $this->   deleteReportArray=array();
82      $this->   errorReportArray=array();
83      $this->   excludedFilesArray=array();
84
85      $this->   pathTranslationArray=array();
86      $this->   urlTranslationArray=array();
87
88
89      //these are retained until FTP activity changes them
90           $this->   needsUploadArray=array();
91           $this->   newFileArray=array();
92
93      $this->   globArray=$this->   getAllFilesInDir($this->   baseDirPath);
94
95      //moved to client page: $this->analyzeFiles();
96  }
97
98
99  public function showArrayProperty($property){
100     $outString='';
101     if (is_array($property)){
102         foreach ($property as $label=>   $data){
103             if (!empty($this->   arrayItemWrapperTemplate)){
104                 $outString.=str_replace("<!message!>"               , $data, $this-
>   arrayItemWrapperTemplate);
105             }
106             else{
107
108                 $outString.=$data;
109             }
110         }
111     if (empty($outString)){
112
113                 $outString.=str_replace("<!message!>"               , '--none--', $this-
>   arrayItemWrapperTemplate);
114     }
115     }
116   return($outString);
117   }
118
119   public function getControlFilePath(){
120       return $this->   controlFilePath;
121       }
122
123   public function showHelp(){
124       $stringList='';
125       foreach ($this->   exclusionStringArray as $data){
126           $stringList.="$data<BR>      ";
127       }
128
129       $outString="
130
131       File analysis excludes and does not upload:
132       1) the transferDirectory control file ({$this->   controlFileName}<BR>
133       2) files that have any of the following strings anywhere in their file path (eg, ~/foo/XXX/bar,
or XXX.php<p/>
134       $stringList
135       ";
136
137       $this->   helpArray[]=$outString;
138
139   }
140
141   private function getThisObject($filePath){
142       $resultString = file_get_contents($filePath);
```

```
143         return(unserialize($resultString));
144     }
145
146
147     public function saveThisObject(){
148
149             if ($this->   dryRunFlag==false or $this->   initDatabase==true){
150
151                 if (!is_dir($this->   baseDirPath)){
152                     mkdir($this->   baseDirPath, 0755, true);
153                 }
154
155                 $filePath=$this->   controlFilePath;
156                 $contentString=serialize($this);
157                 $result=file_put_contents($filePath, $contentString);
158
159             }
160             else{
161                 $this->   processStatusArray[]="dry run flag set, control file ({$this-
>   controlFileName}) not saved";
162             }
163
164         return(true);
165     }
166     private function getAllFilesInDir($dirPath){
167             $fileListArray = Array();
168             $resultList= glob($dirPath.'*', GLOB_MARK | GLOB_NOSORT);
169             foreach($resultList as $item){
170                 if(substr($item,-1)!=DIRECTORY_SEPARATOR)
171                     $fileListArray[] = $item;
172                 else
173                     $fileListArray = array_merge($fileListArray, $this->   getAllFilesInDir($item));
174     }
175
176             return $fileListArray;
177         }
178
179     private function hashFile($filePath){
180         $fileString=file_get_contents($filePath);
181         return(md5($fileString));
182     }
183
184     private function fileIsExcluded($filePath){
185         foreach ($this->   exclusionStringArray as $label=>   $data){
186                 if (contains($data, strtolower($filePath))){
187                     $this->   excludedFilesArray[]=$filePath;
188                     return(true); //true means exclude, don't send, even if changed
189                 }
190             }
191
192         return(false); //false means it is NOT excluded and should be sent if changed
193     }
194
195     public function analyzeFiles(){
196         $this->   referenceFileArray=$this->   allFilesArray;
197         $potentiallyDeletedFilesArray=$this->   allFilesArray; //will remove found files to leave
deletable ones behind
198         $newAllFilesArray=array();
199
200         foreach ($this->   globArray as $label=>   $filePath){
201
202             $fileName=basename($filePath);
203
204             if (!$this->   fileIsExcluded($filePath)){
205                 $fileName=basename($filePath);
206                 $fileHash=$this->   hashFile($filePath);
207                 $nameHash=md5($filePath);
208
209
210                 unset($potentiallyDeletedFilesArray[$nameHash]);
211
212
213                 if (isset($this->   referenceFileArray[$nameHash])){
214
215                     if ($fileHash==$this->   referenceFileArray[$nameHash]['fileHash']){
216                         $this->   unchangedArray[$nameHash]=$this->   referenceFileArray[$nameHash];
217
218
219                         $newAllFilesArray[$nameHash]['fileHash']=$fileHash;
220                         $newAllFilesArray[$nameHash]['filePath']=$filePath;
```

```
221
222                 }
223                     else{
224
225                         $this->   needsUploadArray[$nameHash]['fileHash']=$fileHash;
226                         $this->   needsUploadArray[$nameHash]['filePath']=$filePath;
227
228                         $newAllFilesArray[$nameHash]['fileHash']=$fileHash;
229                         $newAllFilesArray[$nameHash]['filePath']=$filePath;
230
231                     }
232
233                 }
234                 else{
235                         $this->   newFileArray[$nameHash]['fileHash']=$fileHash;
236                         $this->   newFileArray[$nameHash]['filePath']=$filePath;
237
238                         $newAllFilesArray[$nameHash]['fileHash']=$fileHash;
239                         $newAllFilesArray[$nameHash]['filePath']=$filePath;
240
241
242                 }
243
244             }
245
246         }
247
248         $this->   deletedFilesArray=$potentiallyDeletedFilesArray; //all files found by glob were
removed
249         $this->   allFilesArray=$newAllFilesArray;
250     }
251
252     private function figureFtpPath($localPath){
253     /*
254         $localPath=realpath($localPath);
255         if (empty($localPath)){
256             return('');
257         }
258     */
259         $ftpOutPath=str_replace($this->   baseDirPath, '', $localPath);
260
261         if (is_array($this->   pathTranslationArray)){
262
263             foreach ($this->   pathTranslationArray as $label=>   $pathArray){
264
265                 if (contains($pathArray['local'], $localPath)){
266
267                     $ftpOutPath=str_replace($pathArray['local'], $pathArray['remote'], $localPath);
268
269                 }
270
271             }
272
273         }
274     return ($ftpOutPath);
275     }
276
277
278     public function sendFile($localPath){
279
280         $destPath=$this->   figureFtpPath($localPath);
281
282         if (!empty($destPath)){
283             try {
284
285
286
287                 if ($this->   dryRunFlag==false){
288
289                     $ftp = new Ftp;
290                     $ftp->   connect($this->   ftpHost);
291                     $ftp->   login($this->   ftpUser, $this->   ftpPass);
292                     $ftp->   pasv(true);
293
294                 //    $result=$ftp->nlist('./public_html'); //get name list, ie, ls, sort of
295
296
297                     $result=$ftp->   mkDirRecursive(dirname($destPath));
298
299                     $result=$ftp->   put('./'.$destPath, $localPath, FTP_BINARY);
```

```
300
301
302                 $ftp->   close();
303
304             $this->   uploadedFileCount++;
305             $this->   processStatusArray['ftpMessage']="uploaded {$this->   uploadedFileCount}
files";
306
307             }
308             else{
309
310                 $this->   processStatusArray['ftpMessage']="dry run flag set, no ftp
executed"     ;
311                 $result='';
312             }
313
314             $url=str_replace($this->   urlTranslationArray['docRootDir'], $this-
>   urlTranslationArray['baseUrl'], $destPath);
315
316             if(contains($this->   urlTranslationArray['baseUrl'], $url)==1){
317
318             //file is in docrRoot, needs link
319
320                 $report['url']=$url;
321
322             }
323             else {
324
325                 $report['url']='';
326             }
327
328         //    $report=$this->uploadFileTemplate;
329
330         //    $report=str_replace('<!link!>', $linkTemplate, $report);
331
332
333             $report['function']='upload';
334             $report['destPath']=$destPath;
335             $report['localPath']=$localPath;
336             $report['result']=$result;
337
338
339
340         }
341             catch (FtpException $e) {
342
343             $report['function']='upload';
344             $report['destPath']=$destPath;
345             $report['localPath']=$localPath;
346             $report['result']=$e->   getMessage();
347
348
349             }
350
351         $this->   uploadReportArray[]=processTemplateArray($this->   uploadFileTemplate, $report);
352
353     }
354
355  }
356  public function deleteFile($localPath){
357
358     $destPath=$this->   figureFtpPath($localPath);
359
360     if (!empty($destPath) or 5==5){
361         try {
362
363         if ($this->   dryRunFlag==false){
364
365             $ftp = new Ftp;
366             $ftp->   connect($this->   ftpHost);
367             $ftp->   login($this->   ftpUser, $this->   ftpPass);
368
369
370             $result=$ftp->   delete('./'.$destPath);
371             $ftp->   close();
372
373             }
374             else{
375             $this->   processStatusArray['deleteMessage']="dry run flag set, no ftp deletions
executed"     ;
```

```
376                 }
377
378
379                 $report['function']='delete';
380                 $report['destPath']=$destPath;
381                 $report['localPath']=$localPath;
382                 $report['result']=$result;
383
384
385             }
386                 catch (FtpException $e) {
387
388                 $report['function']='delete';
389                 $report['destPath']=$destPath;
390                 $report['localPath']=$localPath;
391                 $report['result']=$e->   getMessage();
392
393                 }
394
395             $this->   deleteReportArray[]=processTemplateArray($this->   uploadFileTemplate, $report);
396
397
398         }
399
400     }
401     public function executeFTP(){
402
403
404         foreach ($this->   newFileArray as $label=>   $data){
405
406             $this->   sendFile($data['filePath']);
407
408         }
409
410         foreach ($this->   needsUploadArray as $label=>   $data){
411
412             $this->   sendFile($data['filePath']);
413
414         }
415
416
417         foreach ($this->   deletedFilesArray as $label=>   $data){
418
419             $this->   deleteFile($data['filePath']);
420
421         }
422
423
424     }
425
426     public function showFileArray($fileArray){
427         $outString='';
428
429         foreach ($fileArray as $label=>   $data){
430             $showPath=str_replace($this->   baseDirPath, '', $data);
431             $showName=basename($data);
432
$outString.="<TR><TD>            $showName=</TD><TD>            $showPath</TD></TR>            ";
433         }
434
435         $outString="
436         <TABLE cellpadding=0 cellspacing=0 border=0>
437             $outString
438         </TABLE>
439         ";
440         echo $outString;
441     }
442
443     public function clearExclusionString(){
444         $this->   exclusionStringArray=array();
445         $this->   exclusionStringArray=$this->   mandatoryExcludes;
446         $this->   addExclusionString($this->   controlFileName);
447     }
448
449     public function addExclusionString($item){
450         $this->   exclusionStringArray[]=$item;
451     }
452
```

# File Source for camelCaseToUnderscore.php

*Documentation for this file is available at camelCaseToUnderscore.php*

```
1     <?php
2
3
4     function camelCaseToUnderscore($inString){
5         $inString=preg_replace('/^([A-Z])/e', 'strtolower("\1")'          , $inString);
6         $inString=preg_replace('/([A-Z])/', '_\1', $inString);
7         $inString=preg_replace('*/_*', '/', $inString); //don't want spurious slash if first letter is
upper case
8         $outString=strtolower($inString);
9         return $outString;
10    }
```

# File Source for contains.php

*Documentation for this file is available at contains.php*

```
1     <?php
2     function contains($pattern, $inString, $noCase='true')
3     {
4             if ($noCase=='true')
5             {
6                     $pattern=strtolower($pattern);
7                     $inString=strtolower($inString);
8             }
9             return(is_int(strpos($inString, $pattern)));
10    }
```

# File Source for dump.php

*Documentation for this file is available at dump.php*

```php
<?php
/**
 * Write out human readable and PHP includable data.  Keep the data
 * types the same, so null and false and 0 and '' are all different.
 * Something like an improved var_dump() or a neater var_export().
 *
 * Unlike the built-in functions, this does write out null, 0, '', and
 * false appropriately.  Think of it as a data pretty-printer or a data
 * formatter for writing include files.  It even detects and handles
 * circular references.
 *
 * Of course, it would be nice if it could dump the recursive variables
 * so that running the code would set the pointers up, but until I think of
 * an extra-clever way ....
 *
 * function dump($data, $name = false, $html = true, $echo_out = true)
 *   $data = What you want to dump
 *   $name = The name to display, great for naming the dumped data
 *   $html = Escape for HTML, turn off to make an include file
 *   $echo_out = Write to output, turn off if you want the content returned
 *
 * Also includes dump_callstack() to show where you are.
 */


/**
 * Here is some test code.  To enable, add a slash on the next line.
 *
 *
 * $a = array('int' => 0, 'boolean' => true, 'double' => 1.2345, 'string' => '');
 * dump($a, '$a');
 *
 * $b = array('array' => &$a, 'null' => null, 123 => 456);
 * dump($b, '$b');
 *
 * $fp = fopen('dump.php', 'r');
 * $c = array($b, 'file' => $fp);
 * $a[] = &$c;
 * $c[] = 'another thing';
 * dump($c, 'recursive');
 *
 * $type_tests = array(
 * array(0, 'zero'),  // integer
 * array(null, 'null'),  // string
 * array('', 'empty string'),  // string
 * array(false, 'false'),  // integer
 * array(1.2345, 'float'),  // integer
 * array(1.999, 'float'),  // integer
 * );
 *
 * foreach ($type_tests as $v) {
 * $d = array($v[0] => $v[1]);
 * dump($d, 'type_test_' . $v[1]);
 * }
 * exit();
 *
/**
 * End of test code.
 */


/**
 * Dumps data - Calls the __dump_anything() function for the real work
 */
function dump($data, $name = false, $html = null, $echo_out = true) {
    // Create a unique-ish ID
    $uniqId = md5(uniqid(mt_rand(), true));
```

```
68
69          if ($name) {
70              $uniqStack = array(
71                  array(
72                      'base',
73                      $name
74                  )
75              );
76          } else {
77              $uniqStack = array(
78                  array(
79                      'base',
80                      '_BASE'
81                  )
82              );
83          }
84
85          // If $html is null, determine if this is a web-based request
86          if ($html === null) {
87              if (php_sapi_name() != 'cli') {
88                  // Web request
89                  $html = true;
90              } else {
91                  // Command-line execution
92                  $html = false;
93              }
94          }
95
96          // Capture output if desired
97          if (! $echo_out) {
98              ob_start();
99          }
100
101         // HTML
102         if ($html) {
103             echo '<pre style="text-align: left;font-size:14pt;background:#aaaaaa;">'                ;
104         }
105
106         // Name
107         if ($name !== false) {
108             if ($html) {
109                 echo '<b>'       . htmlspecialchars($name) . '</b> = '      ;
110             } else {
111                 echo $name . ' = ';
112             }
113         }
114
115         __dump_anything($data, $html, 0, $uniqId, $uniqStack, false);
116
117         if ($name !== false) {
118             echo ";\n"          ;
119         } else {
120             echo "\n"          ;
121         }
122
123         if ($html) {
124             echo '</pre>'      ;
125         }
126
127         if (! $echo_out) {
128             $c = ob_get_clean();
129             return $c;
130         }
131
132         return false;
133     }
134
135
136     /**
137      * Determines how to write the data - calls __dump_* helper functions
138      */
139     function __dump_anything(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
140         $data_type = gettype($data);
141         $function = '__dump_' . $data_type;
142
143         if (! function_exists($function)) {
144             $function = '__dump_default';
145         }
146
147         $function($data, $html, $indentation, $uniqId, $uniqStack, $noFormat);
```

```
148   }
149
150
151   /**
152    * Writes out an array, avoids recursion.
153    */
154   function __dump_array(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
155       if (isset($data[$uniqId])) {
156           __dump_recursive($data[$uniqId], $html, $noFormat);
157           return;
158       }
159
160       $data[$uniqId] = $uniqStack;
161       echo 'array(';
162       __dump_collapse_start($html, $noFormat, count($data) - 1);
163       $indentation ++;
164       $isFirst = true;
165
166       foreach ($data as $k => &       $d) {
167           if ($k !== $uniqId) {
168               if (! $isFirst) {
169                   echo ',';
170               }
171
172               $isFirst = false;
173               echo "\n"           . str_repeat("\t"          , $indentation);
174               __dump_index($k, $html);
175
176               if ($html) {
177                   if ($noFormat) {
178                       echo ' =&gt; '    ;
179                   } else {
180                       echo ' <font color="#888a85">=&gt;</font> '                        ;
181                   }
182               } else {
183                   echo ' => '   ;
184               }
185
186               array_push($uniqStack, array(
187                       'array', &    $k
188                   ));
189               __dump_anything($d, $html, $indentation, $uniqId, $uniqStack, $noFormat);
190               array_pop($uniqStack);
191           }
192       }
193
194       unset($data[$uniqId]);
195       $indentation --;
196
197       if (count($data)) {
198           echo "\n"           . str_repeat("\t"          , $indentation);
199       }
200
201       __dump_collapse_end($html, $noFormat);
202       echo ')';
203   }
204
205
206   /**
207    * Starts a toggle so the user can show/hide large amounts of data
208    */
209   function __dump_collapse_start($html, $noFormat, $count) {
210       static $spanId = 0;
211
212       if (! $html || $noFormat) {
213           return;
214       }
215
216       if ($spanId == 0) {
217
218           ?><script language="JavaScript"><!--
219   function togC(which) {
220       var e = document.getElementById('togC' + which);
221       if (e) {
222           if (e.style.display == 'none') {
223               e.style.display = 'inline';
224           } else {
225               e.style.display = 'none';
226           }
227       }
```

```
228         return false;
229     }
230     // --></script><?php
231         }
232
233         echo ' <a href="#" style="text-decoration:none" onclick="return
togC(' . $spanId . ')">/*'          . $count . '*/</a> <span id="togC'              . $spanId .
'">'        ;
234         $spanId ++;
235     }
236
237     /**
238      * Closes a toggle
239      */
240     function __dump_collapse_end($html, $noFormat) {
241         if (! $html || $noFormat) {
242             return;
243         }
244
245         echo '</span>'      ;
246     }
247
248     /**
249      * Write out an index to an array or object.
250      *
251      * Only called from __dump_array, __dump_object, and __dump_recursive.
252      * Only handles known data types (try to keep it to strings and integers).
253      */
254     function __dump_index(&    $data, $html) {
255         $data_type = gettype($data);
256         $function = '__dump_' . $data_type;
257         $function($data, $html, 0, false, false, true);
258     }
259
260     /**
261      * Writes out an object, avoids recursion.
262      */
263     function __dump_object(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
264         if (isset($data->   $uniqId)) {
265             $uniqIdInfo = unserialize($data->   $uniqId);
266             __dump_recursive($uniqIdInfo, $html, $noFormat);
267             return;
268         }
269
270         $data->   $uniqId = serialize($uniqStack);
271
272         if (get_class($data) == 'DOMDocument') {
273             echo get_class($data) . '::loadXML(';
274             $s = $data->   saveXML();
275             __dump_anything($s, $html, $indentation, $uniqId, $uniqStack, $noFormat);
276             echo ')';
277             unset($data->   $uniqId);
278             return;
279         }
280
281         echo get_class($data) . '::__set_state(array(';
282         $variables = (array)$data;
283         __dump_collapse_start($html, $noFormat, count($variables) - 1);
284         $indentation ++;
285         $isFirst = true;
286
287         foreach ($variables as $k =>    $d) {
288             if ($k !== $uniqId) {
289                 if (! $isFirst) {
290                     echo ',';
291                 }
292
293                 $isFirst = false;
294                 echo "\n"           . str_repeat("\t"          , $indentation);
295                 $var_type = 0;  // 0 = public, 1 = protected, 2 = private
296                 $key_split = explode("\0"          , $k);
297
298                 if (count($key_split) == 3) {
299                     $k = $key_split[2];
300
301                     if ($key_split[1] == '*') {
302                         $var_type = 1;  // Key:  \0*\0NameOfVariable
303                     } else {
304                         $var_type = 2;  // Key:  \0ClassName\*NameOfVariable
305                     }
```

```
306                 }
307
308                 if ($html && !         $noFormat) {
309                     if ($var_type == 2) {
310                         echo '<font color="#cc0000">'                ;
311                     } elseif ($var_type == 1) {
312                         echo '<font color="#0000cc">'                ;
313                     } else {
314                         echo '<font color="#007700">'                ;
315                     }
316                 }
317
318                 __dump_index($k, $html);
319
320                 if ($html) {
321                     if ($noFormat) {
322                         echo ' =&gt; '    ;
323                     } else {
324                         echo '</font> <font
color="#888a85">=&gt;</font> '                        ;
325                     }
326                 } else {
327                     echo ' => '   ;
328                 }
329
330                 array_push($uniqStack, array(
331                         'object',
332                         $k
333                     ));
334                 __dump_anything($d, $html, $indentation, $uniqId, $uniqStack, $noFormat);
335                 array_pop($uniqStack);
336             }
337         }
338
339         unset($data->   $uniqId);
340         $indentation --;
341
342         if (count($data)) {
343             echo "\n"           . str_repeat("\t"          , $indentation);
344         }
345
346         __dump_collapse_end($html, $noFormat);
347         echo '))';
348     }
349
350     /**
351      * Writes out a recursion message.
352      *
353      * Called by __dump_array and __dump_object when recursion is detected.
354      */
355     function __dump_recursive(&    $uniqStack, $html, $noFormat) {
356         if ($html && !         $noFormat) {
357             echo '<font color="#770000">'                ;
358         }
359
360         echo 'RECURSIVE(';
361
362         foreach ($uniqStack as $member) {
363             switch ($member[0]) {
364                 case 'object':
365
366                     if ($html) {
367                         echo '-&gt;'     . htmlspecialchars($member[1]);
368                     } else {
369                         echo '->'    . $member[1];
370                     }
371
372                     break;
373
374                 case 'array':
375                     echo '[';
376                     __dump_index($member[1], $html);
377                     echo ']';
378                     break;
379
380                 default:
381
382                     if ($html) {
383                         echo htmlspecialchars($member[1]);
384                     } else {
```

```
385                     echo $member[1];
386                 }
387
388                 break;
389             }
390         }
391
392         echo ')';
393
394         if ($html && !         $noFormat) {
395             echo '</font>'      ;
396         }
397     }
398
399     /**
400      * Writes out a boolean value
401      */
402     function __dump_boolean(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
403         if ($data) {
404             if ($html && !         $noFormat) {
405                 echo '<font color="#00cc00">true</font>'                    ;
406             } else {
407                 echo 'true';
408             }
409         } else {
410             if ($html && !         $noFormat) {
411                 echo '<font color="#cc0000">false</font>'                    ;
412             } else {
413                 echo 'false';
414             }
415         }
416     }
417
418     /**
419      * Writes out an escaped string.
420      *
421      * To avoid additional overhead if the string is parsed, single quotes are
422      * preferred.
423      */
424     function __dump_string(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
425         $data_copy = $data;
426
427         if (preg_match('/[^ -~]/', $data)) {
428             // specialer case
429             $data_copy = str_replace('\\', '\\\\', $data_copy);
430             $data_copy = str_replace('"'     , '\\"'     , $data_copy);
431             $data_copy = str_replace('$', '\\$', $data_copy);
432             $data_copy = str_replace("\r"          , '\\r', $data_copy);
433             $data_copy = str_replace("\n"          , '\\n', $data_copy);
434             $data_copy = str_replace("\t"          , '\\t', $data_copy);
435             $data_copy = str_replace("\v"          , '\\v', $data_copy);
436             $data_copy = str_replace("\t"          , '\\t', $data_copy);
437             $data_copy = str_replace("\t"          , '\\t', $data_copy);
438             $data_copy = preg_replace('/([^ -~])/e', "'\x'.bin2hex(\"\\1\")"                    ,
$data_copy);
439             $data_copy = '"'      . $data_copy . '"'     ;
440         } else {
441             // Simple, no double quote needed
442             $data_copy = str_replace('\\', '\\\\', $data_copy);
443             $data_copy = str_replace('\'', '\\\'', $data_copy);
444             $data_copy = '\'' . $data_copy . '\'';
445         }
446
447         if ($html) {
448             $data_copy = htmlspecialchars($data_copy);
449
450             if (! $noFormat) {
451                 $sql = '';
452
453                 if (__dump_sql_test($data)) {
454                     $sql = __dump_sql($data);
455                 }
456
457                 $data_copy = $sql . '<font color="#0000cc">'                    . $data_copy .
'</font>'      ;
458             }
459         }
460
461         echo $data_copy;
462     }
```

```
463
464     /**
465      * Writes out an integer by calling __dump_double
466      */
467     function __dump_integer(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
468         __dump_double($data, $html, $indentation, $uniqId, $uniqStack, $noFormat);
469     }
470
471     /**
472      * Writes out a number.  No special formatting needed.
473      */
474     function __dump_double(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
475         if ($html && !         $noFormat) {
476             echo '<font color="#777700">'                . $data . '</font>'      ;
477         } else {
478             echo $data;
479         }
480     }
481
482     /**
483      * Writes out a null value
484      */
485     function __dump_NULL(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
486         echo 'null';
487     }
488
489     /**
490      * Writes out a resource
491      *
492      * If it is a stream resource, additional attributes are also shown.
493      */
494     function __dump_resource(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
495         $res_type = get_resource_type($data);
496         $out = 'null /* resource(' . intval($data) . ', ' . $res_type;
497
498         if ($res_type == 'stream') {
499             $meta = stream_get_meta_data($data);
500             $out .= ', "'       . $meta[uri] . '", "'           . $meta[mode] . '"'     ;
501         }
502
503         $out .= ') */';
504
505         if ($html) {
506             echo htmlspecialchars($out);
507         } else {
508             echo $out;
509         }
510     }
511
512     /**
513      * This is the default catch-all that should never get used.
514      */
515     function __dump_default(&    $data, $html, $indentation, $uniqId, $uniqStack, $noFormat) {
516         echo 'DUMPING TYPE OF ' . gettype($data) . ' NOT SUPPORTED';
517     }
518
519     /**
520      * Determine if we should allow the display of this statement as SQL
521      */
522     function __dump_sql_test(&    $data) {
523         /* Match on any of the following patterns:
524          * select _multiple_ from
525          * inset into _multiple_ values
526          * update _table_ set
527          * alter table
528          */
529         if
(preg_match('/^(select\s.*\sfrom|insert\s+into\s.*\svalues|update\s+[^\s]+\sset|alter\s+table)\s/i',
$data)) {
530             return true;
531         }
532
533         return false;
534     }
535
536     /**
537      * Write out JS to show the select statement in a new window.
538      * Do not pass $data by reference for this function.
539      */
540     function __dump_sql($data) {
```

```
541         $data = trim($data);
542         $dataPos = 0;
543         $dataLen = strlen($data);
544         $out = '';
545         $token = '';
546         $indent = 1;
547 
548         // Chew through $data
549         while ($dataPos <    $dataLen) {
550             $c = substr($data, $dataPos, 1);
551 
552             if (preg_match('/[a-zA-Z0-9]/', $c)) {
553                 $token .= $c;
554             } elseif ($c == '\'' || $c == '"'      || $c == '`') {
555                 if ($token != '') {
556                     $out .= __dump_sql_token($token, $indent);
557                     $token = '';
558                 }
559 
560                 // Find matching quote
561                 $startPos = $dataPos;
562                 $dataPos ++;
563 
564                 while (($dataPos <    $dataLen) && (        $cMatch = substr($data, $dataPos, 1)) !=
$c) {
565                     if ($cMatch == '\\') {
566                         $dataPos ++;
567                     }
568 
569                     $dataPos ++;
570                 }
571 
572                 $out .= substr($data, $startPos, 1 + $dataPos - $startPos);
573             } else {
574                 if ($token != '') {
575                     $out .= __dump_sql_token($token, $indent);
576                     $token = '';
577                 }
578 
579                 $out .= $c;
580 
581                 if ($c == ',') {
582                     $out .= "\n"           . str_repeat("\t"          , $indent);
583                     $c = substr($data, $dataPos + 1, 1);
584 
585                     while ($c == ' ' || $c == "\t"           || $c == "\r"           || $c ==
"\t"          ) {
586                         $dataPos ++;
587                         $c = substr($data, $dataPos + 1, 1);
588                     }
589                 } elseif ($c == '(') {
590                     $indent ++;
591                 } elseif ($c == ')') {
592                     $indent --;
593                 }
594             }
595 
596             $dataPos ++;
597         }
598 
599         if ($token != '') {
600             $out .= __dump_sql_token($token, $indent);
601             $token = '';
602         }
603 
604         $out = trim($out);
605         return __dump_sql_html($out);
606     }
607 
608 
609     function __dump_sql_token($token, $indent) {
610         static $lastTokenBr = false;
611         $tokenUpper = strtoupper($token);
612         $caps = array(
613             'AS',
614             'INTO',
615             'NULL',
616             'ON'
617         );
618         $br = array(
```

```
619             'ALTER',
620             'FROM',
621             'GROUP',
622             'HAVING',
623             'INNER',
624             'INSERT',
625             'JOIN',
626             'LEFT',
627             'LIMIT',
628             'ON',
629             'ORDER',
630             'RIGHT',
631             'SELECT',
632             'SET',
633             'UPDATE',
634             'VALUES',
635             'WHERE'
636         );
637         $brIndent = array(
638             'AND'
639         );
640 
641         if (in_array($tokenUpper, $caps)) {
642             $lastTokenBr = false;
643             return $tokenUpper;
644         }
645 
646         if (in_array($tokenUpper, $br)) {
647             if ($lastTokenBr) {
648                 return $tokenUpper;
649             }
650 
651             $lastTokenBr = true;
652             return "\n"           . str_repeat("\t"          , $indent - 1) . $tokenUpper;
653         }
654 
655         if (in_array($tokenUpper, $brIndent)) {
656             if ($lastTokenBr) {
657                 return $tokenUpper;
658             }
659 
660             $lastTokenBr = true;
661             return "\n"           . str_repeat("\t"          , $indent) . $tokenUpper;
662         }
663 
664         $lastTokenBr = false;
665         return $token;
666     }
667 
668 
669     function __dump_sql_html(&    $sql) {
670         $stack = debug_backtrace();
671 
672         if (! isset($GLOBALS['__debug_sql_counter'])) {
673             $GLOBALS['__debug_sql_counter'] = 1;
674             $id = 1;
675         } else {
676             $id = ++ $GLOBALS['__debug_sql_counter'];
677         }
678 
679         ob_start();
680 
681         ?><script type='text/javascript' language='JavaScript'><?php
682 
683         if ($id == 1) { ?>
684     function debugSqlWindow(t) {
685         var h = "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"
\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n";
686         h += "<html xmlns=\"http://www.w3.org/1999/xhtml\" xml:lang=\"en\"
lang=\"en\">\n";
687         h += "<head><title>SQL Statement Print
Box</title><head><body>\n";
688         h += "<div style=\"border: 2px outset blue; padding: 5px; background-color:
#c7e0ff;\">\n";
689         h += t;
690         h += "</div><br>\n";
691         h += "<form><p align=\"center\"><input type=\"button\"
value=\"Close\" onClick=\"window.close();\"
/></p></form></body></html>\n";
692 
```

```
693         var w = window.open('', 'newWin', 'width=1000,height=500,scrollbars=yes');
694         w.document.write(h);
695         w.document.close();
696         w.focus();
697         return false;
698     }
699     <?php
700         } ?>
701     var debugText<?php echo    $id; ?> = "<?php
702
703         $out = htmlspecialchars($sql);
704         $out = addslashes($out);
705         $out = str_replace("\n"          , '<br>\\n'      , $out);
706         $out = str_replace("\t"          , '        '               , $out);
707         echo $out;
708
709         ?>";
710     </script><a href="#" onClick="return debugSqlWindow(debugText<?php echo
$id; ?>);">/*Show Window*/</a> <?php                    return ob_get_clean();
711     }
712
713     /**
714      * Dump the call stack to the current point.
715      *
716      * $return = false (wrap in HTML and echo), or true (return text)
717      */
718     function dump_callstack($return = false) {
719         $out = array();
720
721         foreach (debug_backtrace() as $trace) {
722             if (isset($trace['file'])) {
723                 $out[] = $trace['file'] . ', line ' . $trace['line'];
724             }
725
726             $str = "\t"          ;
727
728             if (isset($trace['type'])) {
729                 $str .= $trace['class'] . $trace['type'];
730             }
731
732             $str .= $trace['function'] . '(';
733             $comma = '';
734
735             if (is_array($trace['args'])) {
736                 foreach ($trace['args'] as $arg) {
737                     $str .= $comma;
738                     $comma = ', ';
739
740                     switch (gettype($arg)) {
741                         case 'array':
742                             $str .= '(array)';
743                             break;
744
745                         case 'object':
746                             $str .= get_class($arg);
747                             break;
748
749                         case 'boolean':
750
751                             if ($arg) {
752                                 $str .= 'true';
753                             } else {
754                                 $str .= 'false';
755                             }
756
757                             break;
758
759                         case 'string':
760                             $str .= '"'      . addslashes($arg) . '"'     ;
761                             break;
762
763                         case 'integer':
764                         case 'double':
765                             $str .= $arg;
766                             break;
767
768                         case 'NULL':
769                             $str .= 'NULL';
770                             break;
771
```

```
772                     case 'resource':
773                         $str .= '(resource)';
774                         break;
775 
776                     default:
777                         $str .= '(unknown type: ' . gettype($arg) . ')';
778                 }
779             }
780         }
781 
782         $str .= ')';
783         $out[] = $str;
784     }
785 
786     if ($return) {
787         // Let's not include the call to dump_callstack()
788         array_shift($out);
789         array_shift($out);
790         return implode("\n"          , $out);
791     }
792 
793     echo '<pre>'      ;
794 
795     foreach ($out as $o) {
796         echo htmlspecialchars($o) . "\n"          ;
797     }
798     dump_callstack();
799     echo '</pre>'      ;
800 }
```

# File Source for processTemplateArray.php

*Documentation for this file is available at processTemplateArray.php*

```
1     <?php
2     error_reporting(E_ERROR | E_WARNING | E_PARSE);
3         if (!function_exists("processSubstitutions"          ))
4                 {
5                     function processSubstitutions($template, $inputArray, $ctlArray='', $parmArray='',
$evalArrays='', $dataArrays='')
6                         {
7                             reset($inputArray);
8                             global $PHP_SELF; //TQ made this global for use in eval code, 7/07
9                             $recordArray=$inputArray;
10                            $patternArray=$globalPatternArray; $replaceArray=$globalReplaceArray;
11                            if (is_array($parmArray)) //this is placed here specifically to prevent the
override of inputArray values
12                                {
13                                    while (list($label, $data)=each($parmArray))
14                                        {
15                                            if (!is_int($label))
16                                                {
17                                                    $tag="     <!   $label!>   "     ;
18                                                    if (is_string($data))
19                                                        {
20                                                            $patternArray[$tag]=$tag;
21                                                            $replaceArray[$tag]=$data;
22                                                            $inx=$inx+1;
23                                                            }
24                                                    }
25                                            }
26                                    }
27                            while (list($label, $data)=each($recordArray))
28                                {
29                                    if (!is_int($label))
30                                        {
31                                            $tag="     <!   $label!>   "     ;
32                                            $patternArray[$tag]=$tag;
33                                            $replaceArray[$tag]=$data;
34                                            $inx=$inx+1;
35                                            }
36                                    }
37                            //execute eval arrays
38                                //reset eval arrays
39                                    reset($evalArrays['calc']);
40                                    reset($evalArrays['conv']);
41                                if ($evalArrays['conv'])
42                                    {
43                                        while (list($label, $data)=each($evalArrays['conv']))
44                                            {
45                                                if (!is_int($label))
46                                                    {
47                                                        $tag="     <!   $label!>   "     ;
48
49                                                        $data=str_replace($patternArray, $replaceArray,
$data);
50                                                        $pattern=$tag;
51
52                                                        if ($ctlArray["debug"          ]) { echo
"<HR>Conv: "               .htmlentities($tag)."="          .htmlentities($data)."<BR>"               ;}
53
54                                                        $replace=eval($data);
55
56                                                        $patternArray[$tag]=$pattern;
57                                                        $replaceArray[$tag]=$replace;
58
59
60                                                        if ($ctlArray["debug"          ]) { echo
"     <P>Conv:       $tag result="     .htmlentities($replaceArray[$tag])."<HR>"               ;}
61                                                        $inx=$inx+1;
62                                                        }
```

```
63                                          }
64                                      }
65                                  if ($evalArrays['calc'])
66                                      {
67                                          while (list($label, $data)=each($evalArrays['calc'])) //init
replacement with field values
68                                              {
69                                                  if (!is_int($label))
70                                                      {
71                                                          $tag="     <!   $label!>   "     ;
72                                                          $patternArray[$tag]=$tag;
73                                                          $data=str_replace($patternArray, $replaceArray,
$data);
74                                                          if ($ctlArray["debug"          ]) { echo
"Eval: "          .htmlentities($tag)."="          .htmlentities($data)."<BR>"              ;}
75                                                          $replaceArray[$tag]=eval($data);
76                                                          $inx=$inx+1;
77                                                          }
78                                                  }
79                                              }
80                                  reset($patternArray); reset($replaceArray);
81                                  $outString=str_replace($patternArray, $replaceArray, $template);
82                                  return($outString);
83                                  }
84                          }
85
86
87
88          if (!function_exists('replaceArray')) //and this one is new
89                  {
90                      function replaceArray($inString, $inArray)
91                          {
92                              $i=0;
93                              while (list($label, $data)=each($inArray))
94                                  {
95                                      $patternArray[$i]="     <!   $label!>   "     ;
96                                      $replaceArray[$i]="     $data"     ;
97                                      $i=$i+1;
98                                      }
99                              $outString=str_replace($patternArray, $replaceArray, $inString);
100                             return($outString);
101                             }
102                     }
103
104     if (!function_exists('processText')) //and this one is new
105         {
106             function processText($inString, $autoParagraphs='on', $processOutline='off',
$activeURLs='off', $keepLineBreaksFlag='false', $skipURLdecode='true')
107                 {
108                     //also defined in showOutline in categoryCtl
109                     //changed $skipURLdecode default to 'true' 11/2005 TQ, I think I pretty much always
want to skip that special processing
110                     //plus it was causing all plus signs (+) to disappear from all content
111                     // added this below to compensate $workingString=stripslashes($workingString);
112                     if (!function_exists('activeURLs2')) //I have to declare these versions because I
can't seem to access this class's methods
113                         {
114                             //=========NOTE: this is used outside of classLibs, don't add '$this->'
refs ==========
115     function activeURLs2($textString)
116         {
117             $pattern="^http://([a-zA-Z0-9_./\?=\&\-]*)[ \n\r\t].*"              ;
118             $pattern="^http://([a-zA-Z0-9:%$,\+_./\?=\&\-]*)[< \x0D\x0A\n\r\t'>].*"                  ;
119
120             $inx=1;
121             $workingString=$textString;
122             $nextTagInx=strpos($workingString, 'http');
123             $workingString=substr($workingString, $nextTagInx); //through the end of string
124             $workingString.=" -"          ; //ereg doesn't seem to like finding no space or character
after last URL
125             while (ereg($pattern, $workingString, $partsArray))
126                 {
127
128                     $data=$partsArray[1];
129                     $garbage='http://'.$data;
130                     $replaceTag="     <!   $inx!>   "     ;
131
132                     $replaceArray[$inx]="     <A href=http://   $data>   $data</A>      "     ;
133                     $patternArray[$inx]=$garbage;
134
```

```
135                 $inx=$inx+1;
136                 $partsArray=""          ;
137                 $data=""          ;
138
139                 $workingString=substr($workingString, 3);
140                 $nextTagInx=strpos($workingString, 'http');
141                 $workingString=substr($workingString, $nextTagInx); //through the end of string
142
143                 }
144         $textString=str_replace($patternArray, $replaceArray, $textString);
145         return($textString);
146         }
147
148                         // ================================================START of processOutline
149     function processOutline2($inString)
150         {
151             $cr=chr(13);
152             $lineArray=explode($cr, $inString);
153             while (list($label, $line)=each($lineArray))
154                 {
155                     $length=strlen($line);
156                     $prefix='';
157                     for ($i=1; $i<   $length, $i=$i+1;)
158                         {
159                             $char=substr($line, $i, 1);
160                             if ($char!=' ')
161                                 { break; }
162                             $prefix.=' '    ;
163                             }
164                         $line=$prefix.trim($line);
165                         $outString.=$line.$cr;
166                     }
167             $outString=rtrim($outString);
168             return ($outString);
169             }
170
171                         }
172                 $workingString=$inString;
173                 if ($skipURLdecode=='false')
174                     {
175                         $workingString=stripslashes(urldecode($workingString));
176                         }
177                 if ($autoParagraphs=='on')
178                     {
179                         $cr=chr(13); $nl=chr(10);
180                         $patternArray[$inx]="     $cr$nl$cr$nl"     ;
$replaceArray[$inx]="<P>"                   ; $inx=$inx+1;
181                         $patternArray[$inx]="     $cr$cr"     ;
$replaceArray[$inx]="<P>"                   ; $inx=$inx+1;
182                         if ($keepLineBreaksFlag=='keepLineBreaks' or $keepLineBreaksFlag=='true' or
$keepLineBreaksFlag=='on')
183                             {
184                                 $patternArray[$inx]="     $cr$nl"     ;
$replaceArray[$inx]="<BR>"                      ; $inx=$inx+1;
185                                 $patternArray[$inx]="     $cr"     ;
$replaceArray[$inx]="<BR>"                      ; $inx=$inx+1;
186                             }
187                         $workingString=str_replace($patternArray, $replaceArray, $workingString);
188                         }
189                 if ($activeURLs=='on')
190                     {
191                         $workingString=activeURLs2($workingString);
192                         }
193                 if ($processOutline=='on')
194                     {
195                         $workingString=processOutline2($workingString);
196                         }
197                 /*
198                 if ($activeURLs=='on')
199                     {
200                         $workingString=$this->activeURLs($workingString);
201                         }
202                 */
203                 $workingString=stripslashes($workingString);
204                 return($workingString);
205                 }
206         }
207
208
209     if (!function_exists('replaceCompoundTags')) //and this one is new
```

```
210             {
211                 function replaceCompoundTags($inString, $dataArray2,
$parmArray="noDBtbl"          , $debug=""          )
212       {
213           //this was replacePseudoTags but has legacy functions removed
214           //this is called (as a macro) by processTemplateArray in channelInitPages, categoryCtl and
channelCtl
215           $workingString=$inString;
216           $imageLink=$parmArray['imageLink'];
217           $pattern="<!([a-zA-Z0-9:()' ;/,\.$=-_]+)!>.*"              ;
218           $patternLength=strlen($pattern);
219           $stringLength=strlen($workingString);
220           $partsArray=""          ;
221           $stringIndex=-1;
222           do //EXTRACT <!fields!> from message
223               {
224                   $foundFlag=ereg($pattern, $workingString, $itemArray);
225                   $parameterArray='';
226                   if ($foundFlag>   0 or $foundFlag===0) //create tag list and get values
227                       {
228
229                           $item=$itemArray[1];
230                           if ($item) { $partsArray[$item]="found"          ; }
231                           $tagParts=explode(":"          , $item);
232                               $tagID=$tagParts[0];
233                               $tagData=$tagParts[1];
234
235                           if ($tagData!='') //null tagData means it's not a compound tag, ignore
236                               {
237                                   $inx=urlencode($item); //creates unique and locatable index
238                                   $valueFound='false';
239                                   switch ($tagID)
240                                       {
241                                           /*case 'macro':
242                                               //getPIXurl2($queryValue, $mode="5",
$size="large", $queryField='glossName', $specHeight="", $specWidth="",
$altTag="")
243                                               function getPIXurl2($queryValue, $mode="5",
$size="large", $queryField='glossName', $specHeight="", $specWidth="",
$altTag="")
244                                                   {
245                                                       global $imageLink;
246                                                       return($imageLink->getPIXurl2($queryValue,
$mode, $size, $queryField, $specHeight, $specWidth, $altTag));
247                                                       }
248                                               $tmp=eval($tagData);
249                                               if ($tmp!='')
250                                                   {
251                                                       $valuesArray[$inx]=$tmp;
252                                                       $valueFound='true';
253                                                       }
254                                               break;
255                                               */
256                                           case 'image':
257                                               $firstSpace=strpos($tagData, " "          );
258                                               $glossName=substr($tagData, 0, $firstSpace);
259                                               $tagData=substr($tagData, $firstSpace);
260                                               $tagData=trim($tagData);
261                                               $tagArray=explode(" "          , $tagData);
262                                               if (!$glossName) {$glossName=$tagData;}
263
264                                               while (list($label, $data)=each($tagArray))
265                                                   {


266                                                       $parmElementArray=explode("="          , $data);
267
$parameterArray[$parmElementArray[0]]=$parmElementArray[1];
268                                                       //height=00
269                                                       //width=00
270                                                       //size=large or  small
271                                                       //scale=0%
272                                                       //function=getPIXurl2
273                                                       }
274                                               if (!$parameterArray['function'])
{$parameterArray['function']='getPIXurl2';}
275                                               if ($parameterArray['function']=='getPIXurl2' or 5==6)
276                                                   {
277                                                       $fName=$parameterArray['function'];
278                                                       $tmp=$imageLink->   $fName($glossName,
```

```
$parameterArray);
279                                                 if ($tmp!='')
280                                                     {
281
282                                                         $valuesArray[$inx]=$tmp;
283                                                         $valueFound='true';
284                                                         }
285                                                 }
286                                             else
287                                                 {
288                                                     $tmp="      (Function Name in pseudoTag,
'{$parameterArray['function']}', not known.) "     ;
289                                                     if ($tmp!='')
290                                                         {
291                                                             $valuesArray[$inx]=$tmp;
292                                                             $valueFound='true';
293                                                             }
294                                                     }
295                                                 break;
296                                             default:
297                                                 if ($dataArray2[$tagID][$tagData])
298                                                     {
299                                                         $tmp=$dataArray2[$tagID][$tagData];
300                                                         if ($tmp!='')
301                                                             {
302                                                                 $valuesArray[$inx]=$tmp;
303                                                                 $valueFound='true';
304                                                                 }
305                                                         }
306                                                 break;
307                                                 }
308                                     if ($valueFound=='true')
309                                         {
310                                             $stringArray[$inx]="     <!   $item!>   "     ;  //ie,
patternArray using later language
311                                             }
312                                     }
313                             }
314                     //prepare the remainder of the string
315                         $tag="     <!   $item!>   "     ;
316                         $workingString=strchr($workingString, $tag);
317                         $workingString=substr($workingString, strlen($tag));
318                         $stringLength=strlen($workingString);
319
320                         $itemArray=""          ;
321                         $item=""          ;
322                     }
323                 while ($stringLength>   0);
324             if ($debug=='true') { echo "<P><B>Compound TAG
Replacement</B><BR>"                 ; }
325
326             if ($debug=='true') { echo
"<P>inString="                .htmlentities($inString)."<P>"                ; }
327             if ($debug=='true') { while (list($label, $data)=each($stringArray)) {  echo
"rreplacing "          .htmlentities($stringArray[$label])." with
"     .htmlentities($valuesArray[$label])."      (inx=$label)<BR>      "     ; } }
328
329             $outString=str_replace($stringArray, $valuesArray, $inString);
330
331             if ($debug=='true') { echo "     <P>outString=      $outString<P>      "     ; }
332             return($outString);
333             }
334
335                     }
336
337
338   function processTemplateArray($template, $inputArray, $inCtlArray='', $inParmArray='',
$inEvalArrays='', $inDataArrays='')
339       {
340       //init functions
341
342
343
344
345
346       //grab global elements if available
347           if (!$inCtlArray) { global $inCtlArray;}
348           if (!$inParmArray) { global $inParmArray;}
349           if (!$inEvalArrays) { global $inEvalArrays;}
350           if (!$inDataArrays) { global $inDataArrays;}
```

```
351       //establish default arrays (some default items are not applicable in this single record context)
352           //create default$ctlArray
353               $ctlArray["itemTemplate"          ]="<!bodyText!><P>"                      ;
354               $ctlArray["blockTemplate"          ]="<!itemText!><!editLink!>"                    ;
355               $ctlArray["startRec"           ]="0"          ;
356               $ctlArray["count"          ]="1"          ;
357               $ctlArray["orderString"          ]="seqNum DESC"          ;
358               $ctlArray["queryString"          ]="channelCode='<!channelCode!>' and
siteCode='<!siteCode!>'"          ; //<!channelCode!> and <!siteCode!> come from $parmArray
359               $ctlArray["debug"          ]=""          ; //if it's defined at all, it triggers SQL
messages and other stuff to help figure out what's wrong
360               //ctlArray['autoParagraphs'] added 2/05, if present serves as default if not present in
data record
361               //label $ctlArray items
362                   $ctlArray["nextLinkLabel"          ]="next"          ;
363                   $ctlArray["prevLinkLabel"          ]="prev"          ;
364                   $ctlArray["moreLinkLabel"          ]="more"          ;
365                   $ctlArray["permaLinkLabel"          ]="#"          ;
366                   $ctlArray["viewItemLabel"          ]="view"          ;
367
368           //create default $parmArray
369               global $siteCode;
370               $parmArray["siteCode"          ]='siteCode not defined by global or parmArray';
371               $parmArray["userID"          ]=""          ;
372               global $imageLink;
373               $parmArray["imageLink"          ]=$imageLink;
374               $parmArray["previewPage"          ]=$PHP_SELF; //the page we return to after
manageChannel entry
375               $parmArray["editPage"          ]=$PHP_SELF; //manageChannel.php
376               $parmArray["infoPage"          ]=$PHP_SELF; //the place to go to view a permalink
377               $parmarray['editPermission']='false'; //=='true' turns on the default eval for
<!editLink!>
378           //create default $evalArrays (blockCalc, itemCalc, conv)
379
380               //$evalArrays['conv']["lastUseTime"]='return(date("m/d/y h:i a",
"<!lastUseTime!>"));'; //redefine <!lastUseTime!>
381               //$evalArrays['conv']["creationTime"]='return(date("m/d/y h:i a",
"<!creationTime!>"));'; //redefine <!creationTime!>
382
//$evalArrays['conv']["bodyText"]='return(processText("<!bodyText!>",
"<!autoParagraphs!>", "<!processOutLine!>",
"<!activeURLs!>"));';
383
//$evalArrays['conv']["headline"]='return(processText("<!headline!>",
"<!autoParagraphs!>", "<!processOutLine!>",
"<!activeURLs!>"));';
384
$evalArrays['conv']["bodyText"          ]='return(stripslashes(processText($recordArray["bodyText&quo
t;], "<!autoParagraphs!>", "<!processOutLine!>",
"<!activeURLs!>", "<!keepLineBreaksFlag!>")));'                                    ;
385
$evalArrays['conv']["headline"          ]='return(stripslashes(processText($recordArray["headline&quo
t;], "<!autoParagraphs!>", "<!processOutLine!>",
"<!activeURLs!>", "<!keepLineBreaksFlag!>")));'                                    ;
386
387               $evalArrays['calc']['slashedBodyText']='return($recordArray["bodyText"]);'          ;
388               $evalArrays['calc']['slashedHeadline']='return($recordArray["headline"]);'          ;
389               $evalArrays['calc']['permaLink']='return("<A
href=<!infoPage!>?permaLink=<!itemID!>><!permaLinkLabel!></A>");'                                  ;
390               $evalArrays['calc']['itemEditLink']='if
("<!editPermission!>"=="true") {return(" (<A
href=<!editPage!>?operation=editChannel&itemID=<!itemID!>&channelCode=<!channelCode
!>&siteCode=<!siteCode!>&previewPage=<!previewPage!>><SPAN
class=editLink>e</SPAN></A>/<A
href=<!editPage!>?operation=newChannel&channelCode=<!channelCode!>&siteCode=<!siteC
ode!>&previewPage=<!previewPage!>><SPAN
class=editLink>n</SPAN></A>)");} else{return("");}'                                 ;
391
392               // revised for the following 10/2004
$evalArrays['blockCalc']['moreLink']='return("<A
href=<!infoPage!>?operation=viewMore&channelCode=<!channelCode!>&startRec=<!nextRec
No!>><!moreLinkLabel!></A>");'; //create new <!moreLink!>
393               $inEvalArrays['blockCalc']['moreLink']='return("<A
href=<!infoPage!>?operation=viewMore&channelCode=<!channelCode!>&siteCode=<!siteCod
e!>&startRec=<!nextRecNo!>><!moreLinkLabel!></A>");'                                ; //create new
<!moreLink!>
394               $evalArrays['blockCalc']['editLink']='if
("<!editPermission!>"=="true") {return(" (<A
href=<!editPage!>?operation=editChannel&itemID=<!lastItemID!>&channelCode=<!channel
Code!>&siteCode=<!siteCode!>&previewPage=<!previewPage!>><SPAN
```

```
class=editLink>e</SPAN></A>/<A
href=<!editPage!>?operation=newChannel&channelCode=<!channelCode!>&siteCode=<!siteC
ode!>&previewPage=<!previewPage!>><SPAN
class=editLink>n</SPAN></A>)");} else{return("");}'                                    ;
395             //$evalArrays['blockCalc']['editLink']='return(" (<A
href=<!editPage!>?operation=editChannel&itemID=<!lastItemID!>&channelCode=<!channel
Code!>&siteCode=<!siteCode!>&previewPage=<!previewPage!>
class=editLink>e</A>/<A
href=<!editPage!>?operation=newChannel&channelCode=<!channelCode!>&siteCode=<!siteC
ode!>&previewPage=<!previewPage!> class=editLink>n</A>)");';
396             //blockCalc only refers to last record's field values and are calculated last
397
398         //create default $dataArrays
399             $dataArrays['glossary']['codeCopyright']="Copyright TQ White II 2003 -
"     .date("Y"          );
400         //update arrays with incoming (user) parameters
401         if ($inCtlArray) { reset($inCtlArray); while (list($label, $data)=each($inCtlArray)) {
$ctlArray[$label]=$data; } } //add/replace incoming parameters
402         if ($inParmArray) { reset($inParmArray); while (list($label, $data)=each($inParmArray)) {
$parmArray[$label]=$data; } } //add/replace incoming parameters
403         if ($channelCode) { $parmArray["channelCode"          ]=$channelCode; }//the parameter in
the function call is for convenience, the routine operates on arrays
404         if ($inEvalArrays)
405             {
406                 reset($inEvalArrays);
407                 while (list($glossName, $array)=each($inEvalArrays))
408                     {
409                         if (is_array($array))
410                             {
411                                 while (list($label, $data)=each($array))
412                                     {
413                                         $evalArrays[$glossName][$label]=$data;
414                                         }
415                                 }
416                         }
417                 }
418         if (is_array($inDataArrays))
419             {
420                 reset($inDataArrays);
421                 while (list($glossName, $array)=each($inDataArrays))
422                     {
423                         if (is_array($array))
424                             {
425                                 while (list($label, $data)=each($array))
426                                     {
427                                         $dataArrays[$glossName][$label]=$data;
428                                         }
429                                 }
430                         }
431                 }
432         //*****************************************
433         //<B>this usually  is called from channelInitPages</B>
434         //*****************************************
435
436         if (!is_array($inputArray))
437             {
438                 $outString="<DIV>processTemplateArray() says: input is not a record
array</DIV>"          ;
439                 }
440             elseif (is_array(pos($inputArray))) //ie, array of arrays from getRecordSet
441                 {
442                     reset($inputArray);
443                     $recInx=1;
444                     //$dataArrays['prevRec'] could be initialized from calling prog
445                     while (list($label, $recordArray)=each($inputArray))
446                         {
447                             $recordArray['recInx']=$recInx; $recInx++;
448                             $tmp=processSubstitutions($template, $recordArray, $ctlArray, $parmArray,
$evalArrays, $dataArrays);
449                             if ($ctlArray["debug"          ]=='true') { echo "<B>simple tag
replacement Result</B><P>processString="                .htmlentities($tmp)."<P>"                ; }
450                             $tmp=replaceCompoundTags($tmp, $dataArrays, $parmArray, $ctlArray['debug']);
451                             $outString.=$tmp;
452                             $dataArrays['prevRec']=$recordArray;
453                             }
454
455                 }
456                 else //conventional, array of scalars
457                     {
458                         $recordArray=$inputArray;
```

```
459                     if (!$recInx)
460                         {
461                             $recInx=1;
462                             }
463                         else
464                             {
465                                 $recInx=$recInx+1;
466                                 }
467                     $recordArray['recInx']=$recInx;
468                     $tmp=processSubstitutions($template, $recordArray, $ctlArray, $parmArray,
$evalArrays, $dataArrays);
469                     if ($ctlArray["debug"          ]=='true') { echo "<B>simple tag
replacement Result</B><P>processString="                .htmlentities($tmp)."<P>"               ; }
470                     $tmp=replaceCompoundTags($tmp, $dataArrays, $parmArray, $ctlArray['debug']);
471                     $outString.=$tmp;
472                     }
473           return($outString);
474       }
```

# File Source for index.ctp

*Documentation for this file is available at index.ctp*

```
1     <?php
2
3
4     $outString="
5     <div style=font-family:sans-serif;margin-bottom:15px;>
6     <div style='margin:15px;border:1pt solid gray;background:#BBBB66;color:#994444;font-family:sans-
serif;height:100px;width:400px;padding-top:20px;text-align:center;'>
7     <div style=font-weight:bold;margin-bottom:5px;>
8     This is page one.
9     </div>
10    In this case, we have static text. The controller serves only as a dispatcher. It provides no
values.
11    </div>
12    <div style=font-size:80%;margin-left:15px;>Brought to you
by<BR>"           .str_replace('tq.org', 'tq.org<BR>'      , __FILE__)."</div>
13    </div>
14    "     ;
15
16
17    echo $outString;
```

# File Source for page2.ctp

*Documentation for this file is available at page2.ctp*

```
1     <?php
2
3
4     $outString="
5     <div style=font-family:sans-serif;margin-bottom:15px;>
6     <div style='margin:15px;border:1pt solid gray;background:#bb66bb;color:#aaffaa;font-family:sans-
serif;width:400px;padding-top:20px;text-align:center;'>
7     <div style=font-weight:bold;margin-bottom:5px;>
8     This is page TWO.
9     </div>
10    This example makes use of the new facility that works correctly on URLs of the form<br/>
11    http://domain.com/controllerName/parameter<br/>
12    Of course, it also works with<br/>
13
14    http://domain.com/controllerName/parameter/action/parameter, too<br/>
15    </div>
16    <div style=font-size:80%;margin-left:15px;>Brought to you
by<BR>"           .str_replace('tq.org', 'tq.org<BR>'      , __FILE__)."</div>
17    </div>
18    "     ;
19
20
21    echo $outString;
```

# File Source for environment_info.ctp

*Documentation for this file is available at environment_info.ctp*

```
1     <?php
2
3     //echo "<div style=height:50px;><a href=#phpInfo>Jump Down to
phpinfo()</A></div>";
4     //dump($_SERVER, '_SERVER');
5     echo "<div style=height:50px;><a name=phpInfo></div>"                          ;
6     phpinfo();
```

# File Source for index.ctp

*Documentation for this file is available at index.ctp*

```
1     <?php
2
3
4
5     $linkString='';
6     foreach ($urlList as $data){
7     $linkString.="     <a href={   $data['url']}>{   $data['text']}</a><p/>            "     ;
8     }
9     $outString="
10    <div style=font-family:sans-serif;margin-bottom:15px;>
11    <div style='margin:15px;border:1pt solid gray;background:#BBBB66;color:883333;font-family:sans-
serif;width:400px;padding-top:20px;text-align:center;'>
12    <div style=font-weight:bold;margin-bottom:5px;>
13    $title</div>
14    <div style=font-size:10pt;>
15    $linkString
16    </div>
17    </div>
18    <div style=font-size:80%;margin-left:15px;>Brought to you
by<BR>      "     .str_replace('tq.org', 'tq.org<BR>'      , __FILE__)."</div>
19    </div>
20    <div style=font-size:80%;color:gray;>"           .SYSTEM_TYPE."</div>
21    "     ;
22
23
24    echo $outString;
```

# File Source for install_notes.ctp

*Documentation for this file is available at install_notes.ctp*

```
1     <?php
2
3     echo "
4     Note: two subdomains for framework utilities:
5     Doctrine.tqwhite.org accesses doctrine sandbox<br/>
6     PhpDoc.tqwhite.org accesses PhpDocumentor web interface<br/>
7
8     "     ;
```

# File Source for send_files_to_host.ctp

*Documentation for this file is available at send_files_to_host.ctp*

```
1     <?php
2     //Display ============================================================================
3
4
5     $outString="
6     <div style=color:red;margin-top:20px;margin-bottom:15px;font-family:sans-serif;font-
size:13pt;>
7     Host FTP Transfer Status
8     </div>
9     <table cellpadding=0 cellspacing=0 border=0 style=width:100%;margin-left:20px;>
10    <TR>
11    <TD style=width:50%>
12    {$transferDirectory->   showArrayProperty($transferDirectory->   processStatusArray)}
13
14    <div style=color:red;margin-top:20px;margin-bottom:10px;font-family:sans-serif;font-
size:10pt;>Upload list</div>
15    {$transferDirectory->   showArrayProperty($transferDirectory->   uploadReportArray)}
16
17    <div style=color:red;margin-top:20px;margin-bottom:10px;font-family:sans-serif;font-
size:10pt;>Delete list</div>
18    {$transferDirectory->   showArrayProperty($transferDirectory->   deleteReportArray)}
19
20    <div style=color:red;margin-top:20px;margin-bottom:10px;font-family:sans-serif;font-
size:10pt;>Error Messages</div>
21    {$transferDirectory->   showArrayProperty($transferDirectory->   errorReportArray)}
22
23    <div style=color:red;margin-top:20px;margin-bottom:10px;font-family:sans-serif;font-
size:10pt;>Exclusion strings</div>
24    {$transferDirectory->   showArrayProperty($transferDirectory->   exclusionStringArray)}
25
26    <div style=color:red;margin-top:20px;margin-bottom:10px;font-family:sans-serif;font-
size:10pt;>Exclusion list</div>
27    {$transferDirectory->   showArrayProperty($transferDirectory->   excludedFilesArray)}
28
29    </TD>
30    <TD style=width:50%>
31
32     
33    </TD>
34    </TR>
35    </TABLE>
36
37    "     ;
38
39    echo $outString;
```

# File Source for index.ctp

*Documentation for this file is available at index.ctp*

```
1     <?php
2
3     $linkString='';
4     foreach ($urlList as $data){
5     $linkString.="     <a href={   $data['url']}>{   $data['text']}</a><p/>            "     ;
6     }
7
8     /*
9     <link rel="stylesheet" href="style-original.css" />
10    <link rel="alternate"
11          type="application/atom+xml"
12          title="My Weblog feed"
13          href="/feed/" />
14    */
15
16    $outString="
17
18    <!doctype html>
19    <html lang='en'>
20    <head>
21    <meta charset='utf-8'>
22
23    <div style=font-family:sans-serif;margin-bottom:15px;>
24    <div style='margin:15px;border:1pt solid gray;background:#BBBB66;color:883333;font-family:sans-
serif;width:400px;padding-top:20px;text-align:center;'>
25    <div style=font-weight:bold;margin-bottom:5px;>
26    $title</div>
27    <div style=font-size:10pt;>
28    $linkString
29    </div>
30    </div>
31    <div style=font-size:80%;margin-left:15px;>Brought to you
by<BR>      "     .str_replace('tq.org', 'tq.org<BR>'      , __FILE__)."</div>
32    </div>
33    "     ;
34
35
36    echo $outString;
```

## Package WhiteBoysFramework

# File Source for base_class.php

*Documentation for this file is available at base_class.php*

```
1     <?php
2     namespace mvc;
3     /**
4      * Base Class for White Boys' Framework
5      *
6      * @author  TQ White II <tq@justkidding.com>
7      * @package WhiteBoysFramework
8      *
9      */
10    class BaseClass{
11
12
13
14    }
```

# File Source for base_controller.php

*Documentation for this file is available at base_controller.php*

```php
<?php
namespace mvc\controllers;
/**
 * Base Controller
 *
 * @author  TQ White II <tq@justkidding.com>
 * @package WhiteBoysFramework
 *
 */
class BaseController extends \mvc\BaseClass{

    protected $entryClassName;

    protected $unknownActionHandlerName='index'; //default can be changed by child class

public function __construct(){
        //this space left intentionally blank, but that may change
    }

public function setUnknownActionHandlerName($value){
        $this->   unknownActionHandlerName=$value;
    }

public function getUnknownActionHandlerName(){
        return $this->   unknownActionHandlerName;
    }

protected function _getTemplatePath($templateName){
        $ds=DS;
        $fullClassName=get_class($this);
        $nameSpace=__NAMESPACE__;

        $simpleClassName=str_replace(__NAMESPACE__.'\\', '', $fullClassName);

        $templateName=str_replace("     $fullClassName::"     , '', $templateName);
        $simpleClassName=preg_replace('/controller$/i', '', $simpleClassName);

        $templatePath=camelCaseToUnderscore("     $simpleClassName$ds$templateName"     );

        return $templatePath;
    }

protected function _getControllerNameForUrl(){
    return str_replace(__NAMESPACE__.'\\', '', $this->   entryClassName);
}

protected function _getUrlPath($templateName=''){
    $controller=$this->   _getControllerNameForUrl();
    if (empty($templateName)){
        $path="     /$controller"     ;
    }
    else{
        $path="     /$controller/$templateName"     ;
    }
    return $path;
}

} //end of class
```

# File Source for open_source.php

*Documentation for this file is available at open_source.php*

```php
<?php
namespace mvc\controllers;
/**
* OpenSourceController contains actions for pages
* accessed by a URL of the form http://domain.com/openSource/action
 *
 * @author  TQ White II <tq@justkidding.com>
 * @package WhiteBoysFramework
 *
 */
class OpenSource extends \mvc\controllers\BaseController{

/**
* Calculates default page info, renders appropriate template
* -presently it's demo pages
*
* @param none
* @return none
* @author TQ White II
*
*/
public function __construct(){
        parent::__construct(); //initialize the basic stuff
        $this->   entryClassName=__CLASS__;
        //get_called_class()
    }

public function index($pageNum=''){

        if (empty($pageNum)){
        $pageNum=1;
        }

        switch ($pageNum){
        default:
        case '1':
            $templateName='index';
            break;
        case '2':
            $templateName='page2';
            break;
        }

        $templatePath=$this->   _getTemplatePath($templateName);
        $page=new \mvc\views\BaseView($templatePath);
        $page->   render();


    }
} //end of class
```

# File Source for site_utilities.php

*Documentation for this file is available at site_utilities.php*

```php
<?php
namespace mvc\controllers;
/**
* SiteUtilities contains actions for pages
* accessed by a URL of the form http://domain.com/siteUtilities/action
 *
 * @author  TQ White II <tq@justkidding.com>
 * @package WhiteBoysFramework
 *
 */
class SiteUtilities extends \mvc\controllers\BaseController{


public function __construct(){
        parent::__construct(); //initialize the basic stuff
        $this->   entryClassName=__CLASS__;

        $this->   unknownActionHandlerName='dispatchSpecial';
    }

/**
* NOT FINISHED
* -a list of links
*
* @param none
* @return none
* @author TQ White II
* @example http://domain.com/controller/
*
*/
public function dispatchSpecial(){
    //this is just a demo
    $args=func_get_args();
    $unknownMethod=array_shift($args); //first element is always the method
    switch ($unknownMethod){
        case '':
            $this->   index(); //allows urls like http://domain.com/siteUtilities
            break;
        default:
            $this->   index();
            break;
    }
}

/**
* Calculates default page info, renders appropriate template
* -a list of links
*
* @param none
* @return none
* @author TQ White II
*
*/
public function index(){$urlList=array();
        $item=array();
        $domain=\configs\appConfig::getServerVar('HTTP_HOST');

        if (SYSTEM_TYPE=='development'){
            $item['url']=$this->   _getUrlPath('sendFilesToHost');
            $item['text']='Send Files to Host';
            $urlList[]=$item;
        }


        $item['url']=$this->   _getUrlPath('environmentInfo');
        $item['text']='Environment info';
        $urlList[]=$item;
```

```
68            $item['url']="errata/phpdoc/HTMLframesConverter/index.html"          ;
69            $item['text']='Docs (web)';
70            $urlList[]=$item;
71
72            $item['url']="errata/phpdoc/PDFdefaultConverter/documentation.pdf"          ;
73            $item['text']='Docs (pdf)';
74            $urlList[]=$item;
75
76
77            $item['url']='/start';
78            $item['text']='Front Page';
79            $urlList[]=$item;
80
81
82
83            $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
84            $page->   SetViewScopedValue('title', "A Title Sent by:
SiteUtilitiesController::index"     );
85            $page->   SetViewScopedValue('urlList', $urlList);
86            $page->   SetViewScopedValue('message', "And more words to live by from the correct
source of content."     );
87            $page->   render();
88        }
89
90    /**
91    * Uploads changed development files to web host using FTP
92    * -based on configs/host_ftp_config.php
93    *
94    * @param none
95    * @return none
96    * @author TQ White II
97    *
98    */
99    public function sendFilesToHost(){
100
101           if (SYSTEM_TYPE!='development'){
102               die("Host FTP Sync can only be used from development systems. This is not one of
those."     );
103               }
104
105           /*
106               make user interface
107               figure out interface to initialize (esp baseUrl)
108               cause web page to display progressively, show files as they are moved
109               make list only version of executeFTP
110
111               make command line compatible
112               allow spec of http transfer file name so can redirect to page of interest after upload
113
114               performance:
115               make sendFile check if mkdir is needed
116               make it so it only logs in once
117           */
118
119           $transferDirectory=new transferDirectory(dirname(ROOT));
120
121           HostFtpConfig::copyIntoTransferDirectoryObject(&    $transferDirectory); //from
configs/host_ftp_config.php
122
123           $transferDirectory->   clearExclusionString();
124           $transferDirectory->   addExclusionString('configs');
125
126           $transferDirectory->   uploadFileTemplate='
127               <div style=margin-bottom:10pt;font-family:sans-serif;font-size:8pt;>
128                   <div style=color:green;>
129                       UPLOAD status=<!result!> <!link!>
130                       </div>
131                   <div style=color:gray;>
132                       source: <!localPath!><br>dest: <!destPath!>
133                       </div>
134                   </div>'      ;
135
136           $transferDirectory->   deleteFileTemplate='
137               <div style=margin-bottom:10pt;font-family:sans-serif;font-size:8pt;>
138                   <div style=color:green;>
139                       DELETE status=<!result!>
140                       </div>
141                   <div style=color:gray;>
142                       source: <!localPath!><br>dest: <!destPath!>
143                       </div>
```

```
144                 </div>'      ;
145
146         $transferDirectory->   dryRunFlag=false; //false says we're in production, upload files
147         $transferDirectory->   initDatabase=false; //true says init db even though
dryRunFlag==true, ignored if dryRunFlag==false
148
149         //execute the FTP
150             $transferDirectory->   analyzeFiles();
151             $transferDirectory->   executeFTP();
152             $transferDirectory->   saveThisObject();
153
154
155             //dump($transferDirectory->deletedFilesArray);
156
157                 $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
158                 $page->   SetViewScopedValue('transferDirectory', $transferDirectory);
159                 $page->   render();
160
161 }
162
163 /**
164  * Displays a variety of system info for developers
165  * -phpInfo(), $_SERVER
166  *
167  * @param none
168  * @return none
169  * @author TQ White II
170  *
171  */
172 public function environmentInfo(){
173         $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
174         $page->   render();
175 }
176
177 } //end of class
```

# File Source for start.php

*Documentation for this file is available at start.php*

```
1     <?php
2     namespace mvc\controllers;
3     /**
4      * SiteUtilities contains actions for pages
5      * accessed by a URL of the form http://domain.com/start/action
6      * or http://domain.com
7      *
8      * @author  TQ White II <tq@justkidding.com>
9      * @package WhiteBoysFramework
10     *
11     */
12    class Start extends \mvc\controllers\BaseController{
13
14    /**
15    * Initialize object
16    * -sets $this->entryClassName
17    *
18    * @param none
19    * @return none
20    * @author TQ White II
21    *
22    */
23    public function __construct(){
24            parent::__construct(); //initialize the basic stuff
25            $this->   entryClassName=__CLASS__;
26        }
27
28
29    /**
30    * Calculates default page info, renders appropriate template
31    * -a list of links
32    *
33    * @param none
34    * @return none
35    * @author TQ White II
36    *
37    */
38    public function index(){
39
40            $urlList=array();
41            $item=array();
42
43            $item['url']='/OpenSource';
44            $item['text']='Open Source Page 1';
45            $urlList[]=$item;
46            $item['url']='/OpenSource/2';
47            $item['text']='Open Source Page 2';
48            $urlList[]=$item;
49            $item['url']='/siteUtilities';
50            $item['text']='Site Utilities';
51            $urlList[]=$item;
52
53
54            $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
55            $page->   SetViewScopedValue('title', "A Title Sent by: startController::index"          );
56            $page->   SetViewScopedValue('urlList', $urlList);
57            $page->   render();
58
59            return;
60        }
61
62    public function fileDemo(){
63
64        //initialization
65            $root=ROOT; //this constant is defined on my system. Probably not yours.
66
$externalImageDirPath=$root.DIRECTORY_SEPARATOR.'framework'.DIRECTORY_SEPARATOR.'tmp'.DIRECTORY_SEPARATOR.
```

```
'external_images';
67            $contents='';
68            $quantity=0;
69            $blockSize=1000;
70            $maxSize=1000000;
71            $sourceUrl='http://justkidding.com/tmp.txt';
72            $sourceUrl='http://justkidding.com/PIX/bush120507.png';
73
74
75        //get source data from web
76            try {
77                $handle=fopen($sourceUrl, 'rb');
78                while (!feof($handle)){
79                    $thisBlock=fread($handle, $blockSize);
80                    $contents.=$thisBlock;
81
82                    $quantity=$quantity+strlen($thisBlock);
83                    if ($quantity>   $maxSize){
84                        throw new Exception("file is too big"          );
85                    }
86                }
87                fclose($handle);
88
89        //write section using demo class instance
90                $urlArray=parse_url($sourceUrl);
91                $fileObj=new fileDemo();
92                $fileObj->   setValue('destDirPath', $externalImageDirPath);
93                $fileObj->   setValue('destFileName', basename($urlArray['path']));
94
95                $fileObj->   setValue('fileContents', $contents);
96                if (!$fileObj->   putData()){
97                    throw new Exception($fileObj->   getValue('errorMessage'));
98                }
99
100               $message="     successfully wrote file: {$fileObj->   getFilePath()}"     ;
101           }
102
103           catch (Exception $e){
104               $message=$e->   getMessage();
105           }
106
107           echo $message;
108
109   /*
110
111       try{
112           $fileObj=new fileDemo();
113           $fileObj->setValue('destDirPath',
ROOT.DIRECTORY_SEPARATOR.'framework'.DIRECTORY_SEPARATOR.'tmp'.DIRECTORY_SEPARATOR.'external_images');
114           $fileObj->setValue('destFileName', 'tmp.txt');
115           if (!$fileObj->getData()){
116               throw new Exception($fileObj->getValue('errorMessage'));
117           }
118
119           $editCopy=$fileObj->getValue('fileContents');
120           $newVersion=$editCopy.PHP_EOL.$editCopy;
121
122           $fileObj->setValue('fileContents', $newVersion);
123           if (!$fileObj->putData()){
124               throw new Exception($fileObj->getValue('errorMessage'));
125           }
126
127           $message="successfully wrote file: {$fileObj->getFilePath()}";
128       }
129
130       catch (Exception $e){
131           $message=$e->getMessage();
132       }
133       */
134
135       $page=new \views\BaseView($this->   _getTemplatePath(__METHOD__));
136       $page->   SetViewScopedValue('message', $message);
137       $page->   render();
138   }
139
140   } //end of class
```

# File Source for base_view.php

*Documentation for this file is available at base_view.php*

```php
<?php
namespace mvc\views;
/**
* Base mechanism for display of pages
* accessed by a URL of the form http://domain.com/openSource/action
*
* @package WhiteBoysFramework
*/
class BaseView extends \mvc\BaseClass{

    private $templatePathName;

    private $viewScopedValuesArray=array();

    private $templateExtension='ctp';

/**
* Generates a view object and, optionally, sets the template
*
* @param $templatePathName (optional)
* @return none
* @author TQ White II
*
*/
public function __construct($templatePathName=''){
        $this->   templatePathName=$templatePathName;
    }
/**
* Set View-scoped Value makes a value available to a display template.
*
* @param $varName: name by which template will refer to this value
* @param $value: value of the template variable
* @return none
* @author TQ White II
*
*/
public function SetViewScopedValue($varName, $value){
        //todo: restrict to declared variables

        $this->   viewScopedValuesArray[$varName]=$value;
    }

private function _getHtmlDISCARD(){
        $templatePathName=$this->   templatePathName;

    }

private function _createScopeAndRun($filePath){

        foreach ($this->   viewScopedValuesArray as $label=>   $data){
            $$label=$data;
        }

        require_once($filePath); //do the actual rendering

    }

public function render(){
        $mvcRoot=MVCROOT;
        $ds=DS;

        $fileString=$this->   templatePathName;

        $filePath="     $mvcRoot/views/$fileString.{$this->   templateExtension}";

        if (file_exists($filePath)){
            $this->   _createScopeAndRun($filePath);
```

```
68                  return;
69              }
70              else{
71              exit("FATAL ERROR: Trying to render missing template. Invalid filePath: $filePath.");
72              }
73          }
74
```

# File Source for base_class.php

*Documentation for this file is available at base_class.php*

```
1     <?php
2     namespace mvc;
3     /**
4      * Base Class for White Boys' Framework
5      *
6      * @author  TQ White II <tq@justkidding.com>
7      * @package WhiteBoysFramework
8      *
9      */
10    class BaseClass{
11
12
13
14    }
```

# File Source for base_controller.php

*Documentation for this file is available at base_controller.php*

```php
<?php
namespace mvc\controllers;
/**
 * Base Controller
 *
 * @author  TQ White II <tq@justkidding.com>
 * @package WhiteBoysFramework
 *
 */
class BaseController extends \mvc\BaseClass{

    protected $entryClassName;

    protected $unknownActionHandlerName='index'; //default can be changed by child class

public function __construct(){
        //this space left intentionally blank, but that may change
    }

public function setUnknownActionHandlerName($value){
        $this->   unknownActionHandlerName=$value;
    }

public function getUnknownActionHandlerName(){
        return $this->   unknownActionHandlerName;
    }

protected function _getTemplatePath($templateName){
        $ds=DS;
        $fullClassName=get_class($this);
        $nameSpace=__NAMESPACE__;

        $simpleClassName=str_replace(__NAMESPACE__.'\\', '', $fullClassName);

        $templateName=str_replace("     $fullClassName::"     , '', $templateName);
        $simpleClassName=preg_replace('/controller$/i', '', $simpleClassName);

        $templatePath=camelCaseToUnderscore("     $simpleClassName$ds$templateName"     );

        return $templatePath;
    }

protected function _getControllerNameForUrl(){
    return str_replace(__NAMESPACE__.'\\', '', $this->   entryClassName);
}

protected function _getUrlPath($templateName=''){
    $controller=$this->   _getControllerNameForUrl();
    if (empty($templateName)){
        $path="     /$controller"     ;
    }
    else{
        $path="     /$controller/$templateName"     ;
    }
    return $path;
}

} //end of class
```

# File Source for base_view.php

*Documentation for this file is available at base_view.php*

```php
<?php
namespace mvc\views;
/**
* Base mechanism for display of pages
* accessed by a URL of the form http://domain.com/openSource/action
*
* @package WhiteBoysFramework
*/
class BaseView extends \mvc\BaseClass{

    private $templatePathName;

    private $viewScopedValuesArray=array();

    private $templateExtension='ctp';

/**
* Generates a view object and, optionally, sets the template
*
* @param $templatePathName (optional)
* @return none
* @author TQ White II
*
*/
public function __construct($templatePathName=''){
        $this->   templatePathName=$templatePathName;
    }
/**
* Set View-scoped Value makes a value available to a display template.
*
* @param $varName: name by which template will refer to this value
* @param $value: value of the template variable
* @return none
* @author TQ White II
*
*/
public function SetViewScopedValue($varName, $value){
        //todo: restrict to declared variables

        $this->   viewScopedValuesArray[$varName]=$value;
    }

private function _getHtmlDISCARD(){
        $templatePathName=$this->   templatePathName;

    }

private function _createScopeAndRun($filePath){

        foreach ($this->   viewScopedValuesArray as $label=>   $data){
            $$label=$data;
        }

        require_once($filePath); //do the actual rendering

    }

public function render(){
        $mvcRoot=MVCROOT;
        $ds=DS;

        $fileString=$this->   templatePathName;

        $filePath="     $mvcRoot/views/$fileString.{$this->   templateExtension}";

        if (file_exists($filePath)){
            $this->   _createScopeAndRun($filePath);
```

```
68                  return;
69              }
70              else{
71              exit("FATAL ERROR: Trying to render missing template. Invalid filePath: $filePath.");
72              }
73          }
74
```

# File Source for open_source.php

*Documentation for this file is available at open_source.php*

```php
<?php
namespace mvc\controllers;
/**
* OpenSourceController contains actions for pages
* accessed by a URL of the form http://domain.com/openSource/action
 *
 * @author  TQ White II <tq@justkidding.com>
 * @package WhiteBoysFramework
 *
 */
class OpenSource extends \mvc\controllers\BaseController{

/**
* Calculates default page info, renders appropriate template
* -presently it's demo pages
*
* @param none
* @return none
* @author TQ White II
*
*/
public function __construct(){
        parent::__construct(); //initialize the basic stuff
        $this->   entryClassName=__CLASS__;
        //get_called_class()
    }

public function index($pageNum=''){

        if (empty($pageNum)){
        $pageNum=1;
        }

        switch ($pageNum){
        default:
        case '1':
            $templateName='index';
            break;
        case '2':
            $templateName='page2';
            break;
        }

        $templatePath=$this->   _getTemplatePath($templateName);
        $page=new \mvc\views\BaseView($templatePath);
        $page->   render();


    }
} //end of class
```

# File Source for site_utilities.php

*Documentation for this file is available at site_utilities.php*

```php
<?php
namespace mvc\controllers;
/**
* SiteUtilities contains actions for pages
* accessed by a URL of the form http://domain.com/siteUtilities/action
 *
 * @author  TQ White II <tq@justkidding.com>
 * @package WhiteBoysFramework
 *
 */
class SiteUtilities extends \mvc\controllers\BaseController{


public function __construct(){
        parent::__construct(); //initialize the basic stuff
        $this->   entryClassName=__CLASS__;

        $this->   unknownActionHandlerName='dispatchSpecial';
    }

/**
* NOT FINISHED
* -a list of links
*
* @param none
* @return none
* @author TQ White II
* @example http://domain.com/controller/
*
*/
public function dispatchSpecial(){
    //this is just a demo
    $args=func_get_args();
    $unknownMethod=array_shift($args); //first element is always the method
    switch ($unknownMethod){
        case '':
            $this->   index(); //allows urls like http://domain.com/siteUtilities
            break;
        default:
            $this->   index();
            break;
    }
}

/**
* Calculates default page info, renders appropriate template
* -a list of links
*
* @param none
* @return none
* @author TQ White II
*
*/
public function index(){$urlList=array();
        $item=array();
        $domain=\configs\appConfig::getServerVar('HTTP_HOST');

        if (SYSTEM_TYPE=='development'){
            $item['url']=$this->   _getUrlPath('sendFilesToHost');
            $item['text']='Send Files to Host';
            $urlList[]=$item;
        }


        $item['url']=$this->   _getUrlPath('environmentInfo');
        $item['text']='Environment info';
        $urlList[]=$item;
```

```
68            $item['url']="errata/phpdoc/HTMLframesConverter/index.html"          ;
69            $item['text']='Docs (web)';
70            $urlList[]=$item;
71
72            $item['url']="errata/phpdoc/PDFdefaultConverter/documentation.pdf"          ;
73            $item['text']='Docs (pdf)';
74            $urlList[]=$item;
75
76
77            $item['url']='/start';
78            $item['text']='Front Page';
79            $urlList[]=$item;
80
81
82
83            $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
84            $page->   SetViewScopedValue('title', "A Title Sent by:
SiteUtilitiesController::index"     );
85            $page->   SetViewScopedValue('urlList', $urlList);
86            $page->   SetViewScopedValue('message', "And more words to live by from the correct
source of content."     );
87            $page->   render();
88        }
89
90    /**
91    * Uploads changed development files to web host using FTP
92    * -based on configs/host_ftp_config.php
93    *
94    * @param none
95    * @return none
96    * @author TQ White II
97    *
98    */
99    public function sendFilesToHost(){
100
101           if (SYSTEM_TYPE!='development'){
102               die("Host FTP Sync can only be used from development systems. This is not one of
those."     );
103               }
104
105           /*
106               make user interface
107               figure out interface to initialize (esp baseUrl)
108               cause web page to display progressively, show files as they are moved
109               make list only version of executeFTP
110
111               make command line compatible
112               allow spec of http transfer file name so can redirect to page of interest after upload
113
114               performance:
115               make sendFile check if mkdir is needed
116               make it so it only logs in once
117           */
118
119           $transferDirectory=new transferDirectory(dirname(ROOT));
120
121           HostFtpConfig::copyIntoTransferDirectoryObject(&    $transferDirectory); //from
configs/host_ftp_config.php
122
123           $transferDirectory->   clearExclusionString();
124           $transferDirectory->   addExclusionString('configs');
125
126           $transferDirectory->   uploadFileTemplate='
127               <div style=margin-bottom:10pt;font-family:sans-serif;font-size:8pt;>
128                   <div style=color:green;>
129                       UPLOAD status=<!result!> <!link!>
130                       </div>
131                   <div style=color:gray;>
132                       source: <!localPath!><br>dest: <!destPath!>
133                       </div>
134                   </div>'      ;
135
136           $transferDirectory->   deleteFileTemplate='
137               <div style=margin-bottom:10pt;font-family:sans-serif;font-size:8pt;>
138                   <div style=color:green;>
139                       DELETE status=<!result!>
140                       </div>
141                   <div style=color:gray;>
142                       source: <!localPath!><br>dest: <!destPath!>
143                       </div>
```

```
144                 </div>'      ;
145
146         $transferDirectory->   dryRunFlag=false; //false says we're in production, upload files
147         $transferDirectory->   initDatabase=false; //true says init db even though
dryRunFlag==true, ignored if dryRunFlag==false
148
149         //execute the FTP
150             $transferDirectory->   analyzeFiles();
151             $transferDirectory->   executeFTP();
152             $transferDirectory->   saveThisObject();
153
154
155             //dump($transferDirectory->deletedFilesArray);
156
157                 $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
158                 $page->   SetViewScopedValue('transferDirectory', $transferDirectory);
159                 $page->   render();
160
161 }
162
163 /**
164 * Displays a variety of system info for developers
165 * -phpInfo(), $_SERVER
166 *
167 * @param none
168 * @return none
169 * @author TQ White II
170 *
171 */
172 public function environmentInfo(){
173         $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
174         $page->   render();
175 }
176
177 } //end of class
```

# File Source for start.php

*Documentation for this file is available at start.php*

```
1     <?php
2     namespace mvc\controllers;
3     /**
4      * SiteUtilities contains actions for pages
5      * accessed by a URL of the form http://domain.com/start/action
6      * or http://domain.com
7      *
8      * @author  TQ White II <tq@justkidding.com>
9      * @package WhiteBoysFramework
10     *
11     */
12    class Start extends \mvc\controllers\BaseController{
13
14    /**
15    * Initialize object
16    * -sets $this->entryClassName
17    *
18    * @param none
19    * @return none
20    * @author TQ White II
21    *
22    */
23    public function __construct(){
24            parent::__construct(); //initialize the basic stuff
25            $this->   entryClassName=__CLASS__;
26        }
27
28
29    /**
30    * Calculates default page info, renders appropriate template
31    * -a list of links
32    *
33    * @param none
34    * @return none
35    * @author TQ White II
36    *
37    */
38    public function index(){
39
40            $urlList=array();
41            $item=array();
42
43            $item['url']='/OpenSource';
44            $item['text']='Open Source Page 1';
45            $urlList[]=$item;
46            $item['url']='/OpenSource/2';
47            $item['text']='Open Source Page 2';
48            $urlList[]=$item;
49            $item['url']='/siteUtilities';
50            $item['text']='Site Utilities';
51            $urlList[]=$item;
52
53
54            $page=new \mvc\views\BaseView($this->   _getTemplatePath(__METHOD__));
55            $page->   SetViewScopedValue('title', "A Title Sent by: startController::index"          );
56            $page->   SetViewScopedValue('urlList', $urlList);
57            $page->   render();
58
59            return;
60        }
61
62    public function fileDemo(){
63
64        //initialization
65            $root=ROOT; //this constant is defined on my system. Probably not yours.
66
$externalImageDirPath=$root.DIRECTORY_SEPARATOR.'framework'.DIRECTORY_SEPARATOR.'tmp'.DIRECTORY_SEPARATOR.
```

```
'external_images';
67            $contents='';
68            $quantity=0;
69            $blockSize=1000;
70            $maxSize=1000000;
71            $sourceUrl='http://justkidding.com/tmp.txt';
72            $sourceUrl='http://justkidding.com/PIX/bush120507.png';
73
74
75        //get source data from web
76            try {
77                $handle=fopen($sourceUrl, 'rb');
78                while (!feof($handle)){
79                    $thisBlock=fread($handle, $blockSize);
80                    $contents.=$thisBlock;
81
82                    $quantity=$quantity+strlen($thisBlock);
83                    if ($quantity>   $maxSize){
84                        throw new Exception("file is too big"          );
85                    }
86                }
87                fclose($handle);
88
89        //write section using demo class instance
90                $urlArray=parse_url($sourceUrl);
91                $fileObj=new fileDemo();
92                $fileObj->   setValue('destDirPath', $externalImageDirPath);
93                $fileObj->   setValue('destFileName', basename($urlArray['path']));
94
95                $fileObj->   setValue('fileContents', $contents);
96                if (!$fileObj->   putData()){
97                    throw new Exception($fileObj->   getValue('errorMessage'));
98                }
99
100               $message="     successfully wrote file: {$fileObj->   getFilePath()}"     ;
101           }
102
103           catch (Exception $e){
104               $message=$e->   getMessage();
105           }
106
107           echo $message;
108
109   /*
110
111       try{
112           $fileObj=new fileDemo();
113           $fileObj->setValue('destDirPath',
ROOT.DIRECTORY_SEPARATOR.'framework'.DIRECTORY_SEPARATOR.'tmp'.DIRECTORY_SEPARATOR.'external_images');
114           $fileObj->setValue('destFileName', 'tmp.txt');
115           if (!$fileObj->getData()){
116               throw new Exception($fileObj->getValue('errorMessage'));
117           }
118
119           $editCopy=$fileObj->getValue('fileContents');
120           $newVersion=$editCopy.PHP_EOL.$editCopy;
121
122           $fileObj->setValue('fileContents', $newVersion);
123           if (!$fileObj->putData()){
124               throw new Exception($fileObj->getValue('errorMessage'));
125           }
126
127           $message="successfully wrote file: {$fileObj->getFilePath()}";
128       }
129
130       catch (Exception $e){
131           $message=$e->getMessage();
132       }
133       */
134
135       $page=new \views\BaseView($this->   _getTemplatePath(__METHOD__));
136       $page->   SetViewScopedValue('message', $message);
137       $page->   render();
138   }
139
140   } //end of class
```

# Index

## A



## B

## C



## D



## E



## F

## G



## H



## I

## O



## P



## Q



## R

## S



## T

## _