PHPRum Documentation

Contents

index.php

Startup script This script creates an instance of the application and executes

This should be the only PHP script directly accessable via the web Do not modify this script!

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

include **'../system/base/rum.php'**[*line* 19]

load framework and application

appconfiguration.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

applicationbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

appmessage.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

appmessagetype.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

appstate.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

argumentoutofrangeexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

badmembercallexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

badmethodcallexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

classloader.inc.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

string function class_search($className, [$base = '/']) *[line 78]*
Function Parameters:

- *string* **$className** name of class
- **$base**

class_search

Returns path to class folder if found

debug.inc.php

debug tools

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

void function alloc() *[line 112]*

Print memory usage

void function breakpoint() *[line 26]*

Set breakpoint

string function callstack() *[line 36]*

Return callstack string as html

void function clr() *[line 17]*

Clear all output buffers

void function dmp($var, [$halt = true]) *[line 161]*

Function Parameters:

- *mixed* **$var** Variable to print
- *bool* **$halt** Specifies whether to halt application

Print variable to screen

void function stp() *[line 144]*

Stop timer and print results

void function strt() *[line 133]*

Begin timer

void function trace($string, $var) *[line 175]*

Function Parameters:

- *mixed* **$var** Variable to trace
- **$string**

Trace

void function trap() *[line 103]*

begin trapping memory

eventbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

eventhandlerbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

exceptionbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

indexoutofrangeexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

invalidargumentexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

invalidoperationexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

methodnotimplementedexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

missingargumentexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

modelbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

rum.php

Startup script

This script will detect the following environmental variables

__ROOT__ absolute path to the root folder __HTDOCS_PATH__ absolute path to the web folder __APP_URI__ root relative URI to the web folder __PROTOCOL__ protocol prefix (http|https):// __HOST__ host name __PORT__ server port

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

__ACTIVERECORD_AUTO_MAP__ = FALSE *[line 263]*

specifies the number of warnings to dump

__APP_CONF_FILENAME__ = '/application.xml' *[line 153]*

specifies the environment parameter

__APP_PATH__ = __ROOT__.'/app' *[line 283]*

specifies the root relative path to the application folder

__APP_URI__ = str_replace(" ","%20",substr($_SERVER['SCRIPT_NAME'],0,strrpos($_SERVER['SCRIPT_NAME'],'/'))) *[line 118]*

specifies the application uri

__ASSETS_URI__ = __APP_URI__.'/assets' *[line 123]*

specifies the URI to the assets folder

__ASYNC_REQUEST_PARAMETER__ = 'async' *[line 183]*

specifies the async request parameter

__BUILD_PATH__ = __ROOT__.'/.build' *[line 273]*

specifies the root relative path to the cache folder

__CACHE_PATH__ = __ROOT__.'/.cache' *[line 278]*

specifies the root relative path to the cache folder

__CACHE_TABLENAME__ = 'cache' *[line 138]*

specifies the cache table name

__CLASS_EXTENSION__ = '.class.php' *[line 208]*

specifies the class extension

__CONFIG_PATH__ = __ROOT__.'/app/config' *[line 288]*

specifies the root relative path to the configuration folder

__CONTROLLERS_NAMESPACE__ = 'Controllers' *[line 168]*

specifies the models namespace

__CONTROLLERS_PATH__ = __ROOT__.'/app/controllers' *[line 313]*

specifies the default controller path

__CONTROLLER_EXTENSION__ = '.php' *[line 213]*

specifies the controller extension

__CONTROLLER_TESTCASE_SUFFIX__ = 'ControllerTestCase' *[line 233]*

specifies the controller testcase suffix

__DB_SCHEMA_VERSION_TABLENAME__ = 'db_schema_version' *[line 133]*

specifies the db_schema_version table name

__DEFAULT_TIMEZONE__ = 'Canada/Mountain' *[line 27]*

specifies the default timezone

__DEPLOYMENT_EXTENSION__ = '.php' *[line 223]*

specifies the controller extension

__DEPLOY_PATH__ = __CONFIG_PATH__.'/deploy' *[line 298]*

specifies the root relative path to the deployment folder

__DEV_ENV__ = 'dev' *[line 188]*

specifies the dev environment name

__ENV_CONF_FILENAME__ = '/environment.php' *[line 158]*

specifies the environment parameter

__ENV_PARAMETER__ = 'APP_ENV' *[line 163]*

specifies the environment parameter

__ENV_PATH__ = __CONFIG_PATH__.'/environments' *[line 293]*

specifies the root relative path to the configuration folder

__ERROR_LIMIT__ = 10 *[line 243]*

specifies the number of warnings to dump

__FIXTURES_PATH__ = __ROOT__.'/app/tests/fixtures' *[line 333]*

specifies the default fixtures path

__FUNCTIONAL_TESTS_PATH__ = __ROOT__.'/app/tests/functional' *[line 338]*

specifies the default functional test path

__HOST__ = isset($_SERVER['HTTP_HOST'])?$_SERVER['HTTP_HOST']:'' *[line 113]*

specifies the host name

__HTDOCS_PATH__ = substr($_SERVER['SCRIPT_FILENAME'],0,strrpos(str_replace('\\','/',$_SERVER['SCRIPT_FILENAME']),'/')) *[line 51]*

specifies the public web folder path

__LANGS_FILE__ = __CONFIG_PATH__.'/langs.xml' *[line 303]*

specifies the path to the langs file

__LANGS_TABLENAME__ = 'langs' *[line 148]*

specifies the langs table name

__LIB_PATH__ = __SYSTEM_PATH__.'/libs' *[line 363]*

specifies the root relative path to the libs folder

__LOGS_TABLENAME__ = 'logs' *[line 143]*

specifies the logs table name

__LOG_PATH__ = __ROOT__.'/logs' *[line 353]*

specifies the root relative path to the log folder

__MIGRATIONS_PATH__ = __CONFIG_PATH__.'/migrations' *[line 308]*

specifies the root relative path to the migrations folder

__MODELS_NAMESPACE__ = 'Models' *[line 173]*

specifies the models namespace

__MODELS_PATH__ = __ROOT__.'/app/models' *[line 318]*

specifies the default model path

__NAME__ = isset($_SERVER['SERVER_NAME'])?$_SERVER['SERVER_NAME']:'' *[line 103]*

specifies the server name

__PAGE_EXTENSION__ = '/' *[line 203]*

specifies the request page extension

__PAGE_REQUEST_PARAMETER__ = 'page' *[line 178]*

specifies the page request parameter

__PLUGINS_PATH__ = __ROOT__.'/app/plugins' *[line 323]*

specifies the root relative path to the plugins folder

__PORT__ = isset($_SERVER['SERVER_PORT'])?$_SERVER['SERVER_PORT']:'' *[line 108]*

specifies the server port

__PROTOCOL__ = $protocol *[line 98]*

specifies the transfer protocol

__QUOTE_STYLE__ = ENT_COMPAT *[line 248]*

specifies the quote style when escaping

__ROOT__ = $current_dir *[line 64]*
__SSL_PORT__ = 443 *[line 198]*

specifies the SSL port

__SYSTEM_PATH__ = __ROOT__.'/system' *[line 358]*
specifies the root relative path to the system folder

__TEMPLATES_PATH__ = __ROOT__.'/app/layouts' *[line 328]*
specifies the default template path

__TEMPLATE_EXTENSION__ = '.tpl' *[line 218]*
specifies the template extension

__TESTCASE_SUFFIX__ = 'TestCase' *[line 228]*
specifies the testcase suffix

__TEST_ENV__ = 'test' *[line 193]*
specifies the test environment name

__UNIT_TESTS_PATH__ = __ROOT__.'/app/tests/unit' *[line 343]*
specifies the default unit test path

__VALIDATION_TIMEOUT__ = '3000' *[line 238]*
specifies the asyncronous validation timeout

__VIEWS_PATH__ = __ROOT__.'/app/views' *[line 348]*
specifies the default view path

require **__SYSTEM_PATH__.'/base/build.class.php'** *[line 398]*

require **__SYSTEM_PATH__.'/rum.class.php'** *[line 399]*

require **__SYSTEM_PATH__.'/base/framework.info.file'** *[line 396]*

require **__SYSTEM_PATH__.'/base/classloader.inc.php'** *[line 397]*

typemismatchexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

consoleapplicationbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

loggerbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

make.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

rum.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

tester.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewchangeorderevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewchangeordereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewfilterevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewfiltereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewsortevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewsorteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class AppConfiguration

[line 73]

Provides access to the Application Configuration

- **Package** PHPRum
- **Property** AppState $state: specifies app state
- **Property** string $lang: specifies default language
- **Property** string $charset: specifies default character set
- **Property** AppSettingsCollection $appsettings: collection of app settings
- **Property** string $defaultTheme: specifies the default theme
- **Property** bool $viewStateEnabled: specifies whether viewstate is enabled
- **Property** string $viewStateMethod: specifies the viewstate method
- **Property** int $viewStateExpires: specifies the viewstate expiration in seconds form the current time
- **Property** string $themes: path to themes folder
- **Property** string $defaultController: specifies default controller
- **Property** string $requestParameter: specifies page request parameter
- **Property** bool $rewriteURIS: specifies whether to rewrite URI's

- **Property** bool $cookielessSession: specifies whether cookieless session is enabled
- **Property** int $sessionTimeout: specifies the session timeout
- **Property** bool $cacheEnabled: specifies if cache is enabled
- **Property** int $cacheExpires: specifies cache expiration in seconds
- **Property** string $authenticationMethod: specifies authentication method
- **Property** string $authenticationDeny: specifies pages to deny
- **Property** string $authenticationAllow: specifies pages to allow
- **Property** string $authenticationRestrict: specifies allowed ip address
- **Property** int $authenticationMaxInvalidAttempts: specifies max allowed invalid attempts
- **Property** int $authenticationAttemtpWindow: specifies attempt window in seconds
- **Property** bool $authenticationRequireSSL: specifies whether to require SSL for authentication
- **Property** string $authenticationBasicRealm: specifies realm for basic authentication
- **Property** string $authenticationFormsLoginPage: specifies authentication login page
- **Property** string $authenticationFormsCookieName: specifies session cookie name
- **Property** string $authenticationFormsSecret: specifies secret used when checking auth cookies
- **Property** array $authenticationCredentialsUsers: array of authentication users
- **Property** array $authenticationCredentialsTables: array of authentication tables
- **Property** array $authenticationMemberships: array of memberships
- **Property** array $authenticationMembershipsTables: array of membership tables
- **Property** array $authorizationDeny: specifies the roles that are denied access by default
- **Property** array $authorizationAllow: specifies the roles that are granted access by default
- **Property** bool $authorizationRequireSSL: specifies whether to require SSL for authentication
- **Property** array $authorizationPages: contains the page authorization configuration
- **Property** string $dsn: dsn connecting string
- **Property** string $test_dsn: testcase dsn connectin string
- **Property** array $errors: array of errors
- **Property** string $root: path to root folder
- **Property** string $htdocs: path to htdocs folder
- **Property** string $protocol: server protocol

- **Property** string $host: server host
- **Property** string $uri: base application URI
- **Property** string $url: base application URL
- **Property** string $assets: base URI to the assets folder
- **Property** string $controllers: path to controllers folder
- **Property** string $views: path to views folder
- **Property** string $templates: path to templates folder
- **Property** string $functionaltests: path to funcitonal test folder
- **Property** string $unittests: path to unit test folder
- **Property** string $fixtures: path to fixtures folder
- **Property** string $system: path to system folder
- **Property** string $cache: path to cache folder
- **Property** string $tmp: path to tmp folder
- **Property** string $logs: path to logs folder
- **Author** Darnell Shinbine

Constructor *void* function AppConfiguration::__construct() *[line 486]*

Creates an instance of the controller and sets default action map.

- **Access** public

void function AppConfiguration::loadAppConfig($file) *[line 500]*

Function Parameters:

- *string* **$file** path to config file

loads config data from an xml file

- **Access** public

Class ApplicationBase

[line 32]

This class represents the base application. This class recieves command line input and delegates responcibilities to sub components.

- **Package** PHPRum
- **Property** string $applicationId: Specifies the application Id
- **Property** string $namespace: Specifies the application namespace
- **Property** AppConfiguration $config: Reference to the AppConfiguration
- **Property** CacheBase $cache: Reference to the Cache object
- **Property** LoggerBase $logger: Reference to the Logger object
- **Property** TranslatorBase $translator: Reference to the Translator object
- **Property** IMailClient $mailClient: Reference to the Mail client object
- **Property** string $lang: Specifies the language
- **Property** string $charset: Specifies the character set
- **Property** DataAdapter $dataAdapter: Reference to the DataAdapter
- **Property** bool $debug: Specifies whether the application is in debug mode
- **Property** Timer $timer: Reference to the app Timer
- **Property** EventCollection $events: event collection
- **Author** Darnell Shinbine
- **Abstract Element**

ApplicationBase::$events

EventCollection = null *[line 110]*

event collection

- **Access** protected

Constructor *void* function ApplicationBase::__construct() *[line 126]*

Constructor

Creates an instance of the controller and sets default action map.

- **Access** public

void function ApplicationBase::execute() *[line 419]*

execute the application

- **Abstract Element**
- **Access** protected

CacheBase function ApplicationBase::getCache() *[line 450]*

returns the common cache object (overrided this method in the application class)

- **Access** protected

ApplicationBase function ApplicationBase::getInstance([$initialize = null]) *[line 358]*

Function Parameters:

- *ApplicationBase* **$initialize** new instance of an ApplicationBase object

Returns a static instance of the app server. This provides a way of accessing app variables from outside the class.
must be initialized by $controller = ApplicationBase::getinstance($servletObject);

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

LoggerBase function ApplicationBase::getLogger() *[line 461]*
returns the common logger object (overrided this method in the application class)

- **Access** protected

IMailClient function ApplicationBase::getMailClient() *[line 483]*
returns the common mail client object (overrided this method in the application class)

- **Access** protected

void function ApplicationBase::getPluginHooks() *[line 291]*
get plugin hooks

- **Access** public

TranslatorBase function ApplicationBase::getTranslator() *[line 472]*

returns the common translator object (overrided this method in the application class)

- **Access** protected

void function ApplicationBase::handleError($errno, $errstr, $errfile, $errline) *[line 442]*

Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description
- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Abstract Element**
- **Access** protected

void function ApplicationBase::handleException(\$e, $e) *[line 429]*

Function Parameters:

- *\Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden

to provide error handling.

- **Abstract Element**
- **Access** protected

void function ApplicationBase::loadAppConfig($xmlConfig, $xmlconfig) *[line 327]*

Function Parameters:

- *string* **$xmlconfig** path to xml config file
- **$xmlConfig**

loads config data from an xml file

- **Access** public

void function ApplicationBase::run() *[line 221]*

begin processing of the servlet

- **Access** public

Class AppMessage

[line 17]

Represents a single application message

- **Package** PHPRum
- **Author** Darnell Shinbine

Constructor *void* function AppMessage::__construct($msg, [$type = null]) *[line 59]*
Function Parameters:

- *string* **$msg** Message content
- *AppMessageType* **$type** Message type as constant of AppMessageType::Success(), AppMessageType::Fail() or AppMessageType::Notice()

Constructor

- **Access** public

Class AppMessageType

[line 17]

Represents an application message type

- **Package** PHPRum
- **Author** Darnell Shinbine

AppMessageType function AppMessageType::Fail() *[line 42]*

Specifies the message type as Fail

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

AppMessageType function AppMessageType::Info() *[line 30]*

Specifies the message type as Notice

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

AppMessageType function AppMessageType::Success() *[line 36]*

Specifies the message type as Success

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

AppMessageType function AppMessageType::Warning() *[line 48]*

Specifies the message type as Notice

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

string function AppMessageType::__toString() *[line 55]*

Returns a string representing the type

- **Access** public

Class AppState

[line 17]

Represents an application state

- **Package** PHPRum
- **Author** Darnell Shinbine

AppState function AppState::Debug() *[line 36]*

Specifies the application state as Debug

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

AppState function AppState::On() *[line 30]*

Specifies the application state as On

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

Class ArgumentOutOfRangeException

[line 17]

Represents Argument Out Of Range Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class BadMemberCallException

[line 17]

Represents a Bad Member Call Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class BadMethodCallException

[line 17]

Represents a Bad Method Call Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class ConsoleApplicationBase

[line 19]

Provides the base funcionality of a console application. This class recieves command line data and executes the application

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Abstract Element**

Class EventBase

[line 19]

Provides basic event handling

- **Package** PHPRum
- **Property** string $name: event name
- **Author** Darnell Shinbine

- **Abstract Element**

Constructor *void* function EventBase::__construct($name) *[line 34]*

Function Parameters:

- *string* **$name** event name

Constructor

- **Access** protected

Class EventHandlerBase

[line 20]

Provides basic event handling

- **Package** PHPRum
- **Property** string $event: event
- **Property** string $callback: callback
- **Author** Darnell Shinbine
- **Abstract Element**

Constructor *void* function EventHandlerBase::__construct($event, $callback) *[line 43]*

Function Parameters:

- *string* **$event** event

- *string* **$callback** call back

Constructor

- **Access** protected

void function EventHandlerBase::raise(&$sender, [$args = array()], $sender) *[line 79]*
Function Parameters:

- *object* **$sender** Sender object
- *array* **$args** optional args
- **&$sender**

raise event

- **Access** public

Class ExceptionBase

[line 18]

Represents a base Exception

- **Package** PHPRum

- **Author** Darnell Shinbine
- **Abstract Element**

Class GridViewChangeOrderEvent

[line 18]

Provides event handling

- **Package** PHPRum
- **Author** Darnell Shinbine

Constructor *void* function GridViewChangeOrderEvent::__construct() *[line 25]*

Constructor

- **Access** public

Class GridViewChangeOrderEventHandler

[line 18]

Provides event handling

- **Package** PHPRum

- **Author** Darnell Shinbine

Constructor *void* function GridViewChangeOrderEventHandler::__construct($callback) *[line 26]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class GridViewFilterEvent

[line 18]

Provides event handling

- **Package** PHPRum
- **Author** Darnell Shinbine

Constructor *void* function GridViewFilterEvent::__construct() *[line 25]*

Constructor

- **Access** public

Class GridViewFilterEventHandler

[line 18]

Provides event handling

- **Package** PHPRum
- **Author** Darnell Shinbine

Constructor *void* function GridViewFilterEventHandler::__construct($callback) *[line 26]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class GridViewSortEvent

[line 18]

Provides event handling

- **Package** PHPRum
- **Author** Darnell Shinbine

Constructor *void* function GridViewSortEvent::__construct() *[line 25]*

Constructor

- **Access** public

Class GridViewSortEventHandler

[line 18]

Provides event handling

- **Package** PHPRum
- **Author** Darnell Shinbine

Constructor *void* function GridViewSortEventHandler::__construct($callback) *[line 26]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class IndexOutOfRangeException

[line 17]

Represents an Index Out Of Range Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class InvalidArgumentException

[line 17]

Represents an Invalid Argument Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class InvalidOperationException

[line 17]

Represents an Invalid Operation Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class LoggerBase

[line 17]

Provides access to read/write log messages

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Abstract Element**

void function LoggerBase::flush($category) *[line 44]*

Function Parameters:

- *string* **$category** message category

This method flushes all messages

- **Abstract Element**
- **Access** public

void function LoggerBase::log($message, $category) *[line 26]*

Function Parameters:

- *string* **$message** message to log
- *string* **$category** message category

This method writes a log message to memory

- **Abstract Element**
- **Access** public

array function LoggerBase::logs($category) *[line 35]*
Function Parameters:

- *string* **$category** message category

This method retrieves all log messages

- **Abstract Element**
- **Access** public

Class Make

[line 18]

deplomyent script

- **Package** PHPRum
- **Author** Darnell Shinbine

Make::$namespace

string = '' *[line 24]*

namespace

- **Static**

void function Make::execute($argc, $argv) *[line 34]*
Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

execute the application

- **Access** protected

array function Make::getOptions($argc, $argv) *[line 87]*
Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

retrieve command line options

- **Access** protected

void function Make::handleError($errno, $errstr, $errfile, $errline) *[line 134]*
Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description
- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Access** protected

void function Make::handleException(\$e, $e) *[line 121]*
Function Parameters:

- *\Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden to provide error handling.

- **Access** protected

Class MethodNotImplementedException

[line 17]

Represents a Method Not Implemented Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

MethodNotImplementedException::$message

mixed = 'Method Not Implemented' *[line 19]*

- **Access** protected

Class MissingArgumentException

[line 17]

Represents a Missing Arguments Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class ModelBase

[line 17]

This class represents a data model.

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Abstract Element**

ActiveRecordBase function ModelBase::getClass($args) *[line 60]*

Function Parameters:

- *array* **$args** optional associative array of initial properties

static method to create new ActiveRecordBase of this type

- **Static**
- **Access** protected

array function ModelBase::toArray() *[line 51]*

converts the model into an array

- **Abstract Element**
- **Access** public

Class Rum

[line 20]

Provides access to application services

It contains such things as the session data, configuration data, server messages and application state. It also provides methods for forwarding, message handling, URL rewriting and logging.

- **Package** PHPRum
- **Author** Darnell Shinbine

WebApplicationBase function Rum::app() *[line 27]*

This method returns the application

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

string function Rum::baseURI() *[line 38]*

returns the application base uri

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

string function Rum::breadcrumb() *[line 49]*

returns the current page breadcrumb trail as an HTML element

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

CacheBase function Rum::cache() *[line 92]*

This method returns the configuration object

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

void function Rum::clearMessages() *[line 188]*

this method removes any displayed messages

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

AppConfiguration function Rum::config() *[line 103]*

This method returns the configuration object

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

DataAdapter function Rum::db() *[line 114]*

This method returns the application DataAdapter object

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

string function Rum::escape($string, [$quote_style = __QUOTE_STYLE__], [$charset = '']) *[line 128]*

Function Parameters:

- *string* **$string** string to escape
- *int* **$quote_style** quote style
- *string* **$charset** character set

This method returns an escaped string using the application charset for outputting

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

void function Rum::flash($msg, [\\\$type = null], $type) *[line 157]*

Function Parameters:

- *string* **$msg** Message content
- *AppMessageType* **$type** Message type as constant of AppMessageType::Success(), AppMessageType::Fail() or AppMessageType::Notice()
- *AppMessageType* **\\\$type**

this method adds message to message stack

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

void function Rum::forward([$nextPage = ''], [$args = array()], [\\\$method = null], $method) *[line 144]*

Function Parameters:

- *string* **$nextPage** Name of requested page
- *array* **$args** args for next action
- *ForwardMethodType* **$method** forward method as constant of ForwardMethodType::URI() or ForwardMethodType::Request()
- *ForwardMethodType* **\\\$method**

This method sets the next URI once the current controller is finished executing, the servlet will re-request the page with the requested action. This method allows you to replace post data with keep friendly urls that can be bookmarked.

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

string function Rum::link($title, $page, [$args = array()], [$class = '']) *[line 205]*

Function Parameters:

- *string* **$title** title of link
- *string* **$page** name of page
- *array* **$args** array of args
- *string* **$class** css class

returns an HTML anchor element

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

void function Rum::log($message, [$category = 'events']) *[line 220]*

Function Parameters:

- *string* **$message** event to log
- *string* **$category** log category

This method automatically writes a string to a log stamped with the current time

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

string function Rum::messages() *[line 231]*

returns the application messages as an HTML element

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

PageControllerBase function Rum::requestHandler() *[line 272]*

This method returns the current controller

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

void function Rum::sendHTTPError([$statuscode = 500]) *[line 322]*
Function Parameters:

- *int* **$statuscode** HTTP status code

This method sends an HTTP status message to the client

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

string function Rum::tl($stringId, [$default = '']) *[line 336]*
Function Parameters:

- *string* **$stringId** string id to translate
- *string* **$default** string if not found

This method translates a message into the appropriate language

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

void function Rum::trace($var) *[line 284]*
Function Parameters:

- *mixed* **$var** raw variable to trace

This method traces a variable

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

string function Rum::uri([$page = ''], [$args = array()]) *[line 297]*
Function Parameters:

- *string* **$page** name of page
- *array* **$args** array of parameters

This method returns a URI based on the requested page and parameters

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

string function Rum::url([$page = ''], [$args = array()]) *[line 310]*
Function Parameters:

- *string* **$page** name of the page
- *array* **$args** array of parameters

This method returns a URL based on the requested page and parameters

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

Class Tester

[line 18]

Provides the base funcionality of the application. This class recieves command line data and executes the application

- **Package** PHPRum
- **Author** Darnell Shinbine

array function Tester::getAllFunctionalTestCases([$path = '']) *[line 172]*

Function Parameters:

- *string* **$path** initial path

return all functional test cases

- **Access** public

array function Tester::getAllUnitTestCases() *[line 150]*

return all unit test cases

- **Access** public

ControllerTestCaseBase function Tester::getFunctionalTestCase($module) *[line 106]*
Function Parameters:

- *string* **$module** name of module

return functional test case

- **Access** public

UnitTestCaseBase function Tester::getUnitTestCase($testCase) *[line 77]*
Function Parameters:

- *string* **$testCase** name of test

return unit test case

- **Access** public

void function Tester::loadTestConfig($reporter) *[line 206]*
Function Parameters:

- *\SimpleReporter* **$reporter** instance of a \SimpleReporter

run all test cases

- **Access** public

void function Tester::runAllTestCases(\$reporter, $reporter) *[line 52]*
Function Parameters:

- \SimpleReporter **$reporter** instance of a \SimpleReporter
- *SimpleReporter* **\$reporter**

run all test cases

- **Access** public

void function Tester::runFunctionalTestCase($testCase, \$reporter, $reporter) *[line 40]*
Function Parameters:

- *string* **$testCase** name of module
- \SimpleReporter **$reporter** instance of a \SimpleReporter
- *SimpleReporter* **\$reporter**

run functional test case

- **Access** public

void function Tester::runUnitTestCase($testCase, \$reporter, $reporter) *[line 27]*
Function Parameters:

- *string* **$testCase** name of test

- \SimpleReporter **$reporter** instance of a \SimpleReporter
- *SimpleReporter* **\$reporter**

run unit test case

- **Access** public

Class TypeMismatchException

[line 17]

Represents a Type Mismatch Exception

- **Package** PHPRum
- **Author** Darnell Shinbine

Class WebApplicationBase

[line 31]

This class represents the base web application. This class recieves HTTP request input and delegates the request to a Controller. Once the Controller has handled the request, this class will regain control and render the appropriate View selected by the Controller unless there are additional actions to perform.

- **Package** PHPRum
- **Property** PageControllerBase $requestHandler: Reference to the current ControllerBase
- **Property** string $prevPage: Previous page
- **Property** string $currentPage: Current page
- **Property** string $forwardPage: Page to forward to
- **Property** string $forwardURI: URI to forward to
- **Property** array $forwardParams: array of forward parameters
- **Property** Session $session: Contains the Session object
- **Property** HTTPRequest $request: Contains the HTTPRequet object
- **Property** AppMessageCollection $messages: Contains the AppMessageCollection object
- **Author** Darnell Shinbine
- **Abstract Element**

Constructor *void* function WebApplicationBase::__construct() *[line 107]*

Constructor

Creates an instance of the controller and sets default action map.

- **Access** public

void function WebApplicationBase::clearForwardPage() *[line 375]*

This method clear all forward URI's

- **Access** public

void function WebApplicationBase::execute() *[line 490]*

execute the application

- **Access** protected

array function WebApplicationBase::getAllControllers([$path = '']) *[line 400]*
Function Parameters:

- *string* **$path** initial path

return all controllers

- **Access** public

string function WebApplicationBase::getPageURI([$page = ''], [$args = array()]) *[line 228]*
Function Parameters:

- *string* **$page** name of page
- *array* **$args** array of parameters

returns a URI based on the requested page and parameters

- **Access** public

ControllerBase function WebApplicationBase::getRequestHandler($controller) *[line 183]*
Function Parameters:

- *string* **$controller** name of the controller

return the requestHandler

- **Access** public

ITranslator function WebApplicationBase::getSession() *[line 479]*

returns the common translator object (overrided this method in the application class)

- **Access** protected

void function WebApplicationBase::handleError($errno, $errstr, $errfile, $errline) *[line 831]*

Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description
- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Access** protected

void function WebApplicationBase::handleException(\$e, $e) *[line 692]*

Function Parameters:

- *Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden to provide error handling.

- **Access** protected

void function WebApplicationBase::loadApplicationState() *[line 433]*

This method retrieves an instance of the previous action and the servlet messages. These can then be used by the servlet to populate the view.

- **Access** public

void function WebApplicationBase::requestProcessor(\\\&$request, $request) *[line 528]*

Function Parameters:

- *HTTPRequest* **$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

Processe HTTP Request and create actions to handle business logic. Then render the view.

Can be overwridden to provide additional or alternaitive private functionality.

- **Access** protected

void function WebApplicationBase::saveApplicationState() *[line 461]*

This method stores an instance of the previous action and the servlet messages. These can then be used by the servlet after a page request to populate the view.

- **Access** public

void function WebApplicationBase::sendHTTPError([$statuscode = 500]) *[line 313]*

Function Parameters:

- *int* **$statuscode** HTTP status code

send HTTP status message to client

- **Access** public

void function WebApplicationBase::setForwardPage([$nextPage = ''], [$args = array()], [$method = null]) *[line 346]*

Function Parameters:

- *string* **$nextPage** Name of requested page
- *array* **$args** args for next action
- *ForwardMethodType* **$method** forward method as constant of ForwardMethodType::URI() or ForwardMethodType::Request()

This method sets the next URI once the current controller is finished executing, the servlet will re-request the page with the requested action. This method allows you to replace post data with keep friendly urls that can be bookmarked.

- **Access** public

void function WebApplicationBase::trace($var) *[line 388]*

Function Parameters:

- **$var**

Trace a variable

- **Access** public

activerecordbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

activerecordcollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

activerecordexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

relationshiptype.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class ActiveRecordBase

[line 32]

This class represents a table withing a database or an instance of a single record in a table and provides database abstraction.

The ActiveRecordBase exposes 5 protected properties, do not define these properties in the sub class to have the properties auto determined

- **Package** PHPRum
- **Sub-Package** ActiveRecord
- **Property** string $table: Specifies the table name
- **Property** string $pkey: Specifies the primary key (there can only be one primary key defined)
- **Property** array $fields: Specifies field names mapped to field types
- **Property** array $rules: Specifies field names mapped to field rules
- **Property** array $relationahips: Specifies table relationships

  2 Additional public readonly properties are exposed

- **Property** bool $isNull: Specifies whether ActiveRecord is null

- **Property** bool $isEmpty: Specifies whether ActiveRecord is empty
- **Author** Darnell Shinbine
- **Abstract Element**

ActiveRecordBase::$pkey

string = '' *[line 44]*

Specifies the primary key (there can only be one primary key defined)

- **Access** protected

ActiveRecordBase::$relationships

array = array() *[line 50]*

Specifies table relationships

- **Access** protected

ActiveRecordBase::$table

string = '' *[line 38]*

Specifies the table name

- **Access** protected

Constructor *void* function ActiveRecordBase::__construct() *[line 64]*

Constructor

- **Access** protected

void function ActiveRecordBase::afterCreate() *[line 847]*

event called after a new ActiveRecordBase is created

- **Access** protected

void function ActiveRecordBase::afterDelete() *[line 927]*

event called after every delete

- **Access** protected

void function ActiveRecordBase::afterInsert() *[line 895]*

event called after every insert

- **Access** protected

void function ActiveRecordBase::afterRetrieve() *[line 863]*

event called after every retrieve

- **Access** protected

void function ActiveRecordBase::afterSave() *[line 879]*

event called after every save

- **Access** protected

void function ActiveRecordBase::afterUpdate() *[line 911]*

event called after every update

- **Access** protected

DataSet function ActiveRecordBase::all([$args = array()]) *[line 628]*

Function Parameters:

- *array* **$args** associative array of keys and values

static method to return a DataSet of this type

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

void function ActiveRecordBase::beforeCreate() *[line 839]*

event called before a new ActiveRecordBase is created

- **Access** protected

void function ActiveRecordBase::beforeDelete() *[line 919]*

event called before every delete

- **Access** protected

void function ActiveRecordBase::beforeInsert() *[line 887]*

event called before every insert

- **Access** protected

void function ActiveRecordBase::beforeRetrieve() *[line 855]*

event called before every retrieve

- **Access** protected

void function ActiveRecordBase::beforeSave() *[line 871]*

event called before every save

- **Access** protected

void function ActiveRecordBase::beforeUpdate() *[line 903]*

event called before every update

- **Access** protected

DataSet function ActiveRecordBase::count([$args = array()]) *[line 652]*
Function Parameters:

- *array* **$args** associative array of keys and values

static method to return a count of records of this type

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

ActiveRecordBase function ActiveRecordBase::create([$args = array()]) *[line 553]*
Function Parameters:

- *array* **$args** optional associative array of initial properties

static method to create new ActiveRecordBase of this type

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

void function ActiveRecordBase::delete() *[line 471]*

delete ActiveRecordBase

- **Access** public

void function ActiveRecordBase::deleteAll() *[line 492]*

delete ActiveRecordBase and all associated ActiveRecordBases (cascading delete)

- **Access** public

ActiveRecordBase function ActiveRecordBase::find($args) *[line 565]*

Function Parameters:

- *array* **$args** associative array of keys and values

static method to find ActiveRecordBase of this type

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

ActiveRecordCollection function ActiveRecordBase::findAll([$args = array()]) *[line 592]*

Function Parameters:

- *array* **$args** associative array of keys and values

static method to find ActiveRecordBases of this type

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

ActiveRecordBase function ActiveRecordBase::findById($pid) *[line 577]*
Function Parameters:

- *mixed* **$pid** key value to lookup

static method to find ActiveRecordBase by primary key

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

ActiveRecordBase function ActiveRecordBase::first([$args = array()]) *[line 604]*
Function Parameters:

- *string* **$args** associative array of keys and values

static method to find the first ActiveRecordBase of this type

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

Form function ActiveRecordBase::form($controlId) *[line 664]*
Function Parameters:

- *string* **$controlId** form id

static method to return a Form object

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

ActiveRecordBase function ActiveRecordBase::last([$args = array()]) *[line 616]*

Function Parameters:

- *string* **$args** associative array of keys and values

static method to find the last ActiveRecordBase of this type

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

DataSet function ActiveRecordBase::more([$args = array()]) *[line 640]*

Function Parameters:

- *array* **$args** associative array of keys and values

static method to return an extended DataSet of this type

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

void function ActiveRecordBase::save() *[line 445]*

update DataAdapter

- **Access** public

array function ActiveRecordBase::toArray() *[line 536]*

returns an array containing all row data

- **Access** public

Class ActiveRecordCollection

[line 19]

Represents a collection of ActiveRecords

- **Package** PHPRum
- **Sub-Package** ActiveRecord
- **Author** Darnell Shinbine

void function ActiveRecordCollection::add($item) *[line 51]*

Function Parameters:

- *ActiveRecordBase* **$item**

Add an ActiveRecord to the Collection

- **Access** public

ActiveRecordBase function ActiveRecordCollection::itemAt($index) *[line 70]*
Function Parameters:

- *int* **$index** index of ActiveRecord

return ActiveRecord at a specified index

- **Access** public

Class ActiveRecordException

[line 18]

Represents a Bad Member Call Exception

- **Package** PHPRum
- **Sub-Package** ActiveRecord
- **Author** Darnell Shinbine

Class RelationshipType

[line 18]

Represents a relationship type

- **Package** PHPRum
- **Sub-Package** ActiveRecord
- **Author** Darnell Shinbine

RelationshipType function RelationshipType::BelongsTo() *[line 37]*

Specifies the relationship is a many to one

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

RelationshipType function RelationshipType::HasMany() *[line 31]*

Specifies the relationship is a one to many

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

RelationshipType function RelationshipType::HasManyAndBelongsTo() *[line 43]*

Specifies the relationship is a many to many

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

string function RelationshipType::__toString() *[line 49]*

Returns a string representing the type

- **Access** public

applicationrunevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

applicationruneventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagecontrollercreatepageevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagecontrollercreatepageeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationhandlerequestevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationhandlerequesteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationloadstateevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationloadstateeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationsavestateevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webapplicationsavestateeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class ApplicationRunEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function ApplicationRunEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class ApplicationRunEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function ApplicationRunEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class PageControllerCreatePageEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base

- **Author** Darnell Shinbine

Constructor *void* function PageControllerCreatePageEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class PageControllerCreatePageEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function PageControllerCreatePageEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebApplicationHandleRequestEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function WebApplicationHandleRequestEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class WebApplicationHandleRequestEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base

- **Author** Darnell Shinbine

Constructor *void* function WebApplicationHandleRequestEventHandler::__construct($callback) *[line 27]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebApplicationLoadStateEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function WebApplicationLoadStateEvent::__construct() *[line 26]*
Constructor

- **Access** public

Class WebApplicationLoadStateEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function WebApplicationLoadStateEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebApplicationSaveStateEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function WebApplicationSaveStateEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class WebApplicationSaveStateEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Base
- **Author** Darnell Shinbine

Constructor *void* function WebApplicationSaveStateEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

build.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

cachebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

cacheexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dbcache.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filecache.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

memcache.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Build

[line 18]

Provides access to read/write build files

- **Package** PHPRum
- **Sub-Package** Caching
- **Author** Darnell Shinbine

Build::$verbose

bool = false *[line 24]*

verbose

- **Static**

void function Build::build() *[line 155]*

Rebuild build folder

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

void function Build::clean() *[line 128]*

Cleans and prepares build folder

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

void function Build::clear($id) *[line 103]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached

Deletes a value with the specified key from cache

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

string function Build::get($id, $expire) *[line 35]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *integer* **$expire** the expiration time of the value in seconds

1. means never expire,

Retrieves a value from cache with a specified key.

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

array function Build::getFileArray() *[line 185]*

get id array

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

void function Build::put($id, $value) *[line 68]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *string* **$value** the value to be cached

Stores a value identified by a key into cache.

If the cache already contains such a key, the existing value and expiration time will be replaced with the new ones.

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

Class CacheBase

[line 18]

Provides access to read/write cache objects

- **Package** PHPRum
- **Sub-Package** Caching
- **Author** Darnell Shinbine
- **Abstract Element**

void function CacheBase::clear($id) *[line 52]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached

Deletes a value with the specified key from cache

- **Abstract Element**
- **Access** public

void function CacheBase::flush() *[line 60]*

Deletes all values from cache.

- **Abstract Element**
- **Access** public

string function CacheBase::get($id, $expire) *[line 28]*
Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *integer* **$expire** the expiration time of the value in seconds

1. means never expire,

Retrieves a value from cache with a specified key.

- **Abstract Element**
- **Access** public

void function CacheBase::put($id, $value, [$expires = 0]) *[line 43]*
Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *string* **$value** the value to be cached
- *integer* **$expires** the expiration time of the value in seconds

1. means never expire,

Stores a value identified by a key into cache.
If the cache already contains such a key, the existing value and expiration time will be replaced with the new ones.

- **Abstract Element**
- **Access** public

Class CacheException

[line 19]

Represents a Cache Exception

- **Package** PHPRum
- **Sub-Package** Caching
- **Author** Darnell Shinbine

Class DBCache

[line 18]

Provides access to read/write cache objects from a database

- **Package** PHPRum
- **Sub-Package** Caching
- **Author** Darnell Shinbine

Constructor *void* function DBCache::__construct([$table = ''], [$dsn = '']) *[line 39]*
Function Parameters:

- *string* **$table** table name
- *string* **$dsn** database connection string

Constructor

- **Access** public

void function DBCache::clear($id) *[line 146]*
Function Parameters:

- *string* **$id** the key identifying the value to be cached

Deletes a value with the specified key from cache

- **Access** public

void function DBCache::flush() *[line 161]*

Deletes all values from cache.

- **Access** public

string function DBCache::get($id) *[line 63]*
Function Parameters:

- *string* **$id** the key identifying the value to be cached

Retrieves a value from cache with a specified key.

- **Access** public

array function DBCache::getIdArray() *[line 174]*

get id array

- **Access** public

void function DBCache::put($id, $value, [$expires = 0]) *[line 124]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *string* **$value** the value to be cached
- *integer* **$expires** the expiration time of the value in seconds

1. means never expire,

Stores a value identified by a key into cache.

If the cache already contains such a key, the existing value and expiration time will be replaced with the new ones.

- **Access** public

Class FileCache

[line 18]

Provides access to read/write cache files

- **Package** PHPRum
- **Sub-Package** Caching
- **Author** Darnell Shinbine

void function FileCache::clear($id) *[line 98]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached

Deletes a value with the specified key from cache

- **Access** public

void function FileCache::flush() *[line 121]*

Deletes all values from cache.

- **Access** public

string function FileCache::get($id) *[line 26]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached

Retrieves a value from cache with a specified key.

- **Access** public

array function FileCache::getIdArray() *[line 143]*

get id array

- **Access** public

void function FileCache::put($id, $value, [$expires = 0]) *[line 65]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *string* **$value** the value to be cached
- *integer* **$expires** the expiration time of the value in seconds

1. means never expire,

Stores a value identified by a key into cache.

If the cache already contains such a key, the existing value and expiration time will be replaced with the new ones.

- **Access** public

Class MEMCache

[line 18]

Provides access to read/write cache objects from memcache

- **Package** PHPRum
- **Sub-Package** Caching
- **Author** Darnell Shinbine

Constructor *void* function MEMCache::__construct([$server = 'localhost'], [$port = 11211]) *[line 38]*

Function Parameters:

- *string* **$server** server, default is localhost
- *string* **$port** port, default is 11211

Constructor

- **Access** public

void function MEMCache::clear($id) *[line 91]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached

Deletes a value with the specified key from cache

- **Access** public

void function MEMCache::flush() *[line 102]*

Deletes all values from cache.

- **Access** public

string function MEMCache::get($id) *[line 58]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached

Retrieves a value from cache with a specified key.

- **Access** public

void function MEMCache::put($id, $value, [$expires = 0], $expire) *[line 76]*

Function Parameters:

- *string* **$id** the key identifying the value to be cached
- *string* **$value** the value to be cached
- *integer* **$expire** the expiration time of the value in seconds

1. means never expire,

- **$expires**

Stores a value identified by a key into cache.
If the cache already contains such a key, the existing value and expiration time will be replaced with the new ones.

- **Access** public

appmessagecollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

appsettingscollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

eventargs.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

eventcollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

eventhandlercollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

collectionbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dictionarybase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

iteratorbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

keycollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

stringcollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

stringdictionary.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

valuecollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class AppMessageCollection

[line 18]

Represents a collection of Application Messages

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

bool function AppMessageCollection::add($item) *[line 62]*

Function Parameters:

- *AppMessage* **$item**

add AppMessage to Collection

- **Access** public

StringCollection function AppMessageCollection::getByType([$type = null]) *[line 94]*

Function Parameters:

- *constant* **$type** Message type as constant of AppMessageType::Success(), AppMessageType::Fail() or AppMessageType::Notice()

return Collection of messages

- **Access** public

AppMessage function AppMessageCollection::itemAt($index) *[line 82]*

Function Parameters:

- *int* **$index** index of ActiveRecord

return AppMessage at a specified index

- **Access** public

array function AppMessageCollection::toString() *[line 114]*

convert Iterator object into string

- **Access** public

Class AppSettingsCollection

[line 18]

Represents a Collection of user-defined settings

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

Class CollectionBase

[line 18]

Provides the base functionality for creating Collections

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine
- **Abstract Element**

Constructor *void* function CollectionBase::__construct([$collection = null]) *[line 26]*

Function Parameters:

- *mixed* **$collection** can be CollectionBase or array used to initialize Collection

Constructor

- **Access** public

void function CollectionBase::add($item) *[line 70]*
Function Parameters:

- *mixed* **$item** item

add item to Collection

- **Access** public

bool function CollectionBase::contains($item) *[line 82]*
Function Parameters:

- *mixed* **$item** item

returns true if item array contains item

- **Access** public

int function CollectionBase::indexOf($item) *[line 95]*
Function Parameters:

- *mixed* **$item** item

returns first index of item found in Collection

- **Access** public

mixed function CollectionBase::itemAt($index) *[line 115]*
Function Parameters:

- *int* **$index** index of item

return item at a specified index

- **Access** public

bool function CollectionBase::remove($item) *[line 135]*
Function Parameters:

- *mixed* **$item** item

remove item from array

- **Access** public

void function CollectionBase::removeAll() *[line 176]*

emptys the collection

- **Access** public

void function CollectionBase::removeAt($index) *[line 157]*

Function Parameters:

- *int* **$index** index to remove at

remove item from array at a specified index

- **Access** public

Class DictionaryBase

[line 21]

Provides the base functionality for creating Dictionaries

- **Package** PHPRum
- **Sub-Package** Collections
- **Property** KeyCollection $keys: Collection of keys
- **Property** ValueCollection $values: Collection of values
- **Author** Darnell Shinbine
- **Abstract Element**

Constructor *void* function DictionaryBase::__construct([$dictionary = null]) *[line 29]*

Function Parameters:

- *mixed* **$dictionary** can be DictionaryBase or array used to initialize Dictionary

Constructor

- **Access** public

void function DictionaryBase::add($key, $value) *[line 80]*
Function Parameters:

- *mixed* **$key** key
- *mixed* **$value** value

Adds a new key/item pair to a Dictionary object if key does not already exist

- **Access** public

bool function DictionaryBase::contains($key) *[line 118]*
Function Parameters:

- *mixed* **$key** key

returns a Boolean value that indicates whether a specified key exists in the Dictionary object

- **Access** public

int function DictionaryBase::indexOf($key) *[line 130]*
Function Parameters:

- *mixed* **$key** key

returns index of key found in Dictionary

- **Access** public

mixed function DictionaryBase::item($key) *[line 99]*
Function Parameters:

- *mixed* **$key** key

return the value of a key in the Dictionary

- **Access** public

mixed function DictionaryBase::itemAt($index) *[line 150]*
Function Parameters:

- *int* **$index** index

returns the value of a specified index

- **Access** public

mixed function DictionaryBase::keyAt($index) *[line 170]*

Function Parameters:

- *int* **$index** index

returns the key of a specified index

- **Access** public

void function DictionaryBase::remove($key) *[line 190]*

Function Parameters:

- *string* **$key** key to remove

remove key/value pair from Dictionary

- **Access** public

void function DictionaryBase::removeAll() *[line 221]*

emptys the dictionary

- **Access** public

void function DictionaryBase::removeAt($index) *[line 210]*

Function Parameters:

- *int* **$index** index to remove

remove key/value of a specified index

- **Access** public

Class EventArgs

[line 18]

Represents a Collection of Event args

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

Class EventCollection

[line 21]

Represents a Collection of EventHandlerBase objects

- **Package** PHPRum
- **Sub-Package** Collections
- **Property** EventHandlerCollection $eventHandlers: event handler collection
- **Property** EventCollection $events: event collection
- **Author** Darnell Shinbine

Constructor *void* function EventCollection::__construct([$collection = null]) *[line 36]*
Function Parameters:

- *mixed* **$collection** can be CollectionBase or array used to initialize Collection

Constructor

- **Access** public

bool function EventCollection::contains($item) *[line 119]*
Function Parameters:

- *EventBase* **$item** item

returns true if item array contains item

- **Access** public

int function EventCollection::indexOf($item) *[line 144]*
Function Parameters:

- *EventBase* **$item** item

returns first index of item found in Collection

- **Access** public

void function EventCollection::raise($event, &$sender, [$args = array()], $sender) *[line 185]*

Function Parameters:

- *EventBase* **$event** event
- *object* **$sender** Sender object
- *array* **$args** optional args
- **&$sender**

raise all events

- **Access** public

void function EventCollection::registerEventHandler($eventHandler) *[line 171]*

Function Parameters:

- *EventHandlerBase* **$eventHandler**

register event handler

- **Access** public

Class EventHandlerCollection

[line 18]

Represents a Collection of EventHandlerBase objects

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

void function EventHandlerCollection::raise($event, &$sender, [$args = array()], $sender) *[line 86]*

Function Parameters:

- *EventBase* **$event** event
- *object* **$sender** Sender object
- *array* **$args** optional args
- **&$sender**

raise all events

- **Access** public

void function EventHandlerCollection::registerEventHandler($eventHandler) *[line 72]*

Function Parameters:

- *EventHandlerBase* **$eventHandler**

register event handler

- **Access** public

Class IteratorBase

[line 23]

Provides the base functionality for creating Iterators

- **Package** PHPRum
- **Sub-Package** Collections
- **Property** int $count: number of collection items
- **Author** Darnell Shinbine
- **Abstract Element**

IteratorBase::$items

array = array() *[line 29]*

internal array

- **Access** protected

Constructor *void* function IteratorBase::__construct([$iterator = null]) *[line 73]*

Function Parameters:

- *IteratorBase* **$iterator** used to initialize DictionaryBase

Constructor

- **Access** public

array function IteratorBase::toArray() *[line 201]*

convert Iterator object into array

- **Access** public

Class KeyCollection

[line 18]

Represents a collection of keys

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

string function KeyCollection::itemAt($index) *[line 26]*

Function Parameters:

- *int* **$index** index of item

return key at a specified index

- **Access** public

Class StringCollection

[line 18]

Represents a collection of strings

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

void function StringCollection::add($item) *[line 50]*
Function Parameters:

- *string* **$item** item

add item to Collection

- **Access** public

string function StringCollection::itemAt($index) *[line 69]*

Function Parameters:

- *int* **$index** index of item

return item at a specified index

- **Access** public

void function StringCollection::trim() *[line 80]*

trim whitespace for all values

- **Access** public

Class StringDictionary

[line 18]

Represents a dictionary of key/value strings

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

void function StringDictionary::add($key, $value) *[line 51]*

Function Parameters:

- *string* **$key** key
- *string* **$value** value

Adds a new key/item pair to a StringCollection if key does not already exist

- **Access** public

string function StringDictionary::item($key) *[line 77]*

Function Parameters:

- *string* **$key** key

return the value of a key in the Dictionary

- **Access** public

string function StringDictionary::itemAt($index) *[line 89]*

Function Parameters:

- *int* **$index** index

returns the value of a specified index

- **Access** public

string function StringDictionary::keyAt($index) *[line 101]*

Function Parameters:

- *int* **$index** index

returns the key of a specified index

- **Access** public

void function StringDictionary::trim() *[line 112]*

trim whitespace for all values

- **Access** public

Class ValueCollection

[line 18]

Represents a collection of values

- **Package** PHPRum
- **Sub-Package** Collections
- **Author** Darnell Shinbine

string function ValueCollection::itemAt($index) *[line 26]*

Function Parameters:

- *int* **$index** index of item

return value at a specified index

- **Access** public

httpwebrequest.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

httpwebresponse.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class HTTPWebRequest

[line 25]

Represents an HTTP request (supports SSL)

- **Package** PHPRum
- **Sub-Package** Comm
- **Property** string $method: HTTP request method
- **Property** string $contentType: content-type
- **Property** string $referer: HTTP referer
- **Property** string $timeout: timeout
- **Property** string $httpVersion: HTTP version
- **Property** string $url: URL to send to
- **Author** Darnell Shinbine

void function HTTPWebRequest::addHeader($header) *[line 173]*

Function Parameters:

- *string* **$header** header to set

add header to request

SocketStream function HTTPWebRequest::getRequestStream() *[line 332]*

returns a Stream object for request

- **Access** protected

string function HTTPWebRequest::getRequestString() *[line 254]*

returns a string for request

- **Access** protected

HTTPWebResponse function HTTPWebRequest::getResponse() *[line 235]*

sends request and creates HTTPWebResponse object before closing Stream object

void function HTTPWebRequest::sendRequest() *[line 222]*

sends request and closes Stream object

void function HTTPWebRequest::setData($data) *[line 192]*

Function Parameters:

- *mixed* **$data** data to set (string|array)

set data from string or array

Class HTTPWebResponse

[line 23]

Represents the response from a HTTPWebRequest message

- **Package** PHPRum
- **Sub-Package** Comm
- **Property** string $httpStatus: response headers
- **Property** array $headers: response headers
- **Property** string $contentType: response headers
- **Property** string $content: response headers
- **Author** Darnell Shinbine

void function HTTPWebResponse::readFromStream(\\\&$stream, $header) *[line 88]*

Function Parameters:

- *string* **$header** header to set
- *StreamBase* **\\\&$stream**

read data from Stream object

- **Access** public

columncompare.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

columnschema.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dataadapter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dataadapterexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

databaseexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

databaseschema.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dataset.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

datasettype.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dirdataadapter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mssqldataadapter.class.php

- **Package** PHPRum
- **Author** Tahsin Zulkarnine
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mssqlquerybuilder.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mssqltransaction.class.php

- **Package** PHPRum
- **Author** Tahsin Zulkarnine
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mysqlidataadapter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mysqliquerybuilder.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mysqlitransaction.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mysqldataadapter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mysqlquerybuilder.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mysqltransaction.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pdodataadapter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pdoquerybuilder.class.php

- **Package** PHPRum
- **Author** Tahsin Zulkarnine
- **Copyright** Copyright (c) 2013
- **Filesource** Source Code for this file
- **License** see

pdotransaction.class.php

- **Package** PHPRum
- **Author** Tahsin Zulkarnine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

querybuilderbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

queryexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

schemaexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

tableschema.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

textdataadapter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

transactionbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

transactionexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class ColumnCompare

[line 18]

Provides array comparing by type

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

Constructor *void* function ColumnCompare::__construct($index, $field) *[line 34]*

Function Parameters:

- *string* **$field** column name
- **$index**

Constructor

sets column name to compare

- **Access** public

int function ColumnCompare::compareDateString($a, $b) *[line 75]*

Function Parameters:

- *ints* **$a** ints a
- *ints* **$b** ints b

method compares ints and returns -1, 0, 1

- **Access** public

int function ColumnCompare::compareInt($a, $b) *[line 124]*

Function Parameters:

- *ints* **$a** ints a
- *ints* **$b** ints b

method compares ints and returns -1, 0, 1

- **Access** public

int function ColumnCompare::compareNumeric($a, $b) *[line 104]*

Function Parameters:

- *ints* **$a** ints a

- *ints* **$b** ints b

method compares numbers and returns -1, 0, 1

- **Access** public

int function ColumnCompare::compareString($a, $b) *[line 60]*
Function Parameters:

- *string* **$a** string a
- *string* **$b** string b

method compares strings and returns -1, 0, 1

- **Access** public

int function ColumnCompare::compareStringi($a, $b) *[line 45]*
Function Parameters:

- *string* **$a** string a
- *string* **$b** string b

method compares strings and returns -1, 0, 1

- **Access** public

Class ColumnSchema

[line 40]

Represents a column schema

- **Package** PHPRum
- **Sub-Package** DB
- **Property** string $name: column name
- **Property** string $table: table name
- **Property** string $type: column type
- **Property** int $length: column length
- **Property** bool $notNull: specifies whether column is not null
- **Property** bool $primaryKey: specifies whether column is a primary key
- **Property** bool $multipleKey: specifies whether column is a multiple key
- **Property** bool $foreignKey: specifies whether column is a foriegn key
- **Property** bool $autoIncrement: specifies whether column is auto incrementing
- **Property** bool $unique: specifies whether column is unique
- **Property** bool $numeric: specifies whether column is numeric
- **Property** bool $string: specifies whether column is a string
- **Property** bool $integer: specifies whether column is an integer
- **Property** bool $real: specifies whether column is a double
- **Property** bool $boolean: specifies whether column is true/false
- **Property** bool $year: specifies whether column is year string
- **Property** bool $date: specifies whether column is date string
- **Property** bool $time: specifies whether column is time string

- **Property** bool $datetime: specifies whether column is a date/time string
- **Property** bool $blob: specifies whether column is a blob
- **Property** bool $binary: specifies whether column is binary
- **Author** Darnell Shinbine

Constructor *void* function ColumnSchema::__construct($properties) *[line 175]*
Function Parameters:

- *array* **$properties** initial properties

Constructor

- **Access** public

mixed function ColumnSchema::__get($field) *[line 204]*
Function Parameters:

- *string* **$field** name of the field

returns an object property

- **Access** public

Class DataAdapter

[line 21]

Provides access to a generic data source

- **Package** PHPRum
- **Sub-Package** DB
- **Property** bool $opened: Specifies whether the connection is open
- **Property** bool $closed: Specifies whether the connection is closed
- **Author** Darnell Shinbine
- **Abstract Element**

DataAdapter::$args

array = array() *[line 27]*

A collection of connection variables

- **Access** protected

DataAdapter::$cacheEnabled

bool = false *[line 45]*

Specifies whether caching is enabled

- **Access** protected

DataAdapter::$cacheExpires

int = 300 *[line 51]*

Specifies the caching expiration in seconds

- **Access** protected

DataAdapter::$charset

string = 'utf-8' *[line 39]*

Specifies the character set

- **Access** protected

DataAdapter::$collation

string = 'utf8_general_ci' *[line 33]*

Specifies the db collation

- **Access** protected

Constructor *void* function DataAdapter::__construct($args) *[line 102]*
Function Parameters:

- *array* **$args** connection arguments

Constructor

- **Access** protected

DatabaseSchema function DataAdapter::addTableSchema(&$tableSchema) *[line 426]*

Function Parameters:

- ***TableSchema*** **&$tableSchema**

creats a TableSchema object

- **Abstract Element**
- **Access** public

DatabaseSchema function DataAdapter::alterTableSchema(&$tableSchema) *[line 434]*

Function Parameters:

- ***TableSchema*** **&$tableSchema**

alters a TableSchema object

- **Abstract Element**
- **Access** public

TransactionBase function DataAdapter::beginTransaction() *[line 518]*

creats a Transaction object

- **Abstract Element**
- **Access** public

DatabaseSchema function DataAdapter::buildSchema() *[line 418]*

builds a DataBaseSchema object

- **Abstract Element**
- **Access** public

bool function DataAdapter::close() *[line 467]*

closes an open connection

- **Abstract Element**
- **Access** public

DataAdapter function DataAdapter::create($dsn) *[line 171]*

Function Parameters:

- *string* **$dsn** connection string

creates a connection object to a datasource

Examples:

```
adapter=mysql;uid=root;pwd=;server=localhost;database=northwind;charset=utf8;pconnect=false;cache_enabled=false;cache_expires=300;
    adapter=mssql;uid=root;pwd=;server=localhost;database=northwind;charset=utf8;
    adapter=text;format=TabDelimited;source=/northwind.csv;charset=utf8;
    adapter=dir;source=/northwind;
```

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

void function DataAdapter::delete(&$ds) *[line 502]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet

attempt to delete a record in the datasource

- **Abstract Element**
- **Access** public

void function DataAdapter::disableCaching($expires) *[line 296]*

Function Parameters:

- *int* **$expires** cache expiration in seconds

disable caching

- **Access** public

DatabaseSchema function DataAdapter::dropTableSchema(&$tableSchema) *[line 442]*

Function Parameters:

- *TableSchema* **&$tableSchema**

drops a TableSchema object

- **Abstract Element**
- **Access** public

void function DataAdapter::enableCaching([$expires = null]) *[line 279]*
Function Parameters:

- *int* **$expires** cache expiration in seconds

enable caching

- **Access** public

string function DataAdapter::escapeString($unescaped_string) *[line 543]*
Function Parameters:

- *string* **$unescaped_string** String to escape

Returns an escaped string

- **Abstract Element**
- **Access** public

void function DataAdapter::execute($query) *[line 308]*
Function Parameters:

- *string* **$query** query to execute

Executes a query procedure on the current connection

- **Access** public

void function DataAdapter::executeBatch($batch) *[line 320]*

Function Parameters:

- *string* **$batch** sql batch query

Executes a query batch or procedure on the current connection

- **Access** public

void function DataAdapter::fill(&$ds, $ds) *[line 451]*

Function Parameters:

- *DataSet* **$ds** reference to a DataSet object
- *DataSet* **&$ds**

fetches DataSet from datasource string using source string

- **Abstract Element**
- **Access** public

int function DataAdapter::getAffectedRows() *[line 534]*

return affected rows

- **Abstract Element**
- **Access** public

int function DataAdapter::getLastInsertId() *[line 526]*

return id of last record inserted

- **Abstract Element**
- **Access** public

string function DataAdapter::getLastQuery() *[line 340]*

return last executed query

- **Access** public

real function DataAdapter::getLastQueryTime() *[line 351]*

return last executed query time in milliseconds

- **Access** public

int function DataAdapter::getQueryCount() *[line 362]*

returns the query count

- **Access** public

int function DataAdapter::getQueryTime() *[line 373]*

returns the total query time

- **Access** public

DatabaseSchema function DataAdapter::getSchema() *[line 384]*

returns a DataBaseSchema object

- **Access** public

void function DataAdapter::insert(&$ds, $ds) *[line 484]*

Function Parameters:

- *DataSet* **$ds** reference to a DataSet
- *DataSet* **&$ds**

attempt to insert a record into the datasource

- **Abstract Element**
- **Access** public

bool function DataAdapter::open() *[line 459]*

opens a connection to a datasource

- **Abstract Element**
- **Access** public

DataSet function DataAdapter::openDataSet([$source = ''], [$lock_type = null]) *[line 251]*
Function Parameters:

- *string* **$source** source string (or QueryBuilder)
- *DataSetType* **$lock_type** lock type as constant of DataSetType::OpenDynamic(), DataSetType::OpenStatic(), or DataSetType::OpenReadonly()

opens a DataSet specified by the source

- **Access** public

bool function DataAdapter::opened() *[line 475]*
returns true if a connection to a datasource is currently open

- **Abstract Element**
- **Access** public

resource function DataAdapter::query($query, $buffer) *[line 553]*
Function Parameters:

- *string* **$query** query string
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Abstract Element**
- **Access** protected

QueryBuilderBase function DataAdapter::queryBuilder() *[line 510]*

creats a QueryBuilder object

- **Abstract Element**
- **Access** public

void function DataAdapter::rebuildSchema() *[line 407]*

rebuilds the DataBaseSchema object

- **Access** public

resource function DataAdapter::runQuery($query, [$buffer = true]) *[line 563]*

Function Parameters:

- *string* **$query** query to execute
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Access** protected

void function DataAdapter::update(&$ds) *[line 493]*
Function Parameters:

- *DataSet* **&$ds** reference to a DataSet

attempt to update a record in the datasource

- **Abstract Element**
- **Access** public

Class DataAdapterException

[line 19]

Represents a DataAdapter Exception

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

Class DatabaseException

[line 19]

Represents a DataBase Exception

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

Class DatabaseSchema

[line 21]

Represents a database schema

- **Package** PHPRum
- **Sub-Package** DB
- **Property** array $tableSchemas: array of table schemas
- **Property** array $tableNames: array of table names
- **Author** Darnell Shinbine

Constructor *void* function DatabaseSchema::__construct($properties, $tableSchemas, $dataAdapter) *[line 36]*

Function Parameters:

- *DataAdapter* **$dataAdapter** instance of a DataAdapter
- *array* **$properties**
- *array* **$tableSchemas**

Constructor

- **Access** public

TableSchema function DatabaseSchema::seek($tableName) *[line 95]*

Function Parameters:

- *string* **$tableName** table name

seek

- **Access** public

mixed function DatabaseSchema::__get($field) *[line 67]*

Function Parameters:

- *string* **$field** name of the field

returns an object property

- **Access** public

Class DataSet

[line 34]

Represents a DataSet

- **Package** PHPRum
- **Sub-Package** DB
- **Property** int $page: the current page
- **Property** int $pageCount: the current page count
- **Property** int $pageSize: the page size
- **Property** int $cursor: record pointer
- **Property** int $count: number of records
- **Property** bool $eof: specifies if end of record
- **Property** bool $bof: specifies if beginning of record
- **Property** string $source: data source string
- **Property** string $table: source table
- **Property** array $fieldMeta: array of field meta data
- **Property** array $fields: array of fields
- **Property** array $rows: collection of records
- **Property** array $row: record data
- **Property** DataAdapter $dataAdapter: data adapter
- **Author** Darnell Shinbine

DataSet function DataSet::addNew($source, &$da, [$lock_type = null]) *[line 216]*
Function Parameters:

- *$source* **$source** data source
- *$da* **&$da** DataAdapter
- *$lock_type* **$lock_type** lock type as constant of DataSetType::OpenDynamic(), DataSetType::OpenStatic(), or DataSetType::OpenReadonly()

create new DataSet

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

bool function DataSet::bof() *[line 813]*

returns true at beggining of file

- **Access** public

void function DataSet::delete() *[line 1127]*

Delete record from DataSource

- **Access** public

bool function DataSet::eof() *[line 798]*

Returns true on end of file

- **Access** public

void function DataSet::filter($column, $operator, [$value = ''], [$case_insensitive = false]) *[line 648]*

Function Parameters:

- *string* **$column** column name to filter
- *string* **$value** field value to search for

- *string* **$operator** operation to perform
- *int* **$case_insensitive** specifies case insentitive

filter DataSet by removing all records

- **Access** public

void function DataSet::first() *[line 497]*

Moves result pointer to first record

- **Access** public

real function DataSet::getAvg($column) *[line 829]*

Function Parameters:

- *string* **$column** name of column

returns average of values in column

- **Access** public

string function DataSet::getCSVString([$enclosure = "\""], [$fdelimiter = "\t"], [$rdelimiter = "\n"], [$nullexpr = 'NULL'], [$fieldnames = true], [$n = 0]) *[line 1186]*

Function Parameters:

- *char* **$enclosure** string enclosure

- *char* **$fdelimiter** column dilemeter
- *char* **$rdelimiter** row dilemeter
- *string* **$nullexpr** null expression
- *bool* **$fieldnames** include fields names in string
- *int* **$n** no of rows

Returns a DataSet as a CSV string

- **Access** public

real function DataSet::getMax($column) *[line 868]*
Function Parameters:

- *string* **$column** name of column

returns max of values in column

- **Access** public

real function DataSet::getMin($column) *[line 938]*
Function Parameters:

- *string* **$column** name of column

returns min of values in column

- **Access** public

int function DataSet::getSum($column) *[line 1008]*

Function Parameters:

- *string* **$column** name of column

returns sum of all values in column

- **Access** public

string function DataSet::getXMLString() *[line 1273]*

Returns a DataSet as an XML string

- **Since** 3.5.5
- **Access** public

void function DataSet::insert() *[line 1075]*

Inserts record into DataSource

- **Access** public

void function DataSet::last() *[line 509]*

Moves result pointer to last record

- **Access** public

void function DataSet::move($cursor) *[line 523]*

Function Parameters:

- *int* **$cursor** new location of result pointer

Moves record pointer to specified record

- **Access** public

void function DataSet::next() *[line 473]*

Moves result pointer to next record

- **Access** public

int function DataSet::page([$page = null]) *[line 583]*

Function Parameters:

- *int* **$page** page

Moves record pointer to first record on specified page

- **Access** public

int function DataSet::pageCount() *[line 620]*

Returns the number of pages with data in a DataSet object

- **Access** public

void function DataSet::prev() *[line 485]*

Moves result pointer to previous record

- **Access** public

void function DataSet::requery() *[line 1051]*

Requery datasource

- **Access** public

bool function DataSet::seek($field, $value, [$case_insensitive = 0]) *[line 547]*

Function Parameters:

- *string* **$field** field name to search on
- *string* **$value** field value to search for
- *bool* **$case_insensitive** specifies case insentitive

sets result pointer to record based on search criteria

- **Access** public

void function DataSet::setFieldMeta($fieldMeta, $fields) *[line 327]*

Function Parameters:

- *array* **$fields** fields to add to DataSet
- *array* **$fieldMeta**

set fields

- **Access** public

void function DataSet::setFields($fields) *[line 308]*

Function Parameters:

- *array* **$fields** fields to add to DataSet

set fields

- **Access** public

void function DataSet::setRows($rows, $records) *[line 346]*

Function Parameters:

- *array* **$records** records to add to DataSet
- *array* **$rows**

set records

- **Access** public

void function DataSet::setTable($table) *[line 365]*

Function Parameters:

- *string* **$table** source table

set table

- **Access** public

void function DataSet::sort($column, [$reverse = false], [$case_insensitive = false]) *[line 412]*

Function Parameters:

- *string* **$column** column name
- *bool* **$reverse** specifies reverse sort order
- *bool* **$case_insensitive** specifies if case sensitive

sorts DataSet

- **Access** public

array function DataSet::toArray() *[line 1169]*

Returns an array containg all rows

- **Access** public

void function DataSet::update() *[line 1101]*

Update DataSource

- **Access** public

void function DataSet::updateRowData() *[line 383]*

update accessor with data from current record

- **Access** public

mixed function DataSet::__get($field) *[line 103]*

Function Parameters:

- *string* **$field** name of the field

returns an object property

- **Access** public

Class DataSetType

[line 18]

Represents a DataSet type

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

DataSetType function DataSetType::OpenDynamic() *[line 31]*

Specifies the DataSet type is OpenDynamic

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

DataSetType function DataSetType::OpenReadonly() *[line 43]*

Specifies the DataSet type is OpenReadonly

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

DataSetType function DataSetType::OpenStatic() *[line 37]*

Specifies the DataSet type is OpenStatic

- **Static**

- **Access** public

Class DirDataAdapter

[line 19]

Represents an open connection to the file system

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

DatabaseSchema function DirDataAdapter::addTableSchema(\\\&$tableSchema) *[line 73]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

creats a TableSchema object

- **Access** public

DatabaseSchema function DirDataAdapter::alterTableSchema(\\\&$tableSchema) *[line 84]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

alters a TableSchema object

- **Access** public

Transaction function DirDataAdapter::beginTransaction() *[line 296]*

creats a Transaction object

- **Access** public

DatabaseSchema function DirDataAdapter::buildSchema() *[line 46]*

builds a DataBaseSchema object

- **Access** public

bool function DirDataAdapter::close() *[line 220]*

closes an open connection

- **Access** public

void function DirDataAdapter::delete(\\\&$ds, &$ds) *[line 274]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to delete a record in the datasource

- **Access** public

DatabaseSchema function DirDataAdapter::dropTableSchema(\\\&$tableSchema) *[line 95]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

drops a TableSchema object

- **Access** public

string function DirDataAdapter::escapeString($unescaped_string) *[line 330]*
Function Parameters:

- *string* **$unescaped_string** String to escape

Returns escaped string

- **Access** public

void function DirDataAdapter::fill(\\\&$ds, &$ds) *[line 107]*
Function Parameters:

- *DataSet* **&$ds** empty DataSet object

- *DataSet* **\\\&$ds**

fetches DataSet from datasource string using source string

- **Access** public

int function DirDataAdapter::getAffectedRows() *[line 318]*

return affected rows

- **Access** public

int function DirDataAdapter::getLastInsertId() *[line 307]*

return id of last record inserted

- **Access** public

void function DirDataAdapter::insert(\\\&$ds, $ds) *[line 250]*

Function Parameters:

- *DataSet* **$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to insert a record into the datasource

- **Access** public

bool function DirDataAdapter::open() *[line 187]*

opens a connection to a filesystem resource

- **Access** public

bool function DirDataAdapter::opened() *[line 238]*

returns true if a connection to a datasource is currently open

- **Access** public

resource function DirDataAdapter::query($query, $buffer) *[line 35]*

Function Parameters:

- *string* **$query** sql query
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Access** protected

QueryBuilder function DirDataAdapter::queryBuilder() *[line 285]*

creats a QueryBuilder object

- **Access** public

void function DirDataAdapter::update(\\\&$ds, &$ds) *[line 262]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to update a record in the datasource

- **Access** public

Class MSSQLDataAdapter

[line 20]

Represents an open connection to a MSSQL database

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Tahsin Zulkarnine

MSSQLDataAdapter::$charset

string = 'utf8' *[line 32]*

Specifies the character set

- **Access** protected

DatabaseSchema function MSSQLDataAdapter::addTableSchema(\\\&$tableSchema) *[line 74]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

creats a TableSchema object

- **Access** public

DatabaseSchema function MSSQLDataAdapter::alterTableSchema(\\\&$tableSchema) *[line 186]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

alters a TableSchema object

- **Access** public

void function MSSQLDataAdapter::begin($query) *[line 719]*
Function Parameters:

- *string* **$query** sql query

Executes a sql query or procedure on the current connection

- **Access** public

MSSQLTransaction function MSSQLDataAdapter::beginTransaction() *[line 337]*

creats a Transaction object

- **Access** public

DatabaseSchema function MSSQLDataAdapter::buildSchema() *[line 213]*

builds a DataBaseSchema object

- **Access** public

bool function MSSQLDataAdapter::close() *[line 570]*

closes an open connection

- **Access** public

void function MSSQLDataAdapter::commit() *[line 755]*

commit transaction

- **Access** public

void function MSSQLDataAdapter::delete(\\\&$ds, &$ds) *[line 676]*
Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to delete a record in the datasource

- **Access** public

DatabaseSchema function MSSQLDataAdapter::dropTableSchema(\\\&$tableSchema) *[line 195]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

drops a TableSchema object

- **Access** public

string function MSSQLDataAdapter::escapeString($unescaped_string) *[line 382]*
Function Parameters:

- *string* **$unescaped_string** String to escape

Returns escaped string

- **Access** public

resource function MSSQLDataAdapter::executeInternal($query) *[line 861]*
Function Parameters:

- *string* **$query** sql query

Executes a sql query or procedure on the current connection

- **Access** protected

void function MSSQLDataAdapter::fill(\\\&$ds, &$ds) *[line 395]*
Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

fetches DataSet from database string using source string

- **Access** public

int function MSSQLDataAdapter::getAffectedRows() *[line 842]*

return affected rows

- **Access** public

void function MSSQLDataAdapter::getField($table, $fieldName) *[line 1202]*

Function Parameters:

- **$table**
- **$fieldName**

- **Access** public

int function MSSQLDataAdapter::getLastColumnIdOfTable($table, $column) *[line 822]*

Function Parameters:

- **$table**
- **$column**

return id of last record inserted for a column in table

- **Access** public

int function MSSQLDataAdapter::getLastInsertId() *[line 803]*

return id of last record inserted

- **Access** public

string function MSSQLDataAdapter::getQuery($queryBuilder) *[line 907]*

Function Parameters:

- *QueryBuilder* **$queryBuilder** QueryBuilder object

Returns SQL based on the QueryBuilder

- **Access** protected

DataSet function MSSQLDataAdapter::getTables() *[line 792]*

returns a DataSet of tables

- **Access** public

void function MSSQLDataAdapter::insert(\\\&$ds, &$ds) *[line 597]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to insert a record into the datasource

- **Access** public

bool function MSSQLDataAdapter::open() *[line 38]*

opens a connection to a mysql database

- **Access** public

bool function MSSQLDataAdapter::opened() *[line 204]*

returns true if a connection to a datasource is currently open

- **Access** public

resource function MSSQLDataAdapter::query($query, $buffer) *[line 357]*

Function Parameters:

- *string* **$query** sql query
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Access** protected

MySQLQueryBuilder function MSSQLDataAdapter::queryBuilder() *[line 346]*

creats a QueryBuilder object

- **Access** public

void function MSSQLDataAdapter::rollback() *[line 737]*

rollback transaction

- **Access** public

void function MSSQLDataAdapter::setForeignKeyChecks($set) *[line 774]*

Function Parameters:

- *bool* **$set** set/unset foreign key checks

set foreign key checks

- **Access** public

void function MSSQLDataAdapter::update(\\\&$ds, &$ds) *[line 644]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to update a record in the datasource

- **Access** public

Class MSSQLQueryBuilder

[line 21]

Represents an SQL Query

- **Package** PHPRum
- **Sub-Package** DB
- **Property** bool $empty: specifies whether to return empty result set
- **Author** Darnell Shinbine

void function MSSQLQueryBuilder::addColumn([$table = '*'], [$column = '*'], [$alias = '']) *[line 77]*
Function Parameters:

- **$table**
- **$column**
- **$alias**

add column

- **Access** protected

void function MSSQLQueryBuilder::addGroupByClause($table, $column) *[line 169]*
Function Parameters:

- **$table**
- **$column**

add group by clause

- **Access** protected

void function MSSQLQueryBuilder::addHavingClause($column, $operand, $value) *[line 156]*

Function Parameters:

- **$column**
- **$operand**
- **$value**

add having clause

- **Access** protected

void function MSSQLQueryBuilder::addJoin($type, $lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = ''])
[line 113]

Function Parameters:

- **$type**
- **$lefttable**
- **$leftcolumn**
- **$righttable**
- **$rightcolumn**
- **$alias**

add join

- **Access** protected

void function MSSQLQueryBuilder::addOrderByClause($table, $column, [$direction = 'asc']) *[line 143]*
Function Parameters:

- **$table**
- **$column**
- **$direction**

add order by clause

- **Access** protected

void function MSSQLQueryBuilder::addTable($table, [$alias = '']) *[line 101]*
Function Parameters:

- **$table**
- **$alias**

add table

- **Access** protected

void function MSSQLQueryBuilder::addValue($value) *[line 90]*
Function Parameters:

- **$value**

add value

- **Access** protected

void function MSSQLQueryBuilder::addWhereClause($table, $column, $operand, $value) *[line 129]*

Function Parameters:

- **$table**
- **$column**
- **$operand**
- **$value**

add where clause

- **Access** protected

string function MSSQLQueryBuilder::getQuery() *[line 181]*

get SQL query

- **Access** public

void function MSSQLQueryBuilder::is_date($str) *[line 493]*

Function Parameters:

- **$str**

Class MSSQLTransaction

[line 20]

Represents an open connection to a MSSQL database

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Tahsin Zulkarnine

void function MSSQLTransaction::beginTransaction() *[line 25]*

Begins a transaction

- **Access** protected

void function MSSQLTransaction::commitTransaction() *[line 44]*

Implements a commit

- **Access** protected

void function MSSQLTransaction::rollbackTransaction() *[line 35]*

Implements a rollback

- **Access** protected

Class MySQLDataAdapter

[line 19]

Represents an open connection to a MySQL database

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

MySQLDataAdapter::$charset

string = 'utf8' *[line 31]*

Specifies the character set

- **Access** protected

DatabaseSchema function MySQLDataAdapter::addTableSchema(\\\&$tableSchema) *[line 296]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

creats a TableSchema object

- **Access** public

DatabaseSchema function MySQLDataAdapter::alterTableSchema(\\\&$tableSchema) *[line 407]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

alters a TableSchema object

- **Access** public

MySQLTransaction function MySQLDataAdapter::beginTransaction() *[line 532]*

creats a Transaction object

- **Access** public

DatabaseSchema function MySQLDataAdapter::buildSchema() *[line 235]*

builds a DataBaseSchema object

- **Access** public

bool function MySQLDataAdapter::close() *[line 101]*

closes an open connection

- **Access** public

void function MySQLDataAdapter::delete(\\\&$ds, &$ds) *[line 490]*
Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to delete a record in the datasource

- **Access** public

DatabaseSchema function MySQLDataAdapter::dropTableSchema(\\\&$tableSchema) *[line 418]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

drops a TableSchema object

- **Access** public

string function MySQLDataAdapter::escapeString($unescaped_string) *[line 580]*
Function Parameters:

- *string* **$unescaped_string** String to escape

Returns escaped string

- **Access** public

void function MySQLDataAdapter::fill(\\\&$ds, &$ds) *[line 166]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

fills a DataSet object with the current DataAdapter

- **Access** public

int function MySQLDataAdapter::getAffectedRows() *[line 561]*

return affected rows

- **Access** public

int function MySQLDataAdapter::getLastInsertId() *[line 543]*

return id of last record inserted

- **Access** public

void function MySQLDataAdapter::insert(\\\&$ds, &$ds) *[line 430]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to insert a record into the datasource

- **Access** public

bool function MySQLDataAdapter::open() *[line 38]*

opens a connection to a mysql database

- **Access** public

bool function MySQLDataAdapter::opened() *[line 127]*

returns true if a connection to a datasource is currently open

- **Access** public

resource function MySQLDataAdapter::query($query, $buffer) *[line 140]*

Function Parameters:

- *string* **$query** sql query
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Access** protected

MySQLQueryBuilder function MySQLDataAdapter::queryBuilder() *[line 521]*

creats a QueryBuilder object

- **Access** public

void function MySQLDataAdapter::update(\\\&$ds, &$ds) *[line 458]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to update a record in the datasource

- **Access** public

Class MySQLiDataAdapter

[line 19]

Represents an open connection to a MySQL database using the mysqli driver

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

MySQLiDataAdapter::$charset

string = 'utf8' *[line 31]*

Specifies the character set

- **Access** protected

DatabaseSchema function MySQLiDataAdapter::addTableSchema(\\\&$tableSchema) *[line 243]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

creats a TableSchema object

- **Access** public

DatabaseSchema function MySQLiDataAdapter::alterTableSchema(\\\&$tableSchema) *[line 354]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

alters a TableSchema object

- **Access** public

MySQLTransaction function MySQLiDataAdapter::beginTransaction() *[line 479]*

creats a Transaction object

- **Access** public

DatabaseSchema function MySQLiDataAdapter::buildSchema() *[line 199]*

builds a DataBaseSchema object

- **Access** public

bool function MySQLiDataAdapter::close() *[line 84]*

closes an open connection

- **Access** public

void function MySQLiDataAdapter::delete(\\\&$ds, &$ds) *[line 437]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to delete a record in the datasource

- **Access** public

DatabaseSchema function MySQLiDataAdapter::dropTableSchema(\\\&$tableSchema) *[line 365]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

drops a TableSchema object

- **Access** public

string function MySQLiDataAdapter::escapeString($unescaped_string) *[line 527]*
Function Parameters:

- *string* **$unescaped_string** String to escape

Returns escaped string

- **Access** public

void function MySQLiDataAdapter::fill(\\\&$ds, &$ds) *[line 149]*
Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

fills a DataSet object with the current DataAdapter

- **Access** public

int function MySQLiDataAdapter::getAffectedRows() *[line 508]*

return affected rows

- **Access** public

int function MySQLiDataAdapter::getLastInsertId() *[line 490]*

return id of last record inserted

- **Access** public

void function MySQLiDataAdapter::insert(\\\&$ds, &$ds) *[line 377]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to insert a record into the datasource

- **Access** public

bool function MySQLiDataAdapter::open() *[line 38]*

opens a connection to a mysql database

- **Access** public

bool function MySQLiDataAdapter::opened() *[line 110]*

returns true if a connection to a datasource is currently open

- **Access** public

resource function MySQLiDataAdapter::query($query, $buffer) *[line 123]*

Function Parameters:

- *string* **$query** sql query
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Access** protected

MySQLQueryBuilder function MySQLiDataAdapter::queryBuilder() *[line 468]*

creats a QueryBuilder object

- **Access** public

void function MySQLiDataAdapter::update(\\\&$ds, &$ds) *[line 405]*
Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to update a record in the datasource

- **Access** public

Class MySQLiQueryBuilder

[line 21]

Represents an SQL Query

- **Package** PHPRum
- **Sub-Package** DB
- **Property** bool $empty: specifies whether to return empty result set
- **Author** Darnell Shinbine

void function MySQLiQueryBuilder::addColumn([$table = '*'], [$column = '*'], [$alias = '']) *[line 77]*
Function Parameters:

- **$table**
- **$column**
- **$alias**

add column

- **Access** protected

void function MySQLiQueryBuilder::addGroupByClause($table, $column) *[line 168]*

Function Parameters:

- **$table**
- **$column**

add group by clause

- **Access** protected

void function MySQLiQueryBuilder::addHavingClause($column, $operand, $value) *[line 155]*

Function Parameters:

- **$column**
- **$operand**
- **$value**

add having clause

- **Access** protected

void function MySQLiQueryBuilder::addJoin($type, $lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = ''])
[line 112]

Function Parameters:

- **$type**
- **$lefttable**
- **$leftcolumn**
- **$righttable**
- **$rightcolumn**
- **$alias**

add join

- **Access** protected

void function MySQLiQueryBuilder::addOrderByClause($table, $column, [$direction = 'asc']) *[line 142]*

Function Parameters:

- **$table**
- **$column**
- **$direction**

add order by clause

- **Access** protected

void function MySQLiQueryBuilder::addTable($table, [$alias = '']) *[line 100]*

Function Parameters:

- **$table**
- **$alias**

add table

- **Access** protected

void function MySQLiQueryBuilder::addValue($value) *[line 90]*

Function Parameters:

- **$value**

add value

- **Access** protected

void function MySQLiQueryBuilder::addWhereClause($table, $column, $operand, $value) *[line 128]*

Function Parameters:

- **$table**
- **$column**
- **$operand**
- **$value**

add where clause

- **Access** protected

string function MySQLiQueryBuilder::getQuery() *[line 180]*

get SQL query

- **Access** public

Class MySQLiTransaction

[line 19]

Represents an open connection to a MySQL database

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

void function MySQLiTransaction::beginTransaction() *[line 24]*

Begins a transaction

- **Access** protected

void function MySQLiTransaction::commitTransaction() *[line 43]*

Implements a commit

- **Access** protected

void function MySQLiTransaction::rollbackTransaction() *[line 34]*

Implements a rollback

- **Access** protected

Class MySQLQueryBuilder

[line 21]

Represents an SQL Query

- **Package** PHPRum
- **Sub-Package** DB
- **Property** bool $empty: specifies whether to return empty result set
- **Author** Darnell Shinbine

void function MySQLQueryBuilder::addColumn([$table = '*'], [$column = '*'], [$alias = '']) *[line 77]*

Function Parameters:

- **$table**

- **$column**
- **$alias**

add column

- **Access** protected

void function MySQLQueryBuilder::addGroupByClause($table, $column) *[line 168]*
Function Parameters:

- **$table**
- **$column**

add group by clause

- **Access** protected

void function MySQLQueryBuilder::addHavingClause($column, $operand, $value) *[line 155]*
Function Parameters:

- **$column**
- **$operand**
- **$value**

add having clause

- **Access** protected

void function MySQLQueryBuilder::addJoin($type, $lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = '']) *[line 112]*

Function Parameters:

- **$type**
- **$lefttable**
- **$leftcolumn**
- **$righttable**
- **$rightcolumn**
- **$alias**

add join

- **Access** protected

void function MySQLQueryBuilder::addOrderByClause($table, $column, [$direction = 'asc']) *[line 142]*

Function Parameters:

- **$table**
- **$column**
- **$direction**

add order by clause

- **Access** protected

void function MySQLQueryBuilder::addTable($table, [$alias = '']) *[line 100]*
Function Parameters:

- **$table**
- **$alias**

add table

- **Access** protected

void function MySQLQueryBuilder::addValue($value) *[line 90]*
Function Parameters:

- **$value**

add value

- **Access** protected

void function MySQLQueryBuilder::addWhereClause($table, $column, $operand, $value) *[line 128]*
Function Parameters:

- **$table**
- **$column**
- **$operand**

- **$value**

add where clause

- **Access** protected

string function MySQLQueryBuilder::getQuery() *[line 180]*

get SQL query

- **Access** public

Class MySQLTransaction

[line 19]

Represents an open connection to a MySQL database

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

void function MySQLTransaction::beginTransaction() *[line 24]*

Begins a transaction

- **Access** protected

void function MySQLTransaction::commitTransaction() *[line 43]*

Implements a commit

- **Access** protected

void function MySQLTransaction::rollbackTransaction() *[line 34]*

Implements a rollback

- **Access** protected

Class PDODataAdapter

[line 19]

Represents an open connection to a MySQL database using the mysqli driver

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

DatabaseSchema function PDODataAdapter::addTableSchema(\\\&$tableSchema) *[line 225]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

creats a TableSchema object

- **Access** public

DatabaseSchema function PDODataAdapter::alterTableSchema(\\\&$tableSchema) *[line 336]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

alters a TableSchema object

- **Access** public

MySQLTransaction function PDODataAdapter::beginTransaction() *[line 446]*
creats a Transaction object

- **Access** public

DatabaseSchema function PDODataAdapter::buildSchema() *[line 178]*
builds a MSSQL DataBaseSchema object

- **Access** public

bool function PDODataAdapter::close() *[line 69]*

closes an open connection

- **Access** public

void function PDODataAdapter::delete(\\\&$ds, &$ds) *[line 411]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to delete a record in the datasource

- **Access** public

DatabaseSchema function PDODataAdapter::dropTableSchema(\\\&$tableSchema) *[line 347]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

drops a TableSchema object

- **Access** public

string function PDODataAdapter::escapeString($unescaped_string) *[line 501]*

Function Parameters:

- *string* **$unescaped_string** String to escape

Returns escaped string

- **Access** public

void function PDODataAdapter::fill(\\\&$ds, &$ds) *[line 127]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

fills a DataSet object with the current DataAdapter

- **Access** public

int function PDODataAdapter::getAffectedRows() *[line 482]*

return affected rows

- **Access** public

void function PDODataAdapter::getField($table, $fieldName) *[line 674]*

Function Parameters:

- **$table**
- **$fieldName**

  - **Access** public

int function PDODataAdapter::getLastInsertId() *[line 464]*

return id of last record inserted

  - **Access** public

void function PDODataAdapter::insert(\\\&$ds, &$ds) *[line 359]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to insert a record into the datasource

  - **Access** public

bool function PDODataAdapter::open() *[line 33]*

opens a connection to a mysql database

  - **Access** public

bool function PDODataAdapter::opened() *[line 88]*

returns true if a connection to a datasource is currently open

- **Access** public

resource function PDODataAdapter::query($query, $buffer) *[line 100]*

Function Parameters:

- *string* **$query** sql query
- **$buffer**

Executes a query procedure on the current connection and return the result

- **Access** protected

MySQLQueryBuilder function PDODataAdapter::queryBuilder() *[line 434]*

creats a QueryBuilder object

- **Access** public

void function PDODataAdapter::update(\\\&$ds, &$ds) *[line 386]*

Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

attempt to update a record in the datasource

- **Access** public

Class PDOQueryBuilder

[line 21]

Represents an SQL Query

- **Package** PHPRum
- **Sub-Package** DB
- **Property** bool $empty: specifies whether to return empty result set
- **Author** Tahsin Zulkarnine

void function PDOQueryBuilder::addColumn([$table = '*'], [$column = '*'], [$alias = '']) *[line 77]*

Function Parameters:

- **$table**
- **$column**
- **$alias**

add column

- **Access** protected

void function PDOQueryBuilder::addGroupByClause($table, $column) *[line 168]*
Function Parameters:

- **$table**
- **$column**

add group by clause

- **Access** protected

void function PDOQueryBuilder::addHavingClause($column, $operand, $value) *[line 155]*
Function Parameters:

- **$column**
- **$operand**
- **$value**

add having clause

- **Access** protected

void function PDOQueryBuilder::addJoin($type, $lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = '']) *[line 112]*
Function Parameters:

- **$type**
- **$lefttable**
- **$leftcolumn**
- **$righttable**
- **$rightcolumn**
- **$alias**

add join

- **Access** protected

void function PDOQueryBuilder::addOrderByClause($table, $column, [$direction = 'asc']) *[line 142]*
Function Parameters:

- **$table**
- **$column**
- **$direction**

add order by clause

- **Access** protected

void function PDOQueryBuilder::addTable($table, [$alias = '']) *[line 100]*
Function Parameters:

- **$table**
- **$alias**

add table

- **Access** protected

void function PDOQueryBuilder::addValue($value) *[line 90]*

Function Parameters:

- **$value**

add value

- **Access** protected

void function PDOQueryBuilder::addWhereClause($table, $column, $operand, $value) *[line 128]*

Function Parameters:

- **$table**
- **$column**
- **$operand**
- **$value**

add where clause

- **Access** protected

string function PDOQueryBuilder::getMSSQLQuery() *[line 490]*

get MSSQL SQL query

- **Access** public

string function PDOQueryBuilder::getMySQLQuery() *[line 191]*

get MySQL SQL query

- **Access** public

string function PDOQueryBuilder::getQuery() *[line 180]*

get SQL query

- **Access** public

void function PDOQueryBuilder::is_date($str) *[line 802]*

Function Parameters:

- **$str**

Class PDOTransaction

[line 19]

Represents an open connection to a PDO

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

void function PDOTransaction::beginTransaction() *[line 24]*

Begins a transaction

- **Access** protected

void function PDOTransaction::commitTransaction() *[line 42]*

Implements a commit

- **Access** protected

void function PDOTransaction::rollbackTransaction() *[line 33]*

Implements a rollback

- **Access** protected

Class QueryBuilderBase

[line 20]

Represents an database query

- **Package** PHPRum
- **Sub-Package** DB
- **Property** bool $empty: specifies whether to return empty result set
- **Author** Darnell Shinbine
- **Abstract Element**

QueryBuilderBase::$dataAdapter

DataAdapter = null *[line 38]*

Contains a reference to a DataAdapter object

- **Access** protected

QueryBuilderBase::$empty

bool = false *[line 32]*

specifies whether to return empty resultset

- **Access** protected

QueryBuilderBase::$statement

string = '' *[line 26]*

statement

- **Access** protected

Constructor *void* function QueryBuilderBase::__construct(&$dataAdapter, $dataAdapter) *[line 47]*
Function Parameters:

- *DataAdapter* **$dataAdapter** instance of a DataAdapter
- *DataAdapter* **&$dataAdapter**

Constructor

- **Access** public

void function QueryBuilderBase::addColumn([$table = '*'], [$column = '*'], [$alias = '']) *[line 410]*
Function Parameters:

- **$table**
- **$column**
- **$alias**

add column

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addGroupByClause($table, $column) *[line 466]*

Function Parameters:

- **$table**
- **$column**

add group by clause

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addHavingClause($column, $operand, $value) *[line 458]*

Function Parameters:

- **$column**
- **$operand**
- **$value**

add having clause

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addJoin($type, $lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = '']) *[line 434]*

Function Parameters:

- **$type**

- **$lefttable**
- **$leftcolumn**
- **$righttable**
- **$rightcolumn**
- **$alias**

add join

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addOrderByClause($table, $column, [$direction = 'asc']) *[line 450]*
Function Parameters:

- **$table**
- **$column**
- **$direction**

add order by clause

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addTable($table, [$alias = '']) *[line 426]*
Function Parameters:

- **$table**

- **$alias**

add table

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addValue($value) *[line 418]*
Function Parameters:

- **$value**

add value

- **Abstract Element**
- **Access** protected

void function QueryBuilderBase::addWhereClause($table, $column, $operand, $value) *[line 442]*
Function Parameters:

- **$table**
- **$column**
- **$operand**
- **$value**

add where clause

- **Abstract Element**
- **Access** protected

QueryBuilder function QueryBuilderBase::delete() *[line 182]*

impliments `delete` statement

- **Access** public

QueryBuilder function QueryBuilderBase::from($table, [$alias = '']) *[line 195]*

Function Parameters:

- *string* **$table** table name
- *string* **$alias** table alias

impliments `from` statement

- **Access** public

string function QueryBuilderBase::getQuery() *[line 116]*

get query

- **Abstract Element**
- **Access** public

QueryBuilder function QueryBuilderBase::groupBy($table, $column) *[line 357]*

Function Parameters:

- *string* **$table** table name
- *string* **$column** column name

impliments `group by` statement

- **Access** public

QueryBuilder function QueryBuilderBase::having($column, $operand, $value) *[line 343]*

Function Parameters:

- *string* **$column** column name
- *string* **$operand** operation to perform
- *string* **$value** column value

impliments `having` statement

- **Access** public

QueryBuilder function QueryBuilderBase::innerJoin($lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = '']) *[line 263]*

Function Parameters:

- *string* **$lefttable** left table name
- *string* **$leftcolumn** left column name

- *string* **$righttable** right table name
- *string* **$rightcolumn** right column name
- *string* **$alias** left table alias

impliments `inner join` statement

- **Access** public

QueryBuilder function QueryBuilderBase::insertInto($table, $columns) *[line 141]*
Function Parameters:

- *string* **$table** table name
- *array* **$columns** array of columns

impliments `insert into` statement

- **Access** public

QueryBuilder function QueryBuilderBase::leftJoin($lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = ''])
[line 280]
Function Parameters:

- *string* **$lefttable** left table name
- *string* **$leftcolumn** left column name
- *string* **$righttable** right table name
- *string* **$rightcolumn** right column name
- *string* **$alias** left table alias

impliments `left join` statement

- **Access** public

DataSet function QueryBuilderBase::openDataSet([$lock_type = null]) *[line 105]*
Function Parameters:

- *DataSetType* **$lock_type** lock type as constant of DataSetType::OpenDynamic(), DataSetType::OpenStatic(), or DataSetType::OpenReadonly()

open a DataSet

- **Access** public

QueryBuilder function QueryBuilderBase::orderBy($table, $column, [$direction = 'asc']) *[line 328]*
Function Parameters:

- *string* **$table** table name
- *string* **$column** column name
- *string* **$direction** order by direction

impliments `order by` statement

- **Access** public

QueryBuilder function QueryBuilderBase::rightJoin($lefttable, $leftcolumn, $righttable, $rightcolumn, [$alias = '']) *[line 297]*

Function Parameters:

- *string* **$lefttable** left table name
- *string* **$leftcolumn** left column name
- *string* **$righttable** right table name
- *string* **$rightcolumn** right column name
- *string* **$alias** left table alias

impliments `right join` statement

- **Access** public

void function QueryBuilderBase::runQuery() *[line 93]*

run query

- **Access** public

QueryBuilder function QueryBuilderBase::select([$table = '*'], [$column = '*'], [$alias = '']) *[line 127]*

Function Parameters:

- *string* **$table** table name
- *string* **$column** column name
- *string* **$alias** column alias

impliments `select` statement

- **Access** public

QueryBuilder function QueryBuilderBase::set($table, $column, $value) *[line 225]*

Function Parameters:

- *string* **$table** table name
- *string* **$column** column name
- *string* **$value** column value

impliments `set` statement

- **Access** public

QueryBuilder function QueryBuilderBase::setColumns($table, $columns, $values) *[line 241]*

Function Parameters:

- *string* **$table** table name
- *array* **$columns** column names
- *array* **$values** values

impliments `set` statement

- **Access** public

QueryBuilder function QueryBuilderBase::truncate($table) *[line 170]*

Function Parameters:

- *string* **$table** table name

impliments `truncate` statement

- **Access** public

QueryBuilder function QueryBuilderBase::update($table) *[line 157]*
Function Parameters:

- *string* **$table** table name

impliments `update` statement

- **Access** public

QueryBuilder function QueryBuilderBase::values($values) *[line 208]*
Function Parameters:

- *array* **$values** array of column values

impliments `values` statement

- **Access** public

QueryBuilder function QueryBuilderBase::where($table, $column, $operand, $value) *[line 313]*

Function Parameters:

- *string* **$table** table name
- *string* **$column** column name
- *string* **$operand** operation to perform
- *string* **$value** column value

impliments `where` statement

- **Access** public

Class QueryException

[line 19]

Represents a QueryBuilder Exception

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

Class SchemaException

[line 19]

Represents a Schema Exception

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

Class TableSchema

[line 24]

Represents a table schema

- **Package** PHPRum
- **Sub-Package** DB
- **Property** string $name: table name
- **Property** string $primaryKey: primary key
- **Property** array $foreignKeys: array of foreign keys
- **Property** array $columnSchemas: array of column schemas
- **Property** array $columnNames: array of column names
- **Author** Darnell Shinbine

Constructor *void* function TableSchema::__construct($properties, $foreignKeys, $columnSchemas) *[line 59]*

Function Parameters:

- *array* **$properties** initial properties

- *array* **$foreignKeys** array of foreign keys
- *array* **$columnSchemas** array of column schemas

Constructor

- **Access** public

TableSchema function TableSchema::seek($columnName, $tableName) *[line 119]*
Function Parameters:

- *string* **$tableName** table name
- **$columnName**

seek

- **Access** public

mixed function TableSchema::__get($field) *[line 91]*
Function Parameters:

- *string* **$field** name of the field

returns an object property

- **Access** public

Class TextDataAdapter

[line 19]

Represents an open connection to a tab or comma delimited text file

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

DatabaseSchema function TextDataAdapter::addTableSchema(\\\&$tableSchema) *[line 73]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

creats a TableSchema object

- **Access** public

DatabaseSchema function TextDataAdapter::alterTableSchema(\\\&$tableSchema) *[line 84]*

Function Parameters:

- *TableSchema* **\\\&$tableSchema**

alters a TableSchema object

- **Access** public

Transaction function TextDataAdapter::beginTransaction() *[line 332]*

creats a Transaction object

- **Access** public

DatabaseSchema function TextDataAdapter::buildSchema() *[line 46]*

builds a DataBaseSchema object

- **Access** public

bool function TextDataAdapter::close() *[line 210]*

closes an open connection

- **Access** public

void function TextDataAdapter::delete(\\\&$ds, &$ds) *[line 298]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to delete a record in the datasource

- **Access** public

DatabaseSchema function TextDataAdapter::dropTableSchema(\\\&$tableSchema) *[line 95]*
Function Parameters:

- *TableSchema* **\\\&$tableSchema**

drops a TableSchema object

- **Access** public

string function TextDataAdapter::escapeString($unescaped_string) *[line 366]*
Function Parameters:

- *string* **$unescaped_string** String to escape

Returns escaped string

- **Access** public

void function TextDataAdapter::fill(\\\&$ds, &$ds) *[line 107]*
Function Parameters:

- *DataSet* **&$ds** empty DataSet object
- *DataSet* **\\\&$ds**

fetches DataSet from datasource string using source string

- **Access** public

int function TextDataAdapter::getAffectedRows() *[line 354]*

return affected rows

- **Access** public

int function TextDataAdapter::getLastInsertId() *[line 343]*

return id of last record inserted

- **Access** public

void function TextDataAdapter::insert(\\\&$ds, &$ds) *[line 243]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to insert a record into the datasource

- **Access** public

bool function TextDataAdapter::open() *[line 159]*

opens a connection to a text file

- **Access** public

bool function TextDataAdapter::opened() *[line 231]*

returns true if a connection to a datasource is currently open

- **Access** public

resource function TextDataAdapter::query($query, $buffer) *[line 35]*

Function Parameters:

- *string* **$query** sql query
- *bool* **$buffer** buffer resultset

Executes a query procedure on the current connection and return the result

- **Access** protected

QueryBuilder function TextDataAdapter::queryBuilder() *[line 321]*

creats a QueryBuilder object

- **Access** public

void function TextDataAdapter::update(\\\&$ds, &$ds) *[line 277]*

Function Parameters:

- *DataSet* **&$ds** reference to a DataSet
- *DataSet* **\\\&$ds**

attempt to update a record in the datasource

- **Access** public

Class TransactionBase

[line 18]

Represents a transaction

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine
- **Abstract Element**

TransactionBase::$dataAdapter

DataAdapter = null *[line 24]*

Contains a reference to a DataAdapter object

- **Access** protected

Constructor *void* function TransactionBase::__construct(&$dataAdapter, $dataAdapter) *[line 39]*

Function Parameters:

- *DataAdapter* **$dataAdapter** instance of a DataAdapter
- *DataAdapter* **&$dataAdapter**

Constructor

- **Access** public

void function TransactionBase::begin() *[line 49]*

Begins a transaction

- **Access** public

void function TransactionBase::beginTransaction() *[line 100]*

Begins a transaction

- **Abstract Element**
- **Access** protected

void function TransactionBase::commit() *[line 83]*

Implements a commit

- **Access** public

void function TransactionBase::commitTransaction() *[line 112]*

Implements a commit

- **Abstract Element**
- **Access** protected

void function TransactionBase::rollback() *[line 66]*

Implements a rollback

- **Access** public

void function TransactionBase::rollbackTransaction() *[line 106]*

Implements a rollback

- **Abstract Element**
- **Access** protected

Class TransactionException

[line 19]

Represents a Transaction Exception

- **Package** PHPRum
- **Sub-Package** DB
- **Author** Darnell Shinbine

deploy.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

deploymentbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

sshclientbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

get.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Deploy

[line 19]

deplomyent script

- **Package** PHPRum
- **Sub-Package** Deploy
- **Author** Darnell Shinbine

void function Deploy::execute($argc, $argv) *[line 29]*

Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

execute the application

- **Access** protected

array function Deploy::getOptions($argc, $argv) *[line 87]*

Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

retrieve command line options

- **Access** protected

void function Deploy::handleError($errno, $errstr, $errfile, $errline) *[line 134]*

Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description
- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Access** protected

void function Deploy::handleException(\$e, $e) *[line 121]*

Function Parameters:

- *\Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden to provide error handling.

- **Access** protected

Class DeploymentBase

[line 22]

Provides base functionality for deploying applications to remote servers.

The DeploymentBase exposes 2 protected properties, these properties must be defined in the sub class

- **Package** PHPRum
- **Sub-Package** Deploy
- **Property** string $home_path: Specifies the home path on the remote server
- **Property** string $release_path: Specifies the release path on the remote server
- **Author** Darnell Shinbine
- **Abstract Element**

DeploymentBase::$home_path

string = *[line 28]*

Specifies the home path on the remote server

- **Access** protected

DeploymentBase::$release_path

string = *[line 34]*

Specifies the release path on the remote server

- **Access** protected

Constructor *void* function DeploymentBase::__construct() *[line 42]*

Constructor

- **Access** public

Class Get

[line 19]

deplomyent script

- **Package** PHPRum

- **Sub-Package** Deploy
- **Author** Darnell Shinbine

void function Get::execute($argc, $argv) *[line 35]*
Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

execute the application

- **Access** protected

array function Get::getOptions($argc, $argv) *[line 143]*
Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

retrieve command line options

- **Access** protected

void function Get::handleError($errno, $errstr, $errfile, $errline) *[line 190]*
Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description

- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Access** protected

void function Get::handleException(\$e, $e) *[line 177]*
Function Parameters:

- *\Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden to provide error handling.

- **Access** protected

Class SSHClientBase

[line 24]

Provides base functionality for connecting via SSH

The DeploymentBase exposes 4 protected properties, these properties must be defined in the sub class

- **Package** PHPRum
- **Sub-Package** Deploy
- **Property** string $server: Specifies the server name
- **Property** string $port: Specifies the server port
- **Property** string $username: Specifies the login username
- **Property** string $password: Specifies the login password
- **Author** Darnell Shinbine
- **Abstract Element**

SSHClientBase::$password

string = '' *[line 48]*

Specifies the login password

- **Access** protected

SSHClientBase::$port

int = 22 *[line 36]*

Specifies the server port

- **Access** protected

SSHClientBase::$server

string = 'localhost' *[line 30]*

Specifies the server name

- **Access** protected

SSHClientBase::$username

string = 'root' *[line 42]*

Specifies the login username

- **Access** protected

void function SSHClientBase::exec() *[line 94]*

execute deployment script

- **Access** public

void function SSHClientBase::put($local_path, $remote_path, $repository_path, $home_path) *[line 76]*

Function Parameters:

- *string* **$repository_path** local path
- *string* **$home_path** remote path
- **$local_path**
- **$remote_path**

put file on remote server

- **Access** protected

void function SSHClientBase::run($cmd) *[line 63]*

Function Parameters:

- *string* **$cmd** command to run on remote server

execute on remote server

- **Access** protected

dbtranslator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filetranslator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

translatorbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

translatorexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class DBTranslator

[line 18]

Represents a message Translator using a Database

- **Package** PHPRum
- **Sub-Package** I18N
- **Author** Darnell Shinbine

Constructor *void* function DBTranslator::__construct([$table = ''], [$dsn = '']) *[line 39]*

Function Parameters:

- *string* **$table** table name
- *string* **$dsn** database connection string

Constructor

- **Access** public

void function DBTranslator::get($stringId, [$default = '']) *[line 65]*
Function Parameters:

- *string* **$stringId** string id to translate
- *string* **$default** string if not found

get

- **Access** public

Class FileTranslator

[line 18]

Represents a message Translator using a File

- **Package** PHPRum
- **Sub-Package** I18N
- **Author** Darnell Shinbine

Constructor *void* function FileTranslator::__construct([$file = ''], $flie) *[line 38]*
Function Parameters:

- *string* **$flie** file name

- **$file**

Constructor

- **Access** public

void function FileTranslator::get($stringId, [$default = '']) *[line 58]*
Function Parameters:

- *string* **$stringId** string id to translate
- *string* **$default** string if not found

get

- **Access** public

Class TranslatorBase

[line 21]

Represents a message Translator

- **Package** PHPRum
- **Sub-Package** I18N

- **Property** string $lang: specifies the language
- **Property** string $charset: specifies character set
- **Author** Darnell Shinbine
- **Abstract Element**

Constructor *void* function TranslatorBase::__construct() *[line 41]*

Constructor

- **Access** public

void function TranslatorBase::get($stringId, [$default = '']) *[line 119]*

Function Parameters:

- *string* **$stringId** string id to translate
- *string* **$default** string if not found

get

- **Abstract Element**
- **Access** public

void function TranslatorBase::setCharset($charset) *[line 105]*

Function Parameters:

- *string* **$charset** character set

set character set

- **Access** public

void function TranslatorBase::setLang($lang) *[line 93]*

Function Parameters:

- *string* **$lang** language

set language

- **Access** public

Class TranslatorException

[line 19]

Represents a Translator Exception

- **Package** PHPRum
- **Sub-Package** I18N
- **Author** Darnell Shinbine

directorynotfoundexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

directorynotwritableexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

file.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

fileloadexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filenotfoundexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filenotwritableexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filestream.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filesystem.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

folder.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

ioexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

socketstream.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

streambase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

httprequest.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

httpresponse.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

session.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class DirectoryNotFoundException

[line 18]

Represents a Directory Not Found Exception

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

Class DirectoryNotWritableException

[line 18]

Represents a Directory Not Writable Exception

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

Class File

[line 25]

Represents a file on the file system

- **Package** PHPRum
- **Sub-Package** IO
- **Property** string $name: name of file
- **Property** string $path: path to file
- **Property** string $size: size of file
- **Property** int $modified: specifies when file last modified
- **Property** int $accessed: specifies when file last accessed
- **Property** int $created: specifies when file created
- **Author** Darnell Shinbine

Constructor *void* function File::__construct($filename) *[line 109]*

Function Parameters:

- *string* **$filename** Name of File

Constructor

- **Access** public

void function File::changeFileMode([$mode = 0755]) *[line 211]*
Function Parameters:

- *string* **$mode** chmod

change file mode

- **Access** public

void function File::copy([$dest = '']) *[line 133]*
Function Parameters:

- *string* **$dest** path to destination

copy file to destination

- **Access** public

void function File::delete() *[line 171]*
delete file

- **Access** public

void function File::move($dest) *[line 152]*
Function Parameters:

- *string* **$dest** path to destination

move file to destination

- **Access** public

string function File::readLastLine($_file) *[line 230]*
Function Parameters:

- *string* **$_file** path to file to read

returns the last line of text file **without** reading the entire file

- **Access** public

void function File::rename($newname) *[line 191]*
Function Parameters:

- *string* **$newname** new name

rename file

- **Access** public

Class FileLoadException

[line 18]

Represents a File Load Exception

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

Class FileNotFoundException

[line 18]

Represents a File Not Found Exception

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

Class FileNotWritableException

[line 18]

Represents a File Not Writable Exception

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

Class FileStream

[line 18]

Provides a way to stream data to and from a file

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

bool function FileStream::open($path, [$mode = 'ab+']) *[line 53]*

Function Parameters:

- *string* **$path** path to file
- *int* **$mode** a = read/write, w = write, r = readonly

opens a text or binary file for input output

- **Access** public

Class FileSystem

[line 18]

Provides access to the file system

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

void function FileSystem::changeFileMode($glob, [$mode = 0755]) *[line 212]*

Function Parameters:

- *string* **$glob** if string, must be a file name (foo.txt), if glob pattern (*.txt), or directory name.
- **$mode**

Change the file mode (chmod)

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

void function FileSystem::copy($src, [$dest = '']) *[line 27]*

Function Parameters:

- *string* **$src** Source path

- *string* **$dest** Destination path

Recursively copy a file or folder and its contents

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

void function FileSystem::delete($glob) *[line 157]*
Function Parameters:

- *string* **$glob** if string, must be a file name (foo.txt), if glob pattern (*.txt), or directory name.

Recursively delete a file or folder and its contents using wild cards

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

void function FileSystem::move($src, $dest) *[line 100]*
Function Parameters:

- *string* **$src** Source path
- *string* **$dest** Destination path

Recursively move a file or folder and its contents

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

void function FileSystem::removeDirectory($directory) *[line 242]*

Function Parameters:

- *string* **$directory** Path of directory

Recursively delete a dir and its contents

- **Static**

void function FileSystem::rename($old, $new) *[line 185]*

Function Parameters:

- *string* **$old** Old filename
- *string* **$new** New filename

Rename a file or folder to another location

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

Class Folder

[line 26]

Represents a folder on the file system

- **Package** PHPRum
- **Sub-Package** IO
- **Property** string $name: name of folder
- **Property** string $path: path to folder
- **Property** array $files: list of files
- **Property** array $folders: list of sub folders
- **Property** int $modified: specifies when folder last modified
- **Property** int $accessed: specifies when folder last accessed
- **Property** int $created: specifies when folder created
- **Author** Darnell Shinbine

Constructor *void* function Folder::__construct($dirname) *[line 120]*
Function Parameters:

- *string* **$dirname** Name of Directory

Constructor

- **Access** public

void function Folder::changeFileMode([$mode = 0755]) *[line 241]*
Function Parameters:

- *string* **$mode** chmod

change file mode

- **Access** public

void function Folder::copy([$dest = '']) *[line 163]*

Function Parameters:

- *string* **$dest** path to destination

copy folder to destination

- **Access** public

void function Folder::delete() *[line 201]*

delete folder

- **Access** public

void function Folder::move($dest) *[line 182]*

Function Parameters:

- *string* **$dest** path to destination

move folder to destination

- **Access** public

void function Folder::rename($newname) *[line 221]*

Function Parameters:

- *string* **$newname** new name

rename folder

- **Access** public

Class HTTPRequest

[line 23]

Provides a way to access HTTP Request information from the client

- **Package** PHPRum
- **Sub-Package** IO
- **Property** array $post: HTTP POST request variables
- **Property** array $get: HTTP GET request variables
- **Property** array $request: HTTP GET/POST request variables
- **Property** array $cookies: HTTP Cookies
- **Author** Darnell Shinbine

HTTPRequest::$cookie

array = null *[line 41]*

var to hold cookie vars

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

HTTPRequest::$get

array = null *[line 29]*

var to hold get vars

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

HTTPRequest::$post

array = null *[line 35]*

var to hold post vars

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

HTTPRequest::$request

array = null *[line 47]*

var to hold request vars

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

Constructor *void* function HTTPRequest::__construct() *[line 57]*

Constructor

Read values from request, clean up variables, and merge get and post requests.

- **Access** public

string function HTTPRequest::getQueryString() *[line 206]*

return the URL

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

string function HTTPRequest::getRawPostData() *[line 226]*

return raw post data

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

array function HTTPRequest::getRequestHeaders() *[line 256]*

return request headers

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

string function HTTPRequest::getRequestMethod() *[line 246]*

return the request method

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

string function HTTPRequest::getRequestReferer() *[line 236]*

return the request method

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

string function HTTPRequest::getRequestURI() *[line 216]*

return the URI

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

Class HTTPResponse

[line 23]

Pprovides a way to send HTTP response information to the client

- **Package** PHPRum
- **Sub-Package** IO
- **Property** int $statusCode: code HTTP response status code
- **Property** string $status: HTTP response status
- **Property** array $headers: HTTP response headers
- **Property** string $contents: HTTP response
- **Author** Darnell Shinbine

Constructor *void* function HTTPResponse::__construct() *[line 55]*

Constructor

- **Access** public

void function HTTPResponse::addHeader($header) *[line 140]*

Function Parameters:

- *string* **$header** header information

adds header to output stream

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

void function HTTPResponse::clear() *[line 195]*

clear all content from output buffer

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

void function HTTPResponse::end() *[line 213]*

send response to client and stop execution

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

void function HTTPResponse::flush() *[line 177]*

send all content to client

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

string function HTTPResponse::getResponseContent() *[line 233]*

return response content

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

StringCollection function HTTPResponse::getResponseHeaders() *[line 286]*

return response headers

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

void function HTTPResponse::redirect($location, [$terminate = true]) *[line 270]*

Function Parameters:

- *string* **$location** the URL to redirect to
- *bool* **$terminate** specifies whether to stop execution

redirect to new URL

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

void function HTTPResponse::setCookie($name, [$value = ''], [$expires = 0], [$path = '/'], [$domain = null], [$secure = false]) *[line 249]*

Function Parameters:

- *string* **$name** the name of the cookie
- *string* **$value** the value of the cookie
- *int* **$expires** timestamp that the cookie will expire on
- *string* **$path** the path on the server that the cookie will be available on

- *string* **$domain** the domain that the cookie is available on
- *bool* **$secure** requires the cookie be sent over secure HTTPS

set cookie

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

void function HTTPResponse::write($content) *[line 159]*

Function Parameters:

- *string* **$content** information to send

sends information to an output stream

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

Class IOException

[line 19]

Represents an I/O Exception

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

Class Session

[line 21]

Provides a way to access the persistant session

- **Package** PHPRum
- **Sub-Package** IO
- **Property** string $sessionId: Session Id
- **Property** string $sessionName: Session Name
- **Author** Darnell Shinbine

Constructor *void* function Session::__construct() *[line 47]*

Read values from request, clean up variables, and merge get and post requests.

- **Access** public

void function Session::close() *[line 254]*

close the session

- **Access** public

void function Session::destroy() *[line 271]*

destroy the session

- **Access** public

array function Session::getSessionData() *[line 327]*

return session array

- **Access** public

void function Session::read($sessId) *[line 211]*

Function Parameters:

- *string* **$sessId** new session id

initialize and read session data once started (implicit)

- **Access** public

void function Session::regenerateId([$delete = false]) *[line 291]*

Function Parameters:

- *bool* **$delete** specifies whether to delete the old associated session file or not

regenerate the session id

- **Access** public

void function Session::removeAll() *[line 316]*

remove all session keys

- **Access** public

void function Session::start([$sessId = null]) *[line 181]*

Function Parameters:

- *string* **$sessId** new session id

This method will start a new session or continue an existing session.

You can optionally set the session id

- **Access** public

void function Session::write() *[line 230]*

write and close the session

- **Access** public

Class SocketStream

[line 18]

Provides a way to stream data to and from a socket

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine

bool function SocketStream::open($host, [$port = 80], [$timeout = 30]) *[line 41]*

Function Parameters:

- *string* **$host** host
- *int* **$port** port
- *int* **$timeout** timeout is seconds

opens a socket for input output

- **Access** public

Class StreamBase

[line 18]

Provides base functionality for streaming

- **Package** PHPRum
- **Sub-Package** IO
- **Author** Darnell Shinbine
- **Abstract Element**

StreamBase::$handle

resource = null *[line 24]*

handle to file

- **Access** protected

void function StreamBase::close() *[line 44]*

closes an open connection

- **Access** public

bool function StreamBase::eos() *[line 312]*

Returns whether the current position is at the end of the stream or not

- **Access** public

void function StreamBase::flush() *[line 330]*

Sends the contents of the Stream buffer to the associated underlying object

- **Access** public

int function StreamBase::getPosition() *[line 243]*

Returns the current position from the beginning of a Stream object

- **Access** public

int function StreamBase::length() *[line 192]*

Returns the size of the data

- **Access** public

string function StreamBase::read([$bytes = 0]) *[line 63]*

Function Parameters:

- *int* **$bytes** no. of bytes to read

Reads the entire stream or a specified number of bytes from a binary Stream object

- **Access** public

string function StreamBase::readln($bytes) *[line 103]*

Function Parameters:

- *int* **$bytes** no. of bytes to read

Reads one line from a TextStream file and returns the result

- **Access** public

void function StreamBase::setBOS() *[line 270]*

Sets the current position to be the end of the stream (EOS)

- **Access** public

void function StreamBase::setEOS() *[line 291]*

Sets the current position to be the end of the stream (EOS)

- **Access** public

void function StreamBase::setPosition($offset) *[line 215]*

Function Parameters:

- *int* **$offset** number of bytes from beggining of file

Sets the current position from the beginning of a Stream object

- **Access** public

int function StreamBase::write($raw) *[line 132]*
Function Parameters:

- *string* **$raw** raw binary data or text string

Writes binary data to a binary Stream object

- **Access** public

int function StreamBase::writeln($line, $string) *[line 167]*
Function Parameters:

- *string* **$string** string to write
- **$line**

Writes a specified number of new-line character to a TextStream file

- **Access** public

dblogger.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filelogger.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

loggerexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class DBLogger

[line 22]

Provides access to read/write log messages

- **Package** PHPRum
- **Sub-Package** Logger
- **Property** string $path: path to log file
- **Property** string $file: name of log file
- **Property** int $maxFileSize: max size of log file in KB, 0 for infinite
- **Author** Darnell Shinbine

Constructor *void* function DBLogger::__construct([$table = __LOGS_TABLENAME__], [$dsn = '']) *[line 43]*

Function Parameters:

- *string* **$table** table name
- *string* **$dsn** database connection string

Constructor

- **Access** public

void function DBLogger::flush($category) *[line 133]*

Function Parameters:

- *string* **$category** message category

This method flushes all messages

- **Access** public

void function DBLogger::log($message, $category) *[line 65]*

Function Parameters:

- *string* **$message** message to log
- *string* **$category** message category

This method writes a log message to memory

- **Access** public

array function DBLogger::logs($category) *[line 117]*

Function Parameters:

- *string* **$category** message category

This method retrieves all log messages

- **Access** public

Class FileLogger

[line 22]

Provides access to read/write log messages

- **Package** PHPRum
- **Sub-Package** Logger
- **Property** string $path: path to log file
- **Property** string $file: name of log file
- **Property** int $maxFileSize: max size of log file in KB, 0 for infinite
- **Author** Darnell Shinbine

Constructor *void* function FileLogger::__construct([$path = __LOG_PATH__], [$maxFileSize = 0]) *[line 44]*

Function Parameters:

- *string* **$path** path of log files
- *string* **$maxFileSize** max file size

Constructor

- **Access** public

void function FileLogger::flush($category) *[line 166]*
Function Parameters:

- *string* **$category** message category

This method flushes all messages

- **Access** public

void function FileLogger::log($message, $category) *[line 107]*
Function Parameters:

- *string* **$message** message to log
- *string* **$category** message category

This method writes a log message to memory

- **Access** public

array function FileLogger::logs($category) *[line 138]*
Function Parameters:

- *string* **$category** message category

This method retrieves all log messages

- **Access** public

Class LoggerException

[line 19]

Represents a Logger Exception

- **Package** PHPRum
- **Sub-Package** Logger
- **Author** Darnell Shinbine

imailclient.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

mailmessage.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

phpmailclient.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

sendmailclient.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

smtpclient.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

require_once **__LIB_PATH__.'/smtp/class.smtp.php'** *[line 13]*

include lib

Class IMailClient

[line 18]

Provides the interface for Mail Clients

- **Package** PHPRum
- **Sub-Package** Mail
- **Author** Darnell Shinbine

void function IMailClient::send($message) *[line 26]*

Function Parameters:

- *MailMessage* **$message** message to send

sends a single email to all addresses in message

- **Access** public

Class MailMessage

[line 29]

Represents an E-Mail Message

- **Package** PHPRum
- **Sub-Package** Mail
- **Property** string $to: Recipient email address
- **Property** string $from: Sender's email address
- **Property** string $subject: Message subject
- **Property** string $body: Message body
- **Property** string $charset: Message character set
- **Property** string $contentType: Message content-type
- **Property** string $encoding: Message encoding
- **Property** bool $html: Specifies whether to send as html
- **Property** array $cc: array of email addresses to be copied
- **Property** array $bcc: array of email addresses to be blind copied

Constructor *void* function MailMessage::__construct() *[line 109]*

Constructor

- **Access** public

void function MailMessage::addBcc($bcc) *[line 340]*
Function Parameters:

- *string* **$bcc** email address

add email address to blind copy

- **Access** public

void function MailMessage::addCc($cc) *[line 321]*
Function Parameters:

- *string* **$cc** email address

add email address to copy

- **Access** public

void function MailMessage::attach($file, [$mimetype = 'text/plain'], [$filename = ''], [$inline = false], [$alternate = false], [$encoding = '7bit'], $disposition) *[line 364]*
Function Parameters:

- *string* **$file** file path or content
- *string* **$mimetype** attachment mimetype
- *string* **$filename** optional filename (auto detected if adding by path)

- *string* **$disposition** specifies whether the attachment disposition is inline, default is false
- *string* **$alternate** specifies whether the attachment is part of a multipart/alternate message, default is false
- *string* **$encoding** optional encoding, default is 7bit
- **$inline**

add an attachment to the mail message

- **Access** public

string function MailMessage::getContent() *[line 424]*

get message content

- **Access** public

string function MailMessage::getHeaders() *[line 392]*

get message headers

- **Access** public

void function MailMessage::setBody($body) *[line 273]*

Function Parameters:

- *string* **$body** message body

set message body

- **Access** public

void function MailMessage::setCharset($charset) *[line 285]*
Function Parameters:

- *string* **$charset** character set

set character set

- **Access** public

void function MailMessage::setContentType($contentType) *[line 297]*
Function Parameters:

- *string* **$contentType** content type

set content type

- **Access** public

void function MailMessage::setEncoding($encoding) *[line 309]*
Function Parameters:

- *string* **$encoding** encoding

set character encoding

- **Access** public

void function MailMessage::setFrom($from) *[line 242]*
Function Parameters:

- *string* **$from** sender's email address

set sender's email address

- **Access** public

void function MailMessage::setSubject($subject) *[line 261]*
Function Parameters:

- *string* **$subject** message subject

set message subject

- **Access** public

void function MailMessage::setTo($to) *[line 223]*
Function Parameters:

- *string* **$to** recipient email address

set recipient email address

- **Access** public

Class PHPMailClient

[line 18]

Handles email sending via PHP mail() function

- **Package** PHPRum
- **Sub-Package** Mail
- **Author** Darnell Shinbine

void function PHPMailClient::send($message) *[line 26]*

Function Parameters:

- *MailMessage* **$message** message to send

sends a single email to all addresses in message

- **Access** public

Class SendMailClient

[line 20]

Handles email sending via the $sendmail program

- **Package** PHPRum
- **Sub-Package** Mail
- **Property** string $path: specifies the path to the $sendmail program
- **Author** Darnell Shinbine

Constructor *void* function SendMailClient::__construct([$path = '']) *[line 33]*

Function Parameters:

- *string* **$path** specifies the path to the $sendmail program

Constructor

- **Access** public

void function SendMailClient::send($message) *[line 86]*

Function Parameters:

- *MailMessage* **$message** message to send

sends a single email to all addresses in message

- **Access** public

Class SMTPClient

[line 30]

Handles email sending via smtp client

- **Package** PHPRum
- **Sub-Package** Mail
- **Property** string $host: specifies the smtp host
- **Property** int $port: specifies the smtp port
- **Property** int $timeout: specifies the smtp connection timeout
- **Property** bool $keepAlive: specifies whether to keep the connection alive
- **Property** string $helo: specifies the smtp helo message
- **Property** string $authUsername: specifies the smtp authentication username
- **Property** string $authPassword: specifies the smtp authentication password
- **Author** Darnell Shinbine

Constructor *void* function SMTPClient::__construct([$host = '']) *[line 85]*

Function Parameters:

- *string* **$host** specifies the smtp host

Constructor

- **Access** public

void function SMTPClient::close() *[line 239]*

Closes the active SMTP session if one exists.

- **Access** public

void function SMTPClient::open() *[line 184]*

Initiates a connection to an SMTP server operation failed.

- **Access** public

void function SMTPClient::send($message) *[line 259]*

Function Parameters:

- *MailMessage* **$message** message to send

sends a single email to all addresses in message

- **Access** public

makebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

controller.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formcontroller.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcontroller.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

jsonwebservice.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

listcontroller.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

model.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagecontroller.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

restwebservice.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

soapwebservice.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Controller

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

Controller::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

void function Controller::make($target, [$options = array()]) *[line 33]*

Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class FormController

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

FormController::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

FormController::DefaultBaseController

= '\\ApplicationController' *[line 35]*

DefaultBaseController

FormController::ModelNamespace

= '\\Models' *[line 29]*

Default namespace

void function FormController::make($target, [$options = array()]) *[line 45]*

Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class GridViewController

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

GridViewController::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

GridViewController::DefaultBaseController

= '\\ApplicationController' *[line 34]*

DefaultBaseController

GridViewController::ModelNamespace

= '\\Models' *[line 28]*

Default namespace

void function GridViewController::make($target, [$options = array()]) *[line 44]*

Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class JSONWebService

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

JSONWebService::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

void function JSONWebService::make($target, [$options = array()]) *[line 33]*
Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class ListController

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

ListController::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

ListController::DefaultBaseController

= '\\ApplicationController' *[line 34]*

DefaultBaseController

ListController::ModelNamespace

= '\\Models' *[line 28]*

Default namespace

void function ListController::make($target, [$options = array()]) *[line 44]*
Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class MakeBase

[line 18]

Provides the interface for Make scripts

- **Package** PHPRum
- **Sub-Package** Make

- **Author** Darnell Shinbine
- **Abstract Element**

vod function MakeBase::export($path, $contents) *[line 37]*
Function Parameters:

- *string* **$path** path
- *string* **$contents** contents

export

- **Access** protected

void function MakeBase::make($target, [$options = array()]) *[line 27]*
Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Abstract Element**
- **Access** public

Class Model

[line 17]

Provides functionality to generate a model file

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

Model::ModelNamespace

= '\\Models' *[line 23]*

Default namespace

void function Model::make($target, [$options = array()]) *[line 33]*

Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class PageController

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

PageController::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

PageController::DefaultBaseController

= '\\ApplicationController' *[line 29]*

DefaultBaseController

void function PageController::make($target, [$options = array()]) *[line 39]*
Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class RESTWebService

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

RESTWebService::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

void function RESTWebService::make($target, [$options = array()]) *[line 33]*
Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

Class SOAPWebService

[line 17]

Provides functionality to generate controller/view files

- **Package** PHPRum
- **Sub-Package** Make
- **Author** Darnell Shinbine

SOAPWebService::ControllerNamespace

= '\\Controllers' *[line 23]*

Default namespace

void function SOAPWebService::make($target, [$options = array()]) *[line 33]*
Function Parameters:

- *string* **$target** target
- *array* **$options** options

make

- **Access** public

migrate.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

migrationbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

migrationcompare.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

migrations.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

test.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Migrate

[line 19]

deplomyent script

- **Package** PHPRum
- **Sub-Package** Migrate
- **Author** Darnell Shinbine

void function Migrate::execute($argc, $argv) *[line 29]*

Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

execute the application

- **Access** protected

array function Migrate::getOptions($argc, $argv) *[line 72]*

Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

retrieve command line options

- **Access** protected

void function Migrate::handleError($errno, $errstr, $errfile, $errline) *[line 119]*

Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description
- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Access** protected

void function Migrate::handleException(\$e, $e) *[line 106]*

Function Parameters:

- *\Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden to provide error handling.

- **Access** protected

Class MigrationBase

[line 18]

Provides base functionality for database migration

- **Package** PHPRum
- **Sub-Package** Migrate
- **Author** Darnell Shinbine
- **Abstract Element**

MigrationBase::$db

SQLDataAdapter = [line 24]

database

- **Access** protected

MigrationBase::$version

int = [line 30]

version

- **Access** public

Constructor *void* function MigrationBase::__construct() *[line 36]*

Constructor

- **Access** public

void function MigrationBase::down() *[line 51]*

down migration

- **Abstract Element**
- **Access** public

void function MigrationBase::up() *[line 45]*

up migration

- **Abstract Element**

- **Access** public

Class MigrationCompare

[line 18]

Provides migration sorting

- **Package** PHPRum
- **Sub-Package** Migrate
- **Author** Darnell Shinbine

int function MigrationCompare::compareVersion($a, $b) *[line 27]*

Function Parameters:

- *real* **$a** ints a
- *real* **$b** ints b

method compares version numbers

- **Access** public

Class Migrations

[line 18]

Provides base functionality for database migration

- **Package** PHPRum
- **Sub-Package** Migrate
- **Author** Darnell Shinbine

void function Migrations::downgrade([$toVersion = null]) *[line 59]*

Function Parameters:

- *real* **$toVersion** optional version, defaults to previous version

downgrade database

- **Access** public

real function Migrations::getCurrentVersion() *[line 113]*

get database version

- **Access** public

void function Migrations::to($toVersion) *[line 94]*

Function Parameters:

- *real* **$toVersion** version

set database version

- **Access** public

void function Migrations::upgrade([$toVersion = null]) *[line 31]*
Function Parameters:

- *real* **$toVersion** optional version, defaults to latest version

upgrade database

- **Access** public

Class Test

[line 19]

deplomyent script

- **Package** PHPRum
- **Sub-Package** Migrate
- **Author** Darnell Shinbine

void function Test::execute($argc, $argv) *[line 29]*
Function Parameters:

- *int* **$argc** Number of command line arguments
- *array* **$argv** Array of command line arguments

execute the application

- **Access** protected

void function Test::handleError($errno, $errstr, $errfile, $errline) *[line 88]*
Function Parameters:

- *string* **$errno** error code
- *string* **$errstr** error description
- *string* **$errfile** file
- *string* **$errline** line no.

event triggered by an error in the application, can be overridden to provide error handling.

- **Access** protected

void function Test::handleException(\$e, $e) *[line 75]*
Function Parameters:

- *\Exception* **$e**
- *Exception* **\$e**

event triggered by an uncaught Exception thrown in the application, can be overridden to provide error handling.

- **Access** protected

authentication.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

authenticationstatus.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

basicauthentication.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

credentialbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formsauthentication.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

roles.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

tablecredential.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

usercredential.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webserviceauthentication.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Authentication

[line 18]

Provides application wide authentication

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine

Authentication::$identity

string = [line 24]

authenticated id

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

Authentication::$isProtected

bool = null *[line 30]*

specifies whether the current controller is protected

- **Static**
- **Access** protected

AuthenticationStatus function Authentication::authenticate($username, $password) *[line 48]*

Function Parameters:

- *string* **$username** specifies username
- *string* **$password** specifies password

authenticate user/password based on credentials (does not set cookie)

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

bool function Authentication::authenticated() *[line 117]*

returns true if user authenticated

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

bool function Authentication::authorized() *[line 162]*

returns true if user authenticated

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

string function Authentication::generateHash($passwordFormat, $passwordToEncrypt, $salt) *[line 298]*

Function Parameters:

- *string* **$passwordFormat** password format
- *string* **$passwordToEncrypt** password to encrypt
- *string* **$salt** salt

generates a hash based on the password

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

string function Authentication::generateSalt([$len = 15]) *[line 325]*

Function Parameters:

- *string* **$len** length of salt

generates a salt

- **Static**

- **Access** public

string function Authentication::getAuthMethod() *[line 268]*

returns authentication method

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

bool function Authentication::isProtected($controllerId) *[line 279]*

Function Parameters:

- *string* **$controllerId** id of controller

returns true if controller is protected (requires authenctication to process)

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

void function Authentication::redirectToLogin() *[line 242]*

redirct user to login

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

void function Authentication::signout() *[line 92]*

perform sign out (does not end session)

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

Class AuthenticationStatus

[line 18]

Provides application wide authentication

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine

Constructor *void* function AuthenticationStatus::__construct([$invalidCredentials = false], [$disabled = false], [$lockedOut = false]) *[line 50]*

Function Parameters:

- **$invalidCredentials**
- **$disabled**
- **$lockedOut**

???

- **Var** / Constructor

- **Access** public

bool function AuthenticationStatus::authenticated() *[line 63]*

returns true if the account is authenticated

- **Access** public

bool function AuthenticationStatus::disabled() *[line 85]*

returns true if the account is locked out

- **Access** public

bool function AuthenticationStatus::invalidCredentials() *[line 74]*

returns true if invalid credentials supplied

- **Access** public

bool function AuthenticationStatus::lockedOut() *[line 96]*

returns true if the account is locked out

- **Access** public

Class BasicAuthentication

[line 18]

Provides application wide authentication using Basic HTTP Headers

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine

bool function BasicAuthentication::authenticated() *[line 25]*

returns true if user authenticated

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

string function BasicAuthentication::getAuthUser() *[line 66]*

gets auth user

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

bool function BasicAuthentication::isAuthUserSet() *[line 84]*

returns true if auth is set

- **Static**

- **Access** public

void function BasicAuthentication::sendAuthHeaders() *[line 44]*

send auth http headers

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

void function BasicAuthentication::signout() *[line 95]*

perform sign out

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

Class CredentialBase

[line 18]

Provides application wide authentication

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine
- **Abstract Element**

CredentialBase::$credential

array = array() *[line 24]*

credential data

- **Access** protected

Constructor *void* function CredentialBase::__construct($credential) *[line 31]*

Function Parameters:

- *array* **$credential** credential data

Constructor

- **Access** public

AuthenticationStatus function CredentialBase::authenticate($username, $password) *[line 44]*

Function Parameters:

- *string* **$username** specifies username
- *string* **$password** specifies password

authenticates password based on the credential

- **Abstract Element**
- **Access** public

bool function CredentialBase::comparePassword($encryptedPassword, $passwordToCompare, $salt) *[line 54]*
Function Parameters:

- *type* **$encryptedPassword**
- *type* **$passwordToCompare**
- **$salt**

compare passwords, return true on success

- **Abstract Element**
- **Access** public

Class FormsAuthentication

[line 18]

Provides application wide authentication using Forms

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine

bool function FormsAuthentication::authenticated() *[line 25]*

returns true if user authenticated

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

void function FormsAuthentication::authenticateSecret($uid, $secret) *[line 89]*

Function Parameters:

- *string* **$uid** unique value representing user
- *string* **$secret** secret

authenticate secret

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

string function FormsAuthentication::getAuthCookie() *[line 134]*

gets auth cookie

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

string function FormsAuthentication::getAuthSecret() *[line 156]*

gets auth secret

- **Static**

- **Access** public

bool function FormsAuthentication::isAuthCookieSet() *[line 178]*

returns true if auth cookie is set

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

void function FormsAuthentication::redirectFromLoginPage($uid, [$permanent = false]) *[line 42]*

Function Parameters:

- *string* **$uid** unique value representing user
- *string* **$permanent** specifies whether cookie is permanent

sets auth cookie and redirects to original requested resource

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

void function FormsAuthentication::redirectToLoginPage() *[line 64]*

redirects to login page

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

void function FormsAuthentication::setAuthCookie($uid, [$permanent = false], [$expires = 31536000]) *[line 110]*

Function Parameters:

- *string* **$uid** unique value representing user
- *string* **$permanent** specifies whether cookie is permanent
- *int* **$expires** specifies time in seconds for cookie to expire, default 1 year

sets auth cookie

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

void function FormsAuthentication::signout() *[line 200]*

perform sign out (does not end session)

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

Class Roles

[line 18]

Provides application wide authentication

- **Package** PHPRum

- **Sub-Package** Security
- **Author** Darnell Shinbine

array function Roles::getRolesForUser() *[line 64]*

get roles for current user

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

bool function Roles::isUserInRole($roleIdentifier) *[line 34]*

Function Parameters:

- *string* **$roleIdentifier** role identifier

returns true if current user is a member of the provide role

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

bool function Roles::isUserInRoles($roleIdentifiers) *[line 46]*

Function Parameters:

- *array* **$roleIdentifiers** array of role identifiers

returns true if current user is a member of the provided roles

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

Class TableCredential

[line 18]

Provides application wide authentication

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine

AuthenticationStatus function TableCredential::authenticate($username, $password) *[line 27]*
Function Parameters:

- *string* **$username** specifies username
- *string* **$password** specifies password

authenticates password based on the credential

- **Access** public

bool function TableCredential::comparePassword($encryptedPassword, $passwordToCompare, $salt) *[line 75]*
Function Parameters:

- *type* **$encryptedPassword**
- *type* **$passwordToCompare**
- **$salt**

compare passwords, return true on success

- **Access** public

string function TableCredential::generateHash($passwordToEncrypt, $salt) *[line 86]*
Function Parameters:

- **$passwordToEncrypt**
- **$salt**

generate password hash

- **Access** public

Class UserCredential

[line 18]

Provides application wide authentication

- **Package** PHPRum

- **Sub-Package** Security
- **Author** Darnell Shinbine

AuthenticationStatus function UserCredential::authenticate($username, $password) *[line 27]*
Function Parameters:

- *string* **$username** specifies username
- *string* **$password** specifies password

authenticates password based on the credential

- **Access** public

bool function UserCredential::comparePassword($encryptedPassword, $passwordToCompare, $salt) *[line 64]*
Function Parameters:

- *type* **$encryptedPassword**
- *type* **$passwordToCompare**
- **$salt**

compare passwords, return true on success

- **Access** public

Class WebServiceAuthentication

[line 18]

Provides application wide authentication using Basic HTTP Headers

- **Package** PHPRum
- **Sub-Package** Security
- **Author** Darnell Shinbine

bool function WebServiceAuthentication::authenticated() *[line 25]*

returns true if user authenticated

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

void function WebServiceAuthentication::authenticateSecret($uid, $secret) *[line 42]*

Function Parameters:

- *string* **$uid** unique value representing user
- *string* **$secret** secret

authenticate secret

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

string function WebServiceAuthentication::getAuthSecret() *[line 95]*

gets auth secret

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

string function WebServiceAuthentication::getAuthUser() *[line 77]*

gets auth user

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

bool function WebServiceAuthentication::isAuthUserSet() *[line 113]*

returns true if auth is set

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

void function WebServiceAuthentication::setAuthUser($uid) *[line 61]*

Function Parameters:

- *string* **$uid** authenticated user

sets auth user

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

void function WebServiceAuthentication::signout() *[line 131]*

perform sign out (does not end session)

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

controllertestcasebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

require_once **__ROOT__.'/system/web/webcontrols/form.class.php'** *[line 13]*

include Form, required to load constsants

htmltestreporter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

require_once **__LIB_PATH__.'/simpletest/reporter.php'** *[line 14]*

include simpletest framework

migrationstestcasebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

testcasebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

require_once **__LIB_PATH__.'/simpletest/unit_tester.php'** *[line 14]*

include simpletest framework

testreporter.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

require_once **__LIB_PATH__.'/simpletest/reporter.php'** *[line 14]*

include simpletest framework

unittestcasebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class ControllerTestCaseBase

[line 25]

Provides base functionality for the ControllerTestCase

- **Package** PHPRum
- **Sub-Package** TestCase
- **Property** string $response: response as string
- **Property** XMLEntity $responseAsXMLEntity: response as XMLEntity
- **Author** Darnell Shinbine
- **Abstract Element**

ControllerTestCaseBase::$controller

ControllerBase = null *[line 31]*

instance of the controller

- **Access** protected

ControllerTestCaseBase::$view

View = null *[line 37]*

View object

- **Access** protected

Constructor *void* function ControllerTestCaseBase::__construct($testCase) *[line 58]*
Function Parameters:

- *string* **$testCase** Name of test case

Constructor

- **Access** public

bool function ControllerTestCaseBase::assertAssigned($var, [$message = '']) *[line 275]*
Function Parameters:

- *string* **$var** variable to test
- *string* **$message** fail message

asserts true if var is assigned

- **Access** protected

bool function ControllerTestCaseBase::assertMessage($responseMessage, [\\\$messageType = null], [$message = ''], $messageType) *[line 316]*

Function Parameters:

- *string* **$responseMessage** response message
- *AppMessageType* **$messageType** message type
- *string* **$message** fail message
- *AppMessageType* **\\\$messageType**

asserts true if message exists

- **Access** protected

bool function ControllerTestCaseBase::assertNoResponse($string, [$message = ''], $response) *[line 261]*

Function Parameters:

- *string* **$response** response
- *string* **$message** fail message
- **$string**

asserts true if no response exists that matches $string

- **Access** protected

bool function ControllerTestCaseBase::assertRedirectedTo($forward, [$forwardParams = array()], [$message = ''])

[line 294]
Function Parameters:

- *string* **$forward** forward
- *array* **$forwardParams** forward parameters
- *string* **$message** fail message

asserts true if controller redirected

- **Access** protected

bool function ControllerTestCaseBase::assertResponse($string, [$message = ''], $response) *[line 247]*
Function Parameters:

- *string* **$response** response
- *string* **$message** fail message
- **$string**

asserts true if response exists

- **Access** protected

void function ControllerTestCaseBase::get([$params = array()]) *[line 152]*
Function Parameters:

- *array* **$params** data to send

simulate a get request

- **Access** protected

void function ControllerTestCaseBase::post([$params = array()]) *[line 165]*

Function Parameters:

- *array* **$params** data to send

simulate a post request

- **Access** protected

string function ControllerTestCaseBase::response() *[line 211]*

returns the response

- **Access** protected

XMLEntity function ControllerTestCaseBase::responseAsXMLEntity() *[line 232]*

returns the response as XMLEntity object

- **Access** protected

void function ControllerTestCaseBase::setUp() *[line 71]*

setup test module

- **Access** public

void function ControllerTestCaseBase::submit($formId, [$data = array()]) *[line 178]*

Function Parameters:

- *string* **$formId** id of the form
- *array* **$data**

simulate a form submition

- **Access** protected

void function ControllerTestCaseBase::tearDown() *[line 97]*

clean test module

- **Access** public

Class HTMLTestReporter

[line 24]

Provides HTML reporting for test cases

- **Package** PHPRum
- **Sub-Package** TestCase
- **Author** Darnell Shinbine

void function HTMLTestReporter::paintException($exception) *[line 88]*
Function Parameters:

- *Exception* **$exception** Exception to display.

Paints a PHP exception.

void function HTMLTestReporter::paintFooter($test_name) *[line 62]*
Function Parameters:

- *string* **$test_name** Name class of test.

Paints the end of the test with a summary of the passes and failures.

- **Access** public

void function HTMLTestReporter::paintHeader($test_name) *[line 38]*
Function Parameters:

- *string* **$test_name** Name class of test.

Paints the top of the web page setting the title to the name of the starting test.

void function HTMLTestReporter::setElapsedTime($elapsed) *[line 103]*
Function Parameters:

- *float* **$elapsed** time in s

set elapsed time

- **Access** public

Class MigrationsTestCaseBase

[line 17]

Provides base functionality for the UnitTestCase

- **Package** PHPRum
- **Sub-Package** TestCase
- **Author** Darnell Shinbine
- **Abstract Element**

MigrationsTestCaseBase::$state

array = array() *[line 23]*

states

- **Access** protected

void function MigrationsTestCaseBase::setUp() *[line 31]*

setup test module

- **Access** public

void function MigrationsTestCaseBase::tearDown() *[line 41]*

clean test module

- **Access** public

XMLEntity function MigrationsTestCaseBase::xxx() *[line 51]*

returns the response as XMLEntity object

- **Access** protected

Class TestCaseBase

[line 26]

Provides base functionality for the TestCase

- **Package** PHPRum
- **Sub-Package** TestCase
- **Property** string $fixtures: comma seperated string containing fixture filenames
- **Author** Darnell Shinbine

- **Abstract Element**

TestCaseBase::$fixtures

string = '' *[line 32]*

fixtures to load

- **Access** protected

void function TestCaseBase::cleanup() *[line 155]*

called after evey test

- **Access** protected

array function TestCaseBase::getFixtureVariables() *[line 136]*

returns an associative array of fixture variables used for dynamically inserting values inside fixtures

- **Access** protected

bool function TestCaseBase::loadCSVFixture($fixture) *[line 293]*

Function Parameters:

- *string* **$fixture** name of fixture

load csv fixture

- **Access** protected

void function TestCaseBase::loadFixtures($fixtures) *[line 165]*

Function Parameters:

- *array* **$fixtures** array of fixtures

load fixtures

- **Access** protected

void function TestCaseBase::loadSQLFixture($fixture) *[line 321]*

Function Parameters:

- *string* **$fixture** name of fixture

load sql fixture

- **Access** protected

void function TestCaseBase::loadXMLFixture($fixture) *[line 230]*

Function Parameters:

- *string* **$fixture** name of fixture

load xml fixture

- **Access** protected

void function TestCaseBase::prepare() *[line 146]*

called before evey test

- **Access** protected

void function TestCaseBase::run(&$reporter, $reporter) *[line 106]*

Function Parameters:

- *TestReporter* **$reporter** TestReporter object
- **&$reporter**

run report

- **Access** public

Class TestReporter

[line 24]

Provides basic reporting for test cases

- **Package** PHPRum
- **Sub-Package** TestCase
- **Author** Darnell Shinbine

void function TestReporter::setElapsedTime($elapsed) *[line 39]*
Function Parameters:

- *float* **$elapsed** time in s

set elapsed time

- **Access** public

Class UnitTestCaseBase

[line 17]

Provides base functionality for the UnitTestCase

- **Package** PHPRum
- **Sub-Package** TestCase
- **Author** Darnell Shinbine
- **Abstract Element**

Constructor *void* function UnitTestCaseBase::__construct($testCase) *[line 27]*
Function Parameters:

- *string* **$testCase** Name of test case

Constructor

- **Access** public

void function UnitTestCaseBase::setUp() *[line 37]*

setup test module

- **Access** public

void function UnitTestCaseBase::tearDown() *[line 47]*

clean test module

- **Access** public

timer.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Timer

[line 18]

Provides basic timer functionality

- **Package** PHPRum
- **Sub-Package** Utils
- **Author** Darnell Shinbine

Constructor *void* function Timer::__construct([$autostart = false]) *[line 58]*

Function Parameters:

- *bool* **$autostart** Specifies whether to automatically start the timer

Constructor

- **Access** public

real function Timer::elapsed() *[line 141]*

return elapsed time from start, if timer has been stopped then time will be calculated based on stop time

- **Access** public

real function Timer::pause() *[line 87]*

pause the timer

- **Access** public

void function Timer::reset() *[line 161]*

reset timer

- **Access** public

real function Timer::resume() *[line 103]*

resume the timer

- **Access** public

void function Timer::start() *[line 70]*

start timer

- **Access** public

void function Timer::stop() *[line 119]*

stop timer

- **Access** public

booleanvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

datetimevalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

emailvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

enumvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filesizevalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

filetypevalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

integervalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

lengthvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

matchvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

numericvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

patternvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

rangevalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

requiredvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

uniquevalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

urlvalidator.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

validatorbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

validatorcollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class BooleanValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

void function BooleanValidator::onLoad() *[line 27]*

on load

- **Access** protected

void function BooleanValidator::validate() *[line 41]*

sets the controlId and prepares the control attributes

- **Access** public

Class DateTimeValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

void function DateTimeValidator::onLoad() *[line 27]*

on load

- **Access** protected

void function DateTimeValidator::validate() *[line 41]*

sets the controlId and prepares the control attributes

- **Access** public

Class EmailValidator

[line 19]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function EmailValidator::__construct([$errorMessage = '']) *[line 27]*

Function Parameters:

- *string* **$errorMessage** error message

EmailAddressValidator

- **Access** public

void function EmailValidator::onLoad() *[line 38]*

on load

- **Access** protected

Class EnumValidator

[line 17]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Author** Darnell Shinbine

Constructor *void* function EnumValidator::__construct($values, [$errorMessage = '']) *[line 31]*

Function Parameters:

- *string* **$errorMessage** error message
- *array* **$values**

URLValidator

- **Access** public

void function EnumValidator::onLoad() *[line 43]*

on load

- **Access** protected

void function EnumValidator::validate() *[line 65]*

sets the controlId and prepares the control attributes

- **Access** public

Class FileSizeValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function FileSizeValidator::__construct($maxSize, [$errorMessage = '']) *[line 36]*

Function Parameters:

- *double* **$maxSize** max size
- *string* **$errorMessage** error message

FileSizeValidator

- **Access** public

void function FileSizeValidator::onLoad() *[line 49]*

on load

- **Access** protected

bool function FileSizeValidator::validate() *[line 63]*

validates the control

- **Access** public

Class FileTypeValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function FileTypeValidator::__construct([$types = array()], [$errorMessage = '']) *[line 36]*

Function Parameters:

- *array* **$types** array of types
- *string* **$errorMessage** error message

FileTypeValidator

- **Access** public

void function FileTypeValidator::addType($type) *[line 50]*

Function Parameters:

- *string* **$type** file type

adds a file type to the list of allowed types

- **Access** public

void function FileTypeValidator::onLoad() *[line 61]*

on load

- **Access** protected

bool function FileTypeValidator::validate() *[line 75]*

validates the control

- **Access** public

Class IntegerValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

void function IntegerValidator::onLoad() *[line 27]*

on load

- **Access** protected

void function IntegerValidator::validate() *[line 41]*

sets the controlId and prepares the control attributes

- **Access** public

Class LengthValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function LengthValidator::__construct($min, $max, [$errorMessage = '']) *[line 43]*

Function Parameters:

- *double* **$min** min value
- *double* **$max** max value
- *string* **$errorMessage** error message

LengthValidator

- **Access** public

void function LengthValidator::onLoad() *[line 57]*

on load

- **Access** protected

void function LengthValidator::validate() *[line 71]*

sets the controlId and prepares the control attributes

- **Access** public

Class MatchValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

MatchValidator::$controlToMatch

InputBase = *[line 26]*

control to match

- **Access** protected

Constructor *void* function MatchValidator::__construct(\\\\&$controlToMatch, [$errorMessage = ''], $controlToMatch) *[line 36]*

Function Parameters:

- *InputBase* **$controlToMatch** control to match
- *string* **$errorMessage** error message
- *InputBase* **\\\\&$controlToMatch**

MatchValidator

- **Access** public

void function MatchValidator::onLoad() *[line 49]*

on load

- **Access** protected

void function MatchValidator::validate() *[line 63]*

sets the controlId and prepares the control attributes

- **Access** public

Class NumericValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

void function NumericValidator::onLoad() *[line 27]*

on load

- **Access** protected

void function NumericValidator::validate() *[line 41]*

sets the controlId and prepares the control attributes

- **Access** public

Class PatternValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function PatternValidator::__construct($pattern, [$errorMessage = '']) *[line 36]*

Function Parameters:

- *string* **$pattern** pattern
- *string* **$errorMessage** error message

PatternValidator

- **Access** public

void function PatternValidator::onLoad() *[line 49]*

on load

- **Access** protected

void function PatternValidator::validate() *[line 63]*

sets the controlId and prepares the control attributes

- **Access** public

Class RangeValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function RangeValidator::__construct($min, $max, [$errorMessage = '']) *[line 43]*

Function Parameters:

- *double* **$min** min value
- *double* **$max** max value
- *string* **$errorMessage** error message

RangeValidator

- **Access** public

void function RangeValidator::onLoad() *[line 57]*

on load

- **Access** protected

void function RangeValidator::validate() *[line 71]*

sets the controlId and prepares the control attributes

- **Access** public

Class RequiredValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

void function RequiredValidator::onLoad() *[line 27]*

on load

- **Access** protected

bool function RequiredValidator::validate() *[line 42]*

validates the control

- **Access** public

Class UniqueValidator

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

void function UniqueValidator::onLoad() *[line 27]*

on load

- **Access** protected

void function UniqueValidator::validate() *[line 41]*

sets the controlId and prepares the control attributes

- **Access** public

Class URLValidator

[line 19]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $controlId: Control Id
- **Author** Darnell Shinbine

Constructor *void* function URLValidator::__construct([$errorMessage = '']) *[line 27]*

Function Parameters:

- *string* **$errorMessage** error message

URLValidator

- **Access** public

void function URLValidator::onLoad() *[line 38]*

on load

- **Access** protected

Class ValidatorBase

[line 20]

Provides basic validation for web controls

- **Package** PHPRum
- **Sub-Package** Validators
- **Property** string $errorMessage: error message
- **Author** Darnell Shinbine
- **Abstract Element**

ValidatorBase::$controlToValidate

InputBase = [line 33]

control to validate

- **Access** protected

ValidatorBase::$errorMessage

string = [line 26]

error message

- **Access** protected

Constructor *void* function ValidatorBase::__construct([$errorMessage = '']) *[line 42]*

Function Parameters:

- *string* **$errorMessage** error message

ValidatorBase

- **Access** public

void function ValidatorBase::onLoad() *[line 127]*

on load

- **Access** protected

void function ValidatorBase::setControlToValidate(\\\\&$controlToValidate, $controlToValidate) *[line 105]*

Function Parameters:

- *InputBase* **$controlToValidate** control to validate
- *InputBase* **\\\\&$controlToValidate**

set control to validate

- **Access** public

void function ValidatorBase::setErrorMessage($errorMessage) *[line 93]*

Function Parameters:

- *string* **$errorMessage** error message

set error message

- **Access** public

bool function ValidatorBase::validate($controlToValidate) *[line 119]*

Function Parameters:

- *InputBase* **$controlToValidate** control to validate

validates the control

- **Abstract Element**
- **Access** public

Class ValidatorCollection

[line 19]

Represents a Collection of Validator objects

- **Package** PHPRum
- **Sub-Package** Validators
- **Author** Darnell Shinbine

Constructor *void* function ValidatorCollection::__construct(\\\\&$control, $parent) *[line 35]*
Function Parameters:

- *InputBase* **$parent** instance of a InputBase object
- *InputBase* **\\\\&$control**

Constructor

- **Access** public

void function ValidatorCollection::add($item) *[line 72]*
Function Parameters:

- *Validator* **$item**

add Validator to Collection

- **Access** public

bool function ValidatorCollection::contains($item) *[line 92]*
Function Parameters:

- *mixed* **$item** item

returns true if item array contains item

- **Access** public

controllerbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formajaxpostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formajaxposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formpostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcolumnajaxpostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcolumnajaxposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcolumnpostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcolumnposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputajaxchangeevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputajaxchangeeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputajaxpostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputajaxposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputchangeevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputchangeeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputpostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

inputposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

loginformsubmitevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

loginformsubmiteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

masterviewinitevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

masterviewiniteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

masterviewloadevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

masterviewloadeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagepostevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pageposteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagerequestevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagerequesteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportdetailsevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportdetailseventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportfooterevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportfootereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportgroupfooterevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportgroupfootereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportgroupheaderevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportgroupheadereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportheaderevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportheadereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolcreateevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolcreateeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolinitevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontroliniteventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolloadevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolloadeventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolprerenderevent.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolprerendereventhandler.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

formmodelbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

forwardmethodtype.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

pagecontrollerbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

jsonservicebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

restwebservicebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

soapwebservicebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webservicebase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webservicemethod.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

button.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

checkbox.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

checkboxlist.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dateselector.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

datetimeselector.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

dropdownlist.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

fieldset.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

filebrowser.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

form.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

gridview.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

gridviewbutton.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcheckbox.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewcolumn.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

gridviewcolumncollection.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

gridviewcontrolbase.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

gridviewdropdownmenu.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewimage.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewlink.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

gridviewtextbox.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

hyperlink.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

inputbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

label.class.php

- **Package** PHPRum
- **Filesource** Source Code for this file
- **License** see

listbox.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

listcontrolbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

listitemcollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

loginform.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

masterview.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

page.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

radiobutton.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

radiobuttonlist.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

radiogroup.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportgroup.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

reportview.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

textarea.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

textbox.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

timeselector.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

treenode.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

treenodecollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

treeview.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

view.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolattributecollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolbase.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

webcontrolcollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class Button

[line 23]

Represents a Button Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $text: Specifies button text
- **Property** string $src: Specifies button image source
- **Property** string $submitText: Specifies button text to display on submit
- **Property** bool $disableOnSubmit: Specifies whether to disable all other buttons on submit
- **Author** Darnell Shinbine

Button::$disableOnSubmit

bool = false *[line 47]*

specifies whether to disable all other buttons on submit

- **Access** protected

Button::$src

string = '' *[line 35]*

specifies button image source

- **Access** protected

Button::$submitText

string = '' *[line 41]*

specifies button text to display on submit

- **Access** protected

Button::$text

string = '' *[line 29]*

specifies button text

- **Access** protected

Constructor *void* function Button::__construct($controlId, [$text = '']) *[line 57]*
Function Parameters:

- *string* **$controlId** Control Id
- *string* **$text** Button text

Constructor

- **Access** public

DomObject function Button::getDomObject() *[line 145]*

getDomObject

returns a DomObject representing control

- **Access** public

bool function Button::onLoad() *[line 192]*

onLoad

called when control is loaded

- **Access** protected

void function Button::onRequest(&$request) *[line 219]*

Function Parameters:

- *array* **&$request** request data

onRequest

process the HTTP request array

- **Access** protected

Class CheckBox

[line 20]

Represents a CheckBox Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** bool $checked: Specifies whether button is checked
- **Author** Darnell Shinbine

DomObject function CheckBox::getDomObject() *[line 68]*

returns a DomObject representing control

- **Access** public

bool function CheckBox::onLoad() *[line 119]*

called when control is loaded

- **Access** protected

void function CheckBox::onRequest(&$request) *[line 135]*

Function Parameters:

- *array* **&$request**

process the HTTP request array

- **Access** protected

Class CheckBoxList

[line 18]

Represents a CheckBoxList Control

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

CheckBoxList::$multiple

bool = true *[line 24]*

size of listbox

- **Access** protected

DomObject function CheckBoxList::getDomObject() *[line 32]*

returns a DomObject representing control

- **Access** public

Class ControllerBase

[line 25]

This class handles all requests for a specific uri.

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $controllerId: Specifies the controller id
- **Property** int $outputCache: Specifies how long to cache page output in seconds, 0 disables caching
- **Property** bool $requireSSL: Specifies whether SSL is required for this page
- **Property** array $allowRoles: Specifies the roles that have been granted access this page
- **Property** array $denyRoles: Specifies the roles that have been denied access this page
- **Property** EventCollection $events: event collection
- **Author** Darnell Shinbine
- **Abstract Element**

ControllerBase::$allowRoles

array = array() *[line 49]*

Specifies the roles that have been granted access this page

- **Access** protected

ControllerBase::$controllerId

string = '' *[line 31]*

Specifies the controller id

- **Access** protected

ControllerBase::$denyRoles

array = array() *[line 55]*

Specifies the roles that have been denied access this page

- **Access** protected

ControllerBase::$events

EventCollection = null *[line 61]*

event collection

- **Access** protected

ControllerBase::$outputCache

int = 0 *[line 37]*

Specifies how long to cache page output in seconds, 0 disables caching

- **Access** protected

ControllerBase::$requireSSL

bool = false *[line 43]*

Specifies whether the controller requires an SSL connection

- **Access** protected

Constructor *void* function ControllerBase::__construct($controllerId) *[line 70]*
Function Parameters:

- *string* **$controllerId** Controller Id

Constructor

- **Access** public

string function ControllerBase::getCacheId(&$request) *[line 154]*
Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object

this method will generate and return a cache id

- **Access** public

array function ControllerBase::getRoles() *[line 142]*

this method return an array of roles that have been granted access

- **Access** public

View function ControllerBase::getView(\\\&$request, &$request) *[line 166]*

Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

return view component for rendering

- **Abstract Element**
- **Access** public

void function ControllerBase::onLoad() *[line 131]*

Event called when object created

- **Access** protected

Class DateSelector

[line 24]

handles date control element creation (string) abstracts away the presentation logic and data access layer the server-side control for WebWidgets

- **Package** PHPRum
- **Sub-Package** Web
- **Property** int $yearMin: year minimum
- **Property** int $yearMax: year maximum
- **Property** bool $allowNull: specifies whether to allow null values
- **Author** Darnell Shinbine

DateSelector::$allowNull

mixed = false *[line 42]*

specifies whether to allow nulls

- **Access** protected

DateSelector::$yearMax

mixed = 2020 *[line 36]*

specifies the year range

- **Access** protected

DateSelector::$yearMin

mixed = 1900 *[line 30]*

specifies the year range

- **Access** protected

Constructor *void* function DateSelector::__construct($controlId, [$default = null]) *[line 51]*
Function Parameters:

- **$controlId**
- **$default**

Constructor

- **Access** public

void function DateSelector::getDomObject(0) *[line 185]*
Function Parameters:

- *none* **0**

returns widget object

- **Access** public

void function DateSelector::onLoad() *[line 170]*

called when control is loaded

- **Access** protected

void function DateSelector::onRequest(&$httpRequest) *[line 116]*

Function Parameters:

- *array* **&$httpRequest**

process the HTTP request array

- **Access** protected

Class DateTimeSelector

[line 24]

handles date control element creation (string) abstracts away the presentation logic and data access layer the server-side control for WebWidgets

- **Package** PHPRum

- **Sub-Package** Web
- **Property** int $yearMin: year minimum
- **Property** int $yearMax: year maximum
- **Property** bool $allowNull: specifies whether to allow null values
- **Author** Darnell Shinbine

DateTimeSelector::$allowNull

mixed = false *[line 42]*

specifies whether to allow nulls

- **Access** protected

DateTimeSelector::$yearMax

mixed = 2020 *[line 36]*

specifies the year range

- **Access** protected

DateTimeSelector::$yearMin

mixed = 1900 *[line 30]*

specifies the year range

- **Access** protected

Constructor *void* function DateTimeSelector::__construct($controlId, [$default = null]) *[line 51]*

Function Parameters:

- **$controlId**
- **$default**

Constructor

- **Access** public

void function DateTimeSelector::getDomObject(0) *[line 199]*

Function Parameters:

- *none* **0**

returns widget object

- **Access** public

void function DateTimeSelector::onLoad() *[line 115]*

called when control is loaded

- **Access** protected

void function DateTimeSelector::onRequest(&$httpRequest) *[line 129]*

Function Parameters:

- *array* **&$httpRequest**

process the HTTP request array

- **Access** protected

Class DropDownList

[line 18]

Represents a DropDownList Control

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

DropDownList::$listSize

int = 1 *[line 24]*

size of listbox

- **Access** protected

DropDownList::$multiple

bool = false *[line 30]*

size of listbox

- **Access** protected

Class Fieldset

[line 22]

Represents a Fieldset

- **Package** PHPRum
- **Sub-Package** Web
- **Property** bool $ajaxPostBack: specifies whether to perform ajax postback, Default is false
- **Property** bool $ajaxValidation: specifies whether to perform ajax validation, Default is false
- **Property** string $legend: fieldset legend
- **Author** Darnell Shinbine

Fieldset::$legend

string = '' *[line 28]*

Fieldset legend

- **Access** protected

Constructor *void* function Fieldset::__construct($controlId) *[line 38]*

Function Parameters:

- *string* **$controlId** Control Id

Constructor

- **Access** public

void function Fieldset::add($control, &$control) *[line 101]*

Function Parameters:

- *InputBase* **&$control** instance of an InputBase
- *InputBase* **$control**

adds child control to collection

- **Access** public

void function Fieldset::fillDataSource(\&$ds, $ds) *[line 160]*

Function Parameters:

- *\ArrayAccess* **$ds** data source to fill
- *ArrayAccess* **\&$ds**

update data source with Control data

- **Access** public

bool function Fieldset::focus() *[line 123]*

sets focus to the control

- **Access** public

DomObject function Fieldset::getDomObject() *[line 189]*

returns a DomObject representing control

- **Access** public

bool function Fieldset::onLoad() *[line 112]*

called when control is loaded

- **Access** protected

void function Fieldset::readDataSource(\&$ds, $ds) *[line 175]*

Function Parameters:

- *\ArrayAccess* **$ds** data source to read
- *ArrayAccess* **\&$ds**

update Control with data from the data source

- **Access** public

bool function Fieldset::validate([&$errMsg = ''], [&$controlToFocus = null], $errMsg) *[line 139]*

Function Parameters:

- *string* **$errMsg** error message
- **&$errMsg**
- *InputBase* **&$controlToFocus**

validate all controls in form object

- **Access** public

Class FileBrowser

[line 18]

Represents a FileBrowser Control

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function FileBrowser::__construct($controlId) *[line 26]*

Function Parameters:

- *string* **$controlId** Control Id

Constructor

- **Access** public

DomObject function FileBrowser::getDomObject() *[line 125]*

returns a DomObject representing control

- **Access** public

array function FileBrowser::getFileInfo() *[line 39]*

returns information on the uploaded file

- **Access** public

string function FileBrowser::getFileRawData() *[line 57]*

return raw file data

- **Access** public

bool function FileBrowser::onLoad() *[line 140]*

called when control is loaded

- **Access** protected

void function FileBrowser::onRequest(&$request) *[line 157]*

Function Parameters:

- *array* **&$request**

process the HTTP request array

- **Access** protected

Class Form

[line 35]

Represents a Form

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $action: form action
- **Property** string $method: form submit method
- **Property** string $encodeType: form encoding type
- **Property** string $forward: controller to forward to
- **Property** bool $ajaxPostBack: specifies whether to perform ajax postback, Default is false

- **Property** bool $ajaxValidation: specifies whether to perform ajax validation, Default is false
- **Property** bool $autoFocus: specifies whether to auto focus
- **Property** bool $hiddenField: specifies whether to check for hidden honeypot field before processing request
- **Property** string $submitted: specifies if form was submitted
- **Property** RequestParameterCollection $parameters: form parameters
- **Author** Darnell Shinbine

Form::$action

string = '' *[line 48]*

URL to send form data, Default is self

- **Access** protected

Form::$ajaxPostBack

bool = false *[line 78]*

turn on or off ajax post backs

- **Access** protected

Form::$ajaxValidation

bool = false *[line 84]*

turn on or off ajax validation

- **Access** protected

Form::$autoFocus

bool = false *[line 72]*

turn on or off auto focusing

- **Access** protected

Form::$encodeType

string = 'application/x-www-form-urlencoded' *[line 60]*

Form encoding type

- **Access** protected

Form::$forward

string = '' *[line 66]*

controller to forward to, Default is the current controller

- **Access** protected

Form::$hiddenField

bool = false *[line 90]*

specifies whether to check for hidden honeypot field before processing request

- **Access** protected

Form::$method

string = 'POST' *[line 54]*

method to send form data, Default is 'POST'

- **Access** protected

Form::$parameters

StringDictionary = null *[line 102]*

array of variables to post on submit

- **Access** protected

Form::$submitted

bool = false *[line 96]*

set if the form was submitted

- **Access** protected

Constructor *void* function Form::__construct($controlId) *[line 118]*
Function Parameters:

- *string* **$controlId** Control Id

Constructor

- **Access** public

void function Form::add($control, &$control) *[line 278]*
Function Parameters:

- *InputBase* **&$control** instance of an InputBase
- *WebControlBase* **$control**

adds child control to collection

- **Access** public

void function Form::addParameter($key, $value) *[line 298]*
Function Parameters:

- *string* **$key** key
- *string* **$value** value

add parameter

- **Access** public

void function Form::begin([$args = array()]) *[line 387]*

Function Parameters:

- *array* **$args** attribute parameters

renders form open tag

- **Access** public

void function Form::end() *[line 399]*

renders form close tag

- **Access** public

DomObject function Form::getDomObject() *[line 410]*

returns a DomObject representing control

- **Access** public

void function Form::getFormDomObject() *[line 493]*

returns form as widget

- **Access** protected

void function Form::onDataBind() *[line 691]*

bind all child controls in form to datasource

- **Access** protected

bool function Form::onLoad() *[line 584]*

called when control is loaded

- **Access** protected

void function Form::onPost(&$request) *[line 670]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** protected

void function Form::onRequest(&$request) *[line 610]*

Function Parameters:

- *array* **&$request** request array

process the HTTP request array

- **Access** protected

void function Form::save() *[line 309]*

update DataSet using values from child controls

- **Access** public

bool function Form::validate([&$errMsg = ''], [&$controlToFocus = null], $errMsg) *[line 348]*

Function Parameters:

- *string* **$errMsg** error message
- **&$errMsg**
- *InputBase* **&$controlToFocus**

validate all controls in form object

- **Access** public

Class FormAjaxPostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function FormAjaxPostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class FormAjaxPostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function FormAjaxPostEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class FormModelBase

[line 23]

This class represents a form.

The FormModelBase exposes 2 protected properties

- **Package** PHPRum
- **Sub-Package** Web
- **Property** array $fields: Contains an associative array of field names mapped to field types
- **Property** array $rules: Contains an associative array of field names mapped to rules
- **Author** Darnell Shinbine
- **Abstract Element**

FormModelBase::$fields

array = array() *[line 29]*

Contains an associative array of field names mapped to field types

- **Access** protected

FormModelBase::$field_mappings

```
array = array(
        'binary' => 'System\Web\WebControls\FileBrowser',
        'blob' => 'System\Web\WebControls\TextArea',
        'boolean' => 'System\Web\WebControls\CheckBox',
        'date' => 'System\Web\WebControls\DateSelector',
        'datetime' => 'System\Web\WebControls\DateTimeSelector',
        'enum' => 'System\Web\WebControls\DropDownList',
        'integer' => 'System\Web\WebControls\TextBox',
        'numeric' => 'System\Web\WebControls\TextBox',
        'real' => 'System\Web\WebControls\TextBox',
        'ref' => 'System\Web\WebControls\DropDownList',
        'string' => 'System\Web\WebControls\TextBox',
        'time' => 'System\Web\WebControls\TimeSelector'
    ) [line 41]
```

Contains field mappings

- **Static**
- **Access** protected

FormModelBase::$rules

array = array() *[line 35]*

Contains an associative array of field names mapped to rules

- **Access** protected

FormModelBase::$rule_mappings

```
array = array(
        'boolean' => 'System\Validators\BooleanValidator',
        'datetime' => 'System\Validators\DateTimeValidator',
        'email' => 'System\Validators\EmailValidator',
        'enum' => 'System\Validators\EnumValidator',
        'filesize' => 'System\Validators\FileSizeValidator',
        'filetype' => 'System\Validators\FileTypeValidator',
        'integer' => 'System\Validators\IntegerValidator',
        'length' => 'System\Validators\LengthValidator',
        'match' => 'System\Validators\MatchValidator',
        'numeric' => 'System\Validators\NumericValidator',
        'pattern' => 'System\Validators\PatternValidator',
        'range' => 'System\Validators\RangeValidator',
        'required' => 'System\Validators\RequiredValidator',
        'unique' => 'System\Validators\UniqueValidator',
        'url' => 'System\Validators\URLValidator'
      ) [line 60]
```

Contains rule mappings

- **Static**
- **Access** protected

Constructor *void* function FormModelBase::__construct() *[line 92]*

Constructor

Read values from request, clean up variables, and merge get and post requests.

- **Access** protected

Form function FormModelBase::form($controlId) *[line 240]*

Function Parameters:

- *string* **$controlId** form id

static method to return a Form object

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

void function FormModelBase::registerFieldType($field, $type) *[line 342]*

Function Parameters:

- *string* **$field** field type
- *string* **$type** path to control

register new field type

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

void function FormModelBase::registerRuleType($rule, $type) *[line 355]*

Function Parameters:

- *string* **$rule** name of rule
- *string* **$type** path to control

register new rule type

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

void function FormModelBase::save() *[line 231]*

save model state

- **Abstract Element**

- **Access** public

array function FormModelBase::toArray() *[line 220]*

converts the form model into an array

- **Access** public

Class FormPostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function FormPostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class FormPostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function FormPostEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class ForwardMethodType

[line 18]

Represents an application forward method

- **Package** PHPRum
- **Sub-Package** Web

- **Author** Darnell Shinbine

ForwardMethodType function ForwardMethodType::Request() *[line 37]*

Specifies the application forward method as Request

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

ForwardMethodType function ForwardMethodType::URI() *[line 31]*

Specifies the application forward method as URI

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

Class GridView

[line 49]

Represents a GridView Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $caption:
- **Property** int $pageSize:
- **Property** int $page:

- **Property** int $filterGroupSize:
- **Property** bool $canSort:
- **Property** bool $canChangeOrder:
- **Property** bool $showFilters:
- **Property** bool $showHeader:
- **Property** bool $showFooter:
- **Property** bool $showPageNumber:
- **Property** bool $showPrimaryKey:
- **Property** bool $showList:
- **Property** bool $autoGenerateColumns:
- **Property** string $valueField:
- **Property** string $orderByField:
- **Property** GridViewColumnCollection $columns:
- **Property** bool $multiple:
- **Property** array $selected:
- **Property** string $listName:
- **Property** string $sortBy:
- **Property** string $sortOrder:
- **Property** string $moveCursor:
- **Property** string $moveOrder:
- **Property** array $filters:
- **Property** string $onmouseover:
- **Property** string $onmouseout:
- **Property** string $onclick:
- **Property** string $ondblclick:
- **Property** bool $ajaxPostBack: specifies whether to perform ajax postback on change, Default is false
- **Author** Darnell Shinbine
- **Version** 2.0

GridView::$actions

array = array() *[line 199]*

Array of actions

- **Access** protected

GridView::$ajaxPostBack

bool = false *[line 235]*

Specifies whether form will submit ajax postback on change, Default is false

- **Access** protected

GridView::$autoGenerateColumns

bool = false *[line 127]*

Specifies whether to generate columns from datasource, default is false

- **Access** protected

GridView::$canChangeOrder

bool = false *[line 85]*

Specifies if table order can be changed, Default is false

- **Access** protected

GridView::$canSort

bool = true *[line 79]*

Specifies if table is sortable, Default is true

- **Access** protected

GridView::$caption

string = '' *[line 55]*

Table caption, Default is none

- **Access** protected

GridView::$columns

GridViewColumnCollection = *[line 205]*

collection of columns

- **Access** protected

GridView::$filterGroupSize

int = 10 *[line 73]*

Number of records to display in dropdown filter, Default is 10

- **Access** protected

GridView::$filters

array = array() *[line 187]*

current grid filters

- **Access** protected

GridView::$filterValues

array = array() *[line 193]*

grid filter values

- **Access** protected

GridView::$listName

string = '&nbsp;' *[line 157]*

optional name for the list

- **Access** protected

GridView::$moveCursor

string = 0 *[line 175]*

current grid moveCursor field

- **Access** protected

GridView::$moveOrder

bool = '' *[line 181]*

current grid moveCursor order

- **Access** protected

GridView::$multiple

bool = false *[line 133]*

specifies if list can have multiple values (used when showList is true), default is false

- **Access** protected

GridView::$onclick

string = '' *[line 223]*

specifies the action to take on click events

- **Access** protected

GridView::$ondblclick

string = '' [line 229]

specifies the action to take on double click events

- **Access** protected

GridView::$onmouseout

string = '' [line 217]

specifies the action to take on onmouseout events

- **Access** protected

GridView::$onmouseover

string = '' [line 211]

specifies the action to take on mouseover events

- **Access** protected

GridView::$orderByField

string = '' [line 151]

specifies the order by field

- **Access** protected

GridView::$page

int = 1 *[line 67]*

current grid page

- **Access** protected

GridView::$pageSize

int = 20 *[line 61]*

Number of rows to display per page, Default is 20

- **Access** protected

GridView::$selected

array = array() *[line 139]*

array of values for selected items (used when showList is true)

- **Access** protected

GridView::$showFilters

bool = false *[line 91]*

Set to display column filters, Default is false

- **Access** protected

GridView::$showFooter

bool = false *[line 103]*

Set to display table footer, Default is false

- **Access** protected

GridView::$showHeader

bool = true *[line 97]*

Set to display column headers, Default is true

- **Access** protected

GridView::$showList

bool = false *[line 121]*

Show list view, default is false

- **Access** protected

GridView::$showPageNumber

bool = true *[line 109]*

Set to display table footer, Default is true

- **Access** protected

GridView::$showPrimaryKey

bool = false *[line 115]*

Set to display primary key in view, Default is false

- **Access** protected

GridView::$sortBy

string = '' *[line 163]*

current grid sorting field

- **Access** protected

GridView::$sortOrder

bool = '' *[line 169]*

current grid sorting order

- **Access** protected

GridView::$valueField

string = '' *[line 145]*

specifies the value field

- **Access** protected

Constructor *void* function GridView::__construct($controlId) *[line 250]*

Function Parameters:

- *string* **$controlId** Control Id

sets the controlId and prepares the control attributes

- **Access** public

void function GridView::addColumn(&$column, $column) *[line 478]*

Function Parameters:

- *GridViewColumn* **$column** GridView column
- *GridViewColumn* **&$column**

add a bound column

- **Access** public

GridViewColumn function GridView::findColumn($dataField) *[line 466]*

Function Parameters:

- *string* **$dataField** data field

returns GridViewColumn if column is found in Collection

- **Access** public

DomObject function GridView::getDomObject() *[line 502]*

returns a DomObject representing control

- **Access** public

DomObject function GridView::getPagination() *[line 1461]*

generic method for handling the table pagination

- **Access** protected

DomObject function GridView::getRowBody(\\\&$ds, $ds) *[line 1192]*

Function Parameters:

- *DataSet* **$ds** DataSet object with current resultset
- *DataSet* **\\\&$ds**

generic method for handling the row body

- **Access** protected

DomObject function GridView::getRowFilter() *[line 1060]*

generic method for handling the row header

- **Access** protected

DomObject function GridView::getRowFooter() *[line 1413]*

generic method for handling the table footer

- **Access** protected

DomObject function GridView::getRowHeader() *[line 957]*

generic method for handling the row header

- **Access** protected

void function GridView::onDataBind(0) *[line 731]*

Function Parameters:

- *$default* **0** value

bind control to data

- **Access** protected

bool function GridView::onLoad() *[line 754]*

called when control is loaded

- **Access** protected

void function GridView::onLoadViewState(&$viewState, $viewState) *[line 705]*

Function Parameters:

- *array* **$viewState** session data
- *array* **&$viewState**

read view state from session

- **Access** protected

void function GridView::onPost(&$request) *[line 909]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** protected

void function GridView::onRequest(&$request) *[line 772]*

Function Parameters:

- *array* **&$request**

process the HTTP request array

- **Access** protected

void function GridView::onSaveViewState(&$viewState, $viewState) *[line 921]*

Function Parameters:

- *object* **$viewState** session array
- *array* **&$viewState**

write view state to session

- **Access** protected

void function GridView::onUpdateAjax() *[line 939]*

Event called on ajax callback

- **Access** protected

void function GridView::setFilterValues($datafield, $values) *[line 491]*

Function Parameters:

- *string* **$datafield** name of datafield
- *array* **$values** array of values

set filter values

- **Access** public

Class GridViewButton

[line 20]

Represents a GridView button

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $confirmation: confirmation message
- **Author** Darnell Shinbine

GridViewButton::$canFilter

bool = false *[line 26]*

specifies whether column can be filtered

- **Access** protected

GridViewButton::$confirmation

string = '' *[line 32]*

confirmation message

- **Access** protected

Constructor *void* function GridViewButton::__construct($dataField, [$buttonName = ''], [$parameter = ''], [$confirmation = ''], [$headerText = ''], [$footerText = ''], [$className = '']) *[line 50]*

Function Parameters:

- *string* **$dataField** field name
- *string* **$buttonName** name of button
- *string* **$parameter** parameter
- *string* **$headerText** header text
- *string* **$footerText** footer text
- *string* **$className** css class name
- **$confirmation**

- **Access** public

string function GridViewButton::getItemText($dataField, $parameter, $params) *[line 127]*

Function Parameters:

- *string* **$dataField**

- *string* **$parameter**
- *string* **$params**

get item text

- **Access** protected

void function GridViewButton::onPost(&$request) *[line 108]*
Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** public

Class GridViewCheckBox

[line 18]

Represents a GridView CheckBox

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

string function GridViewCheckBox::getItemText($dataField, $parameter, $params) *[line 28]*

Function Parameters:

- *string* **$dataField**
- *string* **$parameter**
- *string* **$params**

get item text

- **Access** protected

Class GridViewColumn

[line 30]

Represents a GridView column

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $dataField: datefield
- **Property** string $headerText: header text
- **Property** string $itemText: item text (templating allowed)
- **Property** string $footerText: footer text
- **Property** string $className: class name
- **Property** EventCollection $events: event collection

- **Property** bool $canFilter: specifies whether column can be filtered
- **Property** string $onmouseover:
- **Property** string $onmouseout:
- **Property** string $onclick:
- **Property** string $ondblclick:

GridViewColumn::$canFilter

bool = true *[line 72]*

specifies whether column can be filtered

- **Access** protected

GridViewColumn::$className

string = '' *[line 60]*

class name

- **Access** protected

GridViewColumn::$dataField

string = '' *[line 36]*

datefield

- **Access** protected

GridViewColumn::$events

EventCollection = null *[line 66]*

event collection

- **Access** protected

GridViewColumn::$footerText

string = '' *[line 54]*

footer text

- **Access** protected

GridViewColumn::$headerText

string = '' *[line 42]*

header text

- **Access** protected

GridViewColumn::$itemText

string = '' *[line 48]*

item text

- **Access** protected

GridViewColumn::$onclick

string = '' [line 90]

specifies the action to take on click events

- **Access** protected

GridViewColumn::$ondblclick

string = '' [line 96]

specifies the action to take on double click events

- **Access** protected

GridViewColumn::$onmouseout

string = '' [line 84]

specifies the action to take on onmouseout events

- **Access** protected

GridViewColumn::$onmouseover

string = '' [line 78]

specifies the action to take on mouseover events

- **Access** protected

Constructor *void* function GridViewColumn::__construct($dataField, [$headerText = ''], [$itemText = ''], [$footerText = ''], [$className = '']) *[line 107]*

Function Parameters:

- *string* **$dataField** name of data field to bind column to
- *string* **$headerText** column header text
- *string* **$itemText** column item text (templating allowed)
- *string* **$footerText** column footer text
- *string* **$className** column css class name

- **Access** public

string function GridViewColumn::getRequestData() *[line 332]*

creates string of all vars currently in the request object seperated by an '&' used to preserve application state

- **Access** protected

void function GridViewColumn::onLoad() *[line 280]*

handle load events

- **Access** public

void function GridViewColumn::onPost(&$request) *[line 298]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** public

void function GridViewColumn::onRender() *[line 306]*

handle render events

- **Access** public

void function GridViewColumn::onRequest(&$request) *[line 289]*

Function Parameters:

- *array* **&$request** request data

handle request events

- **Access** public

Class GridViewColumnAjaxPostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function GridViewColumnAjaxPostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class GridViewColumnAjaxPostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function GridViewColumnAjaxPostEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class GridViewColumnCollection

[line 19]

Represents a Collection of GridViewColumn objects

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

bool function GridViewColumnCollection::add($item) *[line 82]*

Function Parameters:

- *GridViewColumn* **$item**

add GridViewColumn to Collection

- **Access** public

bool function GridViewColumnCollection::addBefore($item, $datafield) *[line 53]*
Function Parameters:

- *GridViewColumn* **$item**
- *string* **$datafield**

add GridViewColumn to Collection before

- **Access** public

int function GridViewColumnCollection::findColumn($dataField, $controlId) *[line 172]*
Function Parameters:

- *string* **$controlId** control id
- **$dataField**

returns index if value is found in collection

- **Access** public

GridViewColumn function GridViewColumnCollection::itemAt($index) *[line 102]*
Function Parameters:

- *int* **$index** index of GridViewColumn

return GridViewColumn at a specified index

- **Access** public

void function GridViewColumnCollection::onLoad() *[line 113]*

handle load events

- **Access** public

void function GridViewColumnCollection::onPost(&$request) *[line 143]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** public

void function GridViewColumnCollection::onRender() *[line 157]*

handle render events

- **Access** public

void function GridViewColumnCollection::onRequest(&$request) *[line 128]*

Function Parameters:

- *array* **&$request** request data

handle request events

- **Access** public

Class GridViewColumnPostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function GridViewColumnPostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class GridViewColumnPostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function GridViewColumnPostEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class GridViewControlBase

[line 21]

Represents a GridView control

- **Package** PHPRum
- **Sub-Package** Web

- **Property** bool $ajaxPostBack: specifies whether to perform ajax postback on change, Default is false
- **Property** bool $escapeOutput: Specifies whether to escape the output
- **Abstract Element**

GridViewControlBase::$ajaxPostBack

string = false *[line 39]*

event request parameter

- **Access** protected

GridViewControlBase::$escapeOutput

bool = true *[line 45]*

determines whether to escape the output

- **Access** protected

GridViewControlBase::$parameter

string = '' *[line 27]*

event request parameter

- **Access** protected

GridViewControlBase::$pkey

string = '' *[line 33]*

primary key

- **Access** protected

Constructor *void* function GridViewControlBase::__construct($dataField, $pkey, [$parameter = ''], [$headerText = ''], [$footerText = ''], [$className = ''], $value) *[line 76]*
Function Parameters:

- *string* **$dataField** field name
- *string* **$pkey** primary key
- *string* **$value** value of Control
- *string* **$parameter** parameter
- *string* **$headerText** header text
- *string* **$footerText** footer text
- *string* **$className** css class name

- **Access** public

string function GridViewControlBase::escape($string) *[line 208]*
Function Parameters:

- *type* **$string** string to escape

escape

- **Access** protected

string function GridViewControlBase::getItemText($dataField, $parameter, $params) *[line 229]*
Function Parameters:

- *string* **$dataField**
- *string* **$parameter**
- *string* **$params**

get item text

- **Abstract Element**
- **Access** protected

void function GridViewControlBase::onPost(&$request) *[line 171]*
Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** public

void function GridViewControlBase::onRender(&$request) *[line 195]*
Function Parameters:

- *array* **&$request** request data

handle request events

- **Access** public

void function GridViewControlBase::onRequest(&$request) *[line 154]*

Function Parameters:

- *array* **&$request** request data

handle request events

- **Access** public

Class GridViewDropDownMenu

[line 18]

Represents a GridView DropDownMenu

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

GridViewDropDownMenu::$items

array = [line 24]

items

- **Access** protected

Constructor *void* function GridViewDropDownMenu::__construct($dataField, $pkey, $values, [$parameter = ''], [$headerText = ''], [$footerText = ''], [$className = ''], $value) *[line 38]*

Function Parameters:

- *string* **$dataField** field name
- *string* **$pkey** primary key
- *array* **$values** list values
- *string* **$value** value of Control
- *string* **$parameter** parameter
- *string* **$headerText** header text
- *string* **$footerText** footer text
- *string* **$className** css class name

- **Access** public

string function GridViewDropDownMenu::getItemText($dataField, $parameter, $params) *[line 53]*

Function Parameters:

- *string* **$dataField**
- *string* **$parameter**
- *string* **$params**

get item text

- **Access** protected

Class GridViewImage

[line 18]

Represents a GridView image

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function GridViewImage::__construct($dataField, [$headerText = ''], [$src = ''], [$alt = '\'\''], [$footerText = ''], [$className = '']) *[line 29]*

Function Parameters:

- *string* **$dataField** field name
- *string* **$headerText** header text
- *string* **$src** src (templating allowed)
- *string* **$alt** alt text
- *string* **$footerText** footer text
- *string* **$className** css class name

- **Access** public

Class GridViewLink

[line 18]

Represents a GridView link

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function GridViewLink::__construct($dataField, [$headerText = ''], [$url = ''], [$title = ''], [$target = ''], [$alt = ''], [$footerText = ''], [$className = '']) *[line 31]*

Function Parameters:

- *string* **$dataField** field name
- *string* **$headerText** header text
- *string* **$url** url (templating allowed)
- *string* **$title** link title (templating allowed)
- *string* **$target** link target
- *string* **$alt** alt text
- *string* **$footerText** footer text
- *string* **$className** css class name

- **Access** public

Class GridViewTextBox

[line 18]

Represents a GridView TextBox

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

GridViewTextBox::$size

int = 30 *[line 24]*

size

- **Access** protected

string function GridViewTextBox::getItemText($dataField, $parameter, $params) *[line 34]*

Function Parameters:

- *string* **$dataField**
- *string* **$parameter**
- *string* **$params**

get item text

- **Access** protected

Class HyperLink

[line 23]

Represents a Button Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $text: Specifies hyperlink text
- **Property** string $url: Specifies hyperlink URL
- **Property** string $target: Specifies hyperlink target
- **Property** string $src: Specifies hyperlink image source
- **Author** Darnell Shinbine

HyperLink::$src

string = '' *[line 47]*

specifies hyperlink image source

- **Access** protected

HyperLink::$target

string = '_self' *[line 41]*

specifies hyperlink target

- **Access** protected

HyperLink::$text

string = '' *[line 29]*

specifies hyperlink text

- **Access** protected

HyperLink::$url

string = '' *[line 35]*

specifies hyperlink URL

- **Access** protected

Constructor *void* function HyperLink::__construct($controlId, [$text = ''], [$url = ''], [$target = '_self']) *[line 57]*
Function Parameters:

- *string* **$controlId** Control Id
- *string* **$text** Button text
- **$url**

- **$target**

Constructor

- **Access** public

DomObject function HyperLink::getDomObject() *[line 149]*

getDomObject

returns a DomObject representing control

- **Access** public

Class InputAjaxChangeEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputAjaxChangeEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class InputAjaxChangeEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputAjaxChangeEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class InputAjaxPostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputAjaxPostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class InputAjaxPostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputAjaxPostEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class InputBase

[line 33]

Provides base functionality for Input Controls

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $defaultHTMLControlId: Specifies the id of the default html control
- **Property** string $dataField: Name of the data field in the datasource
- **Property** bool $autoPostBack: Specifies whether form will perform postback on change, Default is false
- **Property** bool $ajaxPostBack: specifies whether to perform ajax postback on change, Default is false
- **Property** bool $ajaxValidation: specifies whether to perform ajax validation, Default is false
- **Property** bool $readonly: Specifies whether control is readonly
- **Property** bool $disabled: Specifies whether the control is disabled
- **Property** string $label: Specifies control label
- **Property** string $tooltip: Specifies control tooltip
- **Property** int $tabIndex: Specifies the tab order if the control

- **Property** string $value: Gets or sets value of control
- **Property** bool $submitted: Specifies whether the data has been submitted
- **Property** bool $changed: Specifies whether the data has been changed
- **Property** array $validators: Array of validators
- **Author** Darnell Shinbine
- **Abstract Element**

InputBase::$ajaxPostBack

bool = false *[line 63]*

Specifies whether form will submit ajax postback on change, Default is false

- **Access** protected

InputBase::$ajaxValidation

bool = false *[line 69]*

Specifies whether form will submit ajax validation, Default is false

- **Access** protected

InputBase::$autoPostBack

bool = false *[line 57]*

Specifies whether form will submit postback on change, Default is false

- **Access** protected

InputBase::$changed

bool = false *[line 117]*

specifies whether the control has changed

- **Access** protected

InputBase::$dataField

string = '' *[line 51]*

Name of the data field in the datasource

- **Access** protected

InputBase::$defaultHTMLControlId

string = "" *[line 45]*

specifies the id of the default html control

- **Access** protected

InputBase::$disabled

bool = false *[line 81]*

Specifies whether the control is disabled, Default is false

- **Access** protected

InputBase::$enableViewState

bool = false *[line 39]*

specifies whether the server control persists its view state, Default is true

- **Access** protected

InputBase::$label

string = '' *[line 87]*

Specifies control label

- **Access** protected

InputBase::$readonly

bool = false *[line 75]*

Specifies whether the control is readonly, Default is false

- **Access** protected

InputBase::$submitted

bool = false *[line 111]*

Specifies whether the data has been submitted

- **Access** protected

InputBase::$tabIndex

int = 0 *[line 99]*

Specifies the tab order if the control

- **Access** protected

InputBase::$tooltip

string = '' *[line 93]*

specifies control tool tip

- **Access** protected

InputBase::$validators

ValidatorCollection = null *[line 123]*

contains a collection of validators

- **Access** protected

InputBase::$value

string = null *[line 105]*

Gets or sets value of control

- **Access** protected

Constructor *void* function InputBase::__construct($controlId, [$default = null]) *[line 142]*

Function Parameters:

- *string* **$controlId** Control Id
- *string* **$default** Default value

Constructor

The constructor sets attributes based on session data, triggering events, and is responcible for formatting the proper request value and garbage handling

- **Access** public

void function InputBase::addValidator(\\\$validator) *[line 315]*

Function Parameters:

- *ValidatorBase* **\\\$validator**

adds a validator to the control

- **Access** public

void function InputBase::error([$args = array()]) *[line 356]*
Function Parameters:

- *array* **$args** parameters

renders error message if control does not validate

- **Access** public

void function InputBase::fetchError([$args = array()]) *[line 368]*
Function Parameters:

- *array* **$args** parameters

returns the error message element

- **Access** public

void function InputBase::fillDataSource(\&$ds, $ds) *[line 397]*
Function Parameters:

- *\ArrayAccess* **$ds** data source to fill
- *ArrayAccess* **\&$ds**

update data source with Control value

- **Access** public

bool function InputBase::focus() *[line 303]*

sets focus to the control

- **Access** public

DomObject function InputBase::getDomObject() *[line 385]*

returns an input DomObject representing control

- **Access** public

DomObject function InputBase::getInputDomObject() *[line 429]*

returns an input DomObject representing control

- **Access** protected

bool function InputBase::onDataBind() *[line 502]*

bind control to datasource gets record from dataobject and sets the control value to datafield value

- **Access** protected

void function InputBase::onInit() *[line 516]*

called when control is initiated

- **Access** protected

void function InputBase::onLoad() *[line 527]*

called when control is loaded

- **Access** protected

void function InputBase::onLoadViewState(&$viewState) *[line 484]*

Function Parameters:

- *array* **&$viewState** session array

read view state from session

- **Access** protected

void function InputBase::onPost(&$request) *[line 578]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** protected

void function InputBase::onRequest(&$request) *[line 539]*

Function Parameters:

- *array* **&$request** request data

process the HTTP request array

- **Access** protected

void function InputBase::onSaveViewState(&$viewState) *[line 608]*

Function Parameters:

- *array* **&$viewState** session array

write view state to session

- **Access** protected

void function InputBase::onUpdateAjax() *[line 622]*

Event called on ajax callback

- **Access** protected

void function InputBase::readDataSource(\&$ds, $ds) *[line 412]*

Function Parameters:

- *\ArrayAccess* **$ds** data source to read
- *ArrayAccess* **\&$ds**

update Control value with data from the data source

- **Access** public

bool function InputBase::validate([&$errMsg = ''], [&$controlToFocus = null], $errMsg) *[line 327]*

Function Parameters:

- *string* **$errMsg** error message
- **&$errMsg**
- *InputBase* **&$controlToFocus**

validates control data, returns true on success

- **Access** public

Class InputChangeEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputChangeEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class InputChangeEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputChangeEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class InputPostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputPostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class InputPostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function InputPostEventHandler::__construct($callback) *[line 27]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class JSONServiceBase

[line 23]

This class handles all remote procedure calls for a web service

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $encoding: specifies the character set for the soap message, default is ISO-8859-1
- **Property** string $cache: specifies whether to cache the WSDL, default is WSDL_CACHE_NONE

- **Property** string $version: specifies the SOAP version, default is SOAP_1_2
- **Property** string $namespace: specifies the namespace, default is controller id
- **Author** Darnell Shinbine
- **Abstract Element**

JSONServiceBase::$options

string = null *[line 29]*

specifies encoding options

- **Access** protected

void function JSONServiceBase::configure() *[line 85]*

configure the web service

- **Access** protected

void function JSONServiceBase::handle(\\\&$request, &$request) *[line 97]*

Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

this method will handle the web service request

- **Access** public

Class Label

[line 20]

Represents a Button Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $text: Specifies button text
- **Author** Darnell Shinbine

Label::$text

string = '' *[line 26]*

specifies button text

- **Access** protected

Constructor *void* function Label::__construct($controlId, [$text = '']) *[line 36]*

Function Parameters:

- *string* **$controlId** Control Id
- *string* **$text** Button text

Constructor

- **Access** public

DomObject function Label::getDomObject() *[line 108]*

getDomObject

returns a DomObject representing control

- **Access** public

Class ListBox

[line 20]

Represents a ListBox Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** int $listSize: Size of listbox
- **Author** Darnell Shinbine

ListBox::$listSize

int = 6 *[line 26]*

Size of listbox, default is 6

- **Access** protected

DomObject function ListBox::getDomObject() *[line 75]*

returns a DomObject representing control

- **Access** public

void function ListBox::onUpdateAjax() *[line 162]*

Event called on ajax callback

- **Access** protected

Class ListControlBase

[line 23]

Provides base functionality for List Controls

- **Package** PHPRum
- **Sub-Package** Web
- **Property** ListItemCollection $items: Collection of list items
- **Property** bool $multiple: Specifies whether multiple selections are allowed
- **Property** string $textField: Specifies name of value text in datasource
- **Property** string $valueField: Specifies name of value field in datasource

- **Author** Darnell Shinbine
- **Abstract Element**

ListControlBase::$items

ListItemCollection = *[line 29]*

collection of items

- **Access** protected

ListControlBase::$multiple

bool = false *[line 35]*

Specifies whether multiple selections are allowed

- **Access** protected

ListControlBase::$textField

string = '' *[line 41]*

Specifies name of text field in datasource

- **Access** protected

ListControlBase::$valueField

string = '' *[line 47]*

Specifies name of value field in datasource

- **Access** protected

Constructor *void* function ListControlBase::__construct($controlId, [$default = null]) *[line 57]*
Function Parameters:

- *string* **$controlId** Control Id
- *string* **$default** Default value

Constructor

- **Access** public

void function ListControlBase::onDataBind(0) *[line 182]*
Function Parameters:

- *$default* **0** value

bind control to data

- **Access** protected

bool function ListControlBase::onLoad() *[line 213]*

called when control is loaded

- **Access** protected

void function ListControlBase::onLoadViewState(&$viewState, $viewState) *[line 165]*
Function Parameters:

- *object* **$viewState** session array
- *array* **&$viewState**

read view state from session

- **Access** protected

void function ListControlBase::onRequest(&$request) *[line 226]*
Function Parameters:

- *array* **&$request**

process the HTTP request array

- **Access** protected

bool function ListControlBase::validate([&$errMsg = ''], [&$controlToFocus = null], $errMsg) *[line 139]*
Function Parameters:

- *string* **$errMsg** error message
- **&$errMsg**
- *InputBase* **&$controlToFocus**

validates control data, returns true on success

- **Access** public

Class ListItemCollection

[line 19]

Represents a Collection of list items

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

void function ListItemCollection::add($key, $value) *[line 28]*

Function Parameters:

- *mixed* **$key** key
- *mixed* **$value** value

Adds a new key/item pair to a Dictionary object if key does not already exist

- **Access** public

bool function ListItemCollection::containsItem($value) *[line 40]*
Function Parameters:

- *mixed* **$value** value

Returns a Boolean value that indicates whether a specified value exists in the Dictionary object

- **Access** public

int function ListItemCollection::indexOf($key) *[line 59]*
Function Parameters:

- *mixed* **$key** key

returns index of key found in Dictionary

- **Access** public

int function ListItemCollection::indexOfItem($value, $key) *[line 79]*
Function Parameters:

- *mixed* **$key** key
- **$value**

returns index of item found in Dictionary

- **Access** public

Class LoginForm

[line 32]

Represents a LoginForm

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $invalidCredentialsMsg: Specifies the message when login failed
- **Property** string $disabledMsg: Specifies the message when account is disabled
- **Property** string $lockoutMsg: Specifies the message when account is locked out
- **Property** string $emailNotFoundMsg: Specifies the message when email address is not found
- **Property** string $passwordResetSentMsg: Specifies the message when password reset has been sent
- **Property** string $passwordResetMsg: Specifies the message when password has been reset
- **Property** string $successMsg: Specifies the success message
- **Property** string $emailFromAddress: Specifies the from address of the email message
- **Property** string $emailSubject: Specifies the subject line of the email message
- **Property** IMailClient $mailClient: Specifies the mail client to send the message with
- **Property** TextBox $username: username control
- **Property** TextBox $password: password control
- **Property** TextBox $new_password: new password control when reseting password
- **Author** Darnell Shinbine

LoginForm::$disabledMsg

string = 'Account is suspended, please contact the system administrator' *[line 44]*

specifies the message when account is disabled

- **Access** protected

LoginForm::$emailFromAddress

string = 'no-reply@localhost.local' *[line 80]*

Specifies the from address when email is sent

- **Access** protected

LoginForm::$emailNotFoundMsg

string = 'No account with that email address was found' *[line 56]*

specifies the message when email address does not exist

- **Access** protected

LoginForm::$emailSubject

string = 'Password reset confirmation' *[line 86]*

Specifies the subject line when email is sent

- **Access** protected

LoginForm::$invalidCredentialsMsg

string = 'Invalid credentials provided' *[line 38]*

specifies the message when login failed

- **Access** protected

LoginForm::$lockoutMsg

string = 'Account is locked out due to too many attempts, please try again later' *[line 50]*

specifies the message when account is locked out

- **Access** protected

LoginForm::$mailClient

IMailClient = *[line 92]*

Specifies the mail client for sending

- **Access** protected

LoginForm::$passwordResetMsg

string = 'Your account password has been reset' *[line 68]*

specifies the message when email address does not exist

- **Access** protected

LoginForm::$passwordResetSentMsg

string = 'Your account password has been sent to the email address specified' *[line 62]*

specifies the message when email address does not exist

- **Access** protected

LoginForm::$successMsg

string = 'You have successfully logged in' *[line 74]*

specifies the message on successful login

- **Access** protected

Constructor *void* function LoginForm::__construct($controlId) *[line 102]*
Function Parameters:

- *string* **$controlId** Control Id

Constructor

- **Access** public

DomObject function LoginForm::getDomObject() *[line 502]*

returns a DomObject representing control

- **Access** public

void function LoginForm::onInit() *[line 266]*

called when control is initiated

- **Access** protected

void function LoginForm::onRequest(&$request) *[line 318]*

Function Parameters:

- *array* **&$request** request array

process the HTTP request array

- **Access** protected

Class LoginFormSubmitEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function LoginFormSubmitEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class LoginFormSubmitEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function LoginFormSubmitEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class MasterView

[line 18]

Represents a MasterView Control

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function MasterView::__construct($controlId) *[line 35]*

Function Parameters:

- *string* **$controlId** Control Id

Constructor

sets the controlId and prepares the control attributes

- **Access** public

void function MasterView::content() *[line 65]*

render content

- **Access** public

string function MasterView::getContentAreaIdString() *[line 76]*

get content area id string

- **Access** public

void function MasterView::handleMasterViewPostEvents(&$request) *[line 161]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** public

void function MasterView::initMasterView() *[line 87]*

called when control is initiated

- **Access** public

void function MasterView::loadMasterView() *[line 125]*

called when all controls are loaded

- **Access** public

void function MasterView::loadMasterViewViewState(&$viewState) *[line 106]*
Function Parameters:

- *array* **&$viewState** session data

read view state from session

- **Access** public

void function MasterView::masterViewRequestProcessor(&$request) *[line 144]*
Function Parameters:

- *array* **&$request** request data

process the HTTP request array

- **Access** public

void function MasterView::saveMasterViewViewState(&$viewState) *[line 178]*
Function Parameters:

- *array* **&$viewState** session data

write view state to session

- **Access** public

Class MasterViewInitEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function MasterViewInitEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class MasterViewInitEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function MasterViewInitEventHandler::__construct($callback) *[line 27]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class MasterViewLoadEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function MasterViewLoadEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class MasterViewLoadEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function MasterViewLoadEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class Page

[line 20]

Represents an Page Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $onload: Specifies jscript events for page load
- **Author** Darnell Shinbine

Page::$contentType

string = 'text/html' *[line 26]*

specifies content-Type

- **Access** protected

Page::$linkElements

array = array() *[line 50]*

array of link elements

- **Access** protected

Page::$metaElements

array = array() *[line 44]*

array of meta elements

- **Access** protected

Page::$objectElements

array = array() *[line 56]*

array of object elements

- **Access** protected

Page::$onload

string = '' *[line 32]*

specify javascript to execute on page load

- **Access** protected

Page::$scriptElements

array = array() *[line 38]*

array of script elements

- **Access** protected

Constructor *void* function Page::__construct($controlId) *[line 77]*

Function Parameters:

- *string* **$controlId** Control Id

sets the controlId and prepares the control attributes

- **Access** public

int function Page::addLink($href, [$rel = 'stylesheet'], [$type = 'text/css'], [$media = 'all'], [$charset = 'iso-8859-1']) *[line 201]*

Function Parameters:

- *string* **$href** source
- *string* **$rel** relationship, default is stylesheet
- *string* **$type** type, default is text/css
- *string* **$media** media, default is 'all'
- *string* **$charset** character set, default is iso-8859-1

add link element

- **Access** public

int function Page::addMetaTag($name, $content) *[line 140]*

Function Parameters:

- *string* **$name** the meta tag name to add
- *string* **$content** the meta tag content value to add

add metatag

- **Access** public

int function Page::addScript($src, [$type = 'text/javascript'], [$language = 'Javascript'], [$charset = 'utf-8']) *[line 169]*
Function Parameters:

- *string* **$src** source
- *string* **$type** type, default is text/javascript
- *string* **$language** script language, default is 'Javascript'
- *string* **$charset** character set, default is iso-8859-1

add script element

- **Access** public

void function Page::clearAjaxBuffer() *[line 266]*

clear ajax output buffer

- **Access** public

string function Page::fetch([$args = array()]) *[line 289]*
Function Parameters:

- *array* **$args** widget parameters

returns an html string

- **Access** public

string function Page::fetchPageHeader() *[line 351]*

return page header

- **Access** protected

string function Page::getAjaxBuffer() *[line 253]*

return ajax output buffer as a string

- **Access** public

bool function Page::isAjaxBufferReady() *[line 277]*

specifies whether the buffer contains output

- **Access** public

void function Page::loadAjaxBuffer($output) *[line 230]*

Function Parameters:

- *string* **$output** output content

load content into ajax output buffer

- **Access** protected

void function Page::loadAjaxJScriptBuffer($javaScript) *[line 242]*
Function Parameters:

- *string* **$javaScript** output javascript content

load content into ajax jscript output buffer

- **Access** public

void function Page::onInit() *[line 311]*

Event called when control is initiated

- **Access** protected

void function Page::onPost(&$request) *[line 339]*
Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** protected

void function Page::onRequest(&$request) *[line 326]*

Function Parameters:

- *array* **&$request** request data

Event called when request is processed

- **Access** protected

Class PageControllerBase

[line 27]

This class handles all requests for a specific page. In addition provides access to a Page component to manage any WebControl components.

The PageControllerBase exposes 4 protected properties

- **Package** PHPRum
- **Sub-Package** Web
- **Property** Page $page: Contains an instance of the Page component
- **Property** string $theme: Specifies the theme for this page

  2 Additional public readonly properties are exposed

- **Property** bool $isPostBack: Specifies whether the request is a postback
- **Property** bool $isAjaxPostBack: Specifies whether the request is an ajax postback
- **Author** Darnell Shinbine
- **Abstract Element**

PageControllerBase::$page

Page = null *[line 33]*

Contains an instance of the Page component

- **Access** protected

PageControllerBase::$theme

string = '' *[line 39]*

Specifies the theme for this page

- **Access** protected

View function PageControllerBase::getView(\\\&$request, &$request) *[line 140]*

Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

return view component for rendering

- **Access** public

void function PageControllerBase::onLoad() *[line 127]*

Event called when object created

- **Access** protected

Class PagePostEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function PagePostEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class PagePostEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function PagePostEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class PageRequestEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web

- **Author** Darnell Shinbine

Constructor *void* function PageRequestEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class PageRequestEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function PageRequestEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class RadioButton

[line 20]

Represents a RadioButton Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** bool $checked: Specifies whether button is checked
- **Author** Darnell Shinbine

RadioButton::$checked

bool = false *[line 26]*

Specifies checked status

- **Access** protected

Constructor *void* function RadioButton::__construct($controlId, [$default = false]) *[line 39]*

Function Parameters:

- *string* **$controlId** Control Id
- *string* **$default** Default value

Constructor

The constructor sets attributes based on session data, triggering events, and is responsible for formatting the proper request value and garbage handling

- **Access** public

DomObject function RadioButton::getDomObject() *[line 93]*

returns a DomObject representing control

- **Access** public

bool function RadioButton::onLoad() *[line 119]*

called when control is loaded

- **Access** protected

void function RadioButton::onRequest(&$request) *[line 132]*

Function Parameters:

- *array* **&$request**

process the HTTP request array

- **Access** protected

Class RadioButtonList

[line 21]

Represents a RadioButtonList Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $onclick: Specifies the action to take on click events
- **Property** string $ondblclick: Specifies the action to take on double click events
- **Author** Darnell Shinbine

RadioButtonList::$multiple

mixed = false *[line 36]*

Size of listbox

- **Access** protected

RadioButtonList::$onclick

mixed = '' *[line 26]*

Specifies the action to take on click events

- **Access** protected

RadioButtonList::$ondblclick

mixed = '' *[line 31]*

Specifies the action to take on double click events

- **Access** protected

DomObject function RadioButtonList::getDomObject() *[line 101]*

returns a DomObject representing control

- **Access** public

bool function RadioButtonList::onLoad() *[line 184]*

called when control is loaded

- **Access** protected

Class RadioGroup

[line 18]

Represents a RadioGroup Control

- **Package** PHPRum
- **Sub-Package** Web

- **Author** Darnell Shinbine

Constructor *void* function RadioGroup::__construct($controlId, [$default = null]) *[line 30]*

Function Parameters:

- *string* **$controlId** Control Id
- *string* **$default** Default value

Constructor

The constructor sets attributes based on session data, triggering events, and is responcible for formatting the proper request value and garbage handling

- **Access** public

void function RadioGroup::add($control, &$control) *[line 42]*

Function Parameters:

- *RadioButton* **&$control** instance of a RadioButton
- *RadioButton* **$control**

adds child control to collection

- **Access** public

DomObject function RadioGroup::getDomObject() *[line 53]*

returns a DomObject representing control

- **Access** public

bool function RadioGroup::onLoad() *[line 81]*

called when control is loaded

- **Access** protected

void function RadioGroup::onRequest(&$request) *[line 118]*

Function Parameters:

- *array* **&$request** request data

process the HTTP request array

- **Access** protected

Class ReportDetailsEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web

- **Author** Darnell Shinbine

Constructor *void* function ReportDetailsEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class ReportDetailsEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportDetailsEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class ReportFooterEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportFooterEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class ReportFooterEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web

- **Author** Darnell Shinbine

Constructor *void* function ReportFooterEventHandler::__construct($callback) *[line 27]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class ReportGroup

[line 22]

Represents a ReportGroup

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $groupName: Contains the group name
- **Property** DataSet $dataSource: Contains the DataSet for the current group
- **Property** ReportView $report: Contains the ReportView object
- **Author** Darnell Shinbine

Constructor *void* function ReportGroup::__construct(&$report, $groupName, \\\&$ds, $controlId) *[line 50]*
Function Parameters:

- *string* **$controlId** Control Id
- *ReportView* **&$report**
- **$groupName**
- *DataSet* **\\\&$ds**

Constructor

- **Access** public

Class ReportGroupFooterEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportGroupFooterEvent::__construct([$groupName = 'Group']) *[line 27]*

Function Parameters:

- *string* **$groupName** group name

Constructor

- **Access** public

Class ReportGroupFooterEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportGroupFooterEventHandler::__construct($callback, [$groupName = 'Group']) *[line 28]*

Function Parameters:

- *string* **$callback** call back
- *string* **$groupName** group name

Constructor

- **Access** public

Class ReportGroupHeaderEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportGroupHeaderEvent::__construct([$groupName = 'Group']) *[line 27]*

Function Parameters:

- *string* **$groupName** group name

Constructor

- **Access** public

Class ReportGroupHeaderEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportGroupHeaderEventHandler::__construct($callback, [$groupName = 'Group']) *[line 28]*

Function Parameters:

- *string* **$callback** call back
- *string* **$groupName** group name

Constructor

- **Access** public

Class ReportHeaderEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportHeaderEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class ReportHeaderEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function ReportHeaderEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class ReportView

[line 21]

Represents a ReportView Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $title: Report title
- **Property** DataSet $groupDataSource: Contains the DataSet for the current group
- **Author** Darnell Shinbine

Constructor *void* function ReportView::__construct($controlId) *[line 62]*

Function Parameters:

- *string* **$controlId** Control Id

Constructor

- **Access** public

void function ReportView::addFilter($field, $operator, $value) *[line 190]*

Function Parameters:

- **$field**
- **$operator**
- **$value**

adds a filter to the report

- **Access** public

void function ReportView::addGrouping($field, [$ascending = true]) *[line 153]*
Function Parameters:

- **$field**
- **$ascending**

adds a grouping field to the report

- **Access** public

void function ReportView::addSorting($field, [$ascending = true]) *[line 179]*
Function Parameters:

- **$field**
- **$ascending**

adds a sorting field to the report

- **Access** public

string function ReportView::fetch([$array = array()], $args) *[line 206]*
Function Parameters:

- *array* **$args** widget parameters

- *array* **$array**

returns an html string

- **Access** public

DomObject function ReportView::getDomObject() *[line 243]*

returns a DomObject for rendering

- **Access** public

void function ReportView::getGrouping() *[line 273]*

return grouping array

- **Access** protected

void function ReportView::onDataBind(0) *[line 255]*

Function Parameters:

- *$default* **0** value

bind control to data

- **Access** protected

Class RESTWebServiceBase

[line 18]

This class handles all remote procedure calls for a REST web service

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine
- **Abstract Element**

RESTWebServiceBase::$options

string = null *[line 24]*

specifies encoding options

- **Access** protected

void function RESTWebServiceBase::configure() *[line 58]*

configure the server

- **Access** protected

void function RESTWebServiceBase::delete($args) *[line 52]*

Function Parameters:

- *array* **$args**

handle delete requests

- **Access** public

void function RESTWebServiceBase::get($args) *[line 31]*

Function Parameters:

- *array* **$args**

handle get requests

- **Access** public

void function RESTWebServiceBase::handle(\\\&$request, &$request) *[line 84]*

Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

this method will handle the web service request

- **Access** public

void function RESTWebServiceBase::post($args) *[line 38]*

Function Parameters:

- *array* **$args**

handle post requests

- **Access** public

void function RESTWebServiceBase::put($args) *[line 45]*

Function Parameters:

- *array* **$args**

handle put requests

- **Access** public

Class SOAPWebServiceBase

[line 23]

This class handles all remote procedure calls for a SOAP web service

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $encoding: specifies the character set for the soap message, default is ISO-8859-1
- **Property** string $cache: specifies whether to cache the WSDL, default is WSDL_CACHE_NONE
- **Property** string $version: specifies the SOAP version, default is SOAP_1_2
- **Property** string $namespace: specifies the namespace, default is controller id
- **Author** Darnell Shinbine
- **Abstract Element**

SOAPWebServiceBase::$cache

string = WSDL_CACHE_NONE *[line 35]*

specifies whether to cache the WSDL

- **Access** protected

SOAPWebServiceBase::$encoding

string = 'ISO-8859-1' *[line 29]*

specifies the character set for the soap message

- **Access** protected

SOAPWebServiceBase::$namespace

string = *[line 47]*

specifies the namespace

- **Access** protected

SOAPWebServiceBase::$version

string = SOAP_1_2 *[line 41]*

specifies the SOAP version

- **Access** protected

SOAPWebServiceBase::$xreserved_methods

```
array = array(
        '__construct',
        '__set',
        '__get',
        'getView',
        'getRoles',
        'getCacheId',
        'login',
        'configure',
        'handle',
        'getWSDL',
        'getWSDLURL') [line 53]
```

array of reserved methods

- **Static**
- **Access** protected

void function SOAPWebServiceBase::configure() *[line 136]*

configure the web service

- **Access** protected

void function SOAPWebServiceBase::handle(\\\&$request, &$request) *[line 151]*

Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

this method will handle the web service request

- **Access** public

Class TextArea

[line 18]

Represents a TextBox Control

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

TextArea::$multiline

bool = true *[line 24]*

Specifies whether textbox will accept multiple lines of text, default is true

- **Access** protected

Class TextBox

[line 28]

Represents a TextBox Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** bool $multiline: Specifies whether textbox will accept multiple lines of text
- **Property** bool $mask: Specifies whether to characters will be masked
- **Property** int $maxLength: Specifies Max Length of value when defined
- **Property** bool $disableAutoComplete: Specifies whether to disable the browsers auto complete feature
- **Property** bool $disableEnterKey: Specifies whether to disable the enter key
- **Property** int $size: Specifies the size of a textbox
- **Property** int $rows: Specifies that number of rows in multiline textbox
- **Property** int $cols: Specifies that number of columns in multiline textbox
- **Property** string $watermark: Specifies an optional watermark that is displayed when the control is empty
- **Author** Darnell Shinbine

TextBox::$cols

int = 60 *[line 76]*

Specifies that number of columns in multiline textbox, default is 60

- **Access** protected

TextBox::$disableAutoComplete

bool = false *[line 52]*

Specifies whether to disable the browsers auto complete feature, default is false

- **Access** protected

TextBox::$disableEnterKey

bool = false *[line 58]*

Specifies whether to disable the enter key, default is false

- **Access** protected

TextBox::$mask

bool = false *[line 40]*

Specifies whether to characters will be masked, default is false

- **Access** protected

TextBox::$maxLength

int = 0 *[line 46]*

Max Length of value when set to non zero, default is 0

- **Access** protected

TextBox::$multiline

bool = false *[line 34]*

Specifies whether textbox will accept multiple lines of text, default is false

- **Access** protected

TextBox::$rows

int = 5 *[line 70]*

Specifies that number of rows in multiline textbox, default is 5

- **Access** protected

TextBox::$size

int = 30 *[line 64]*

Specifies the size of a textbox, default is 30

- **Access** protected

TextBox::$watermark

string = '' *[line 82]*

Specifies an optional watermark that is displayed when the control is empty

- **Access** protected

void function TextBox::fillDataSet(\\\&$ds, $ds) *[line 174]*

Function Parameters:

- *DataSet* **$ds** DataSet to fill
- *DataSet* **\\\&$ds**

update DataSet with Control data

- **Access** public

DomObject function TextBox::getDomObject() *[line 188]*

returns a DomObject representing control

- **Access** public

bool function TextBox::onLoad() *[line 315]*

called when control is loaded

- **Access** protected

void function TextBox::onRequest(&$request, $request) *[line 339]*

Function Parameters:

- *object* **$request** HTTPRequest Object
- *array* **&$request**

process the HTTP request array

- **Access** protected

Class TimeSelector

[line 22]

handles date control element creation (string) abstracts away the presentation logic and data access layer the server-side control for WebWidgets

- **Package** PHPRum
- **Sub-Package** Web
- **Property** bool $allowNull: specifies whether to allow null values
- **Author** Darnell Shinbine

TimeSelector::$allowNull

mixed = false *[line 28]*

specifies whether to allow nulls

- **Access** protected

void function TimeSelector::getDomObject(0) *[line 145]*
Function Parameters:

- *none* **0**

returns widget object

- **Access** public

void function TimeSelector::onLoad() *[line 74]*
called when control is loaded

- **Access** protected

void function TimeSelector::onRequest(&$httpRequest) *[line 88]*
Function Parameters:

- *array* **&$httpRequest**

process the HTTP request array

- **Access** protected

Class TreeNode

[line 25]

Represents a tree node

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $id: Node id
- **Property** string $textOrHtml: Node text
- **Property** string $imgSrc: Specifies the path of the icon image
- **Property** bool $expanded: Specifies if the node is expanded by default
- **Property** TreeNodeCollection $childNodes: collection of child nodes
- **Property** TreeNode $parentNode: Contains parent node
- **Author** Darnell Shinbine

TreeNode::$childNodes

TreeNodeCollection = null *[line 55]*

collection of child nodes

- **Access** protected

TreeNode::$expanded

bool = false *[line 49]*

Specifies if the node is expanded by default

- **Access** protected

TreeNode::$id

string = '' *[line 31]*

node id

- **Access** protected

TreeNode::$imgSrc

string = '' *[line 43]*

specifies the path of the icon image

- **Access** protected

TreeNode::$parentNode

TreeNode = null *[line 61]*

contains parent node

- **Access** protected

TreeNode::$textOrHtml

string = '' *[line 37]*

node text

- **Access** protected

Constructor *void* function TreeNode::__construct($id, [$textOrHtml = ''], [$expanded = false], [$imgSrc = null]) *[line 71]*

Function Parameters:

- *string* **$id** TreeNode Id
- *string* **$textOrHtml** TreeNode text
- *bool* **$expanded** Specifies whether to expand the TreeNode
- *string* **$imgSrc** Specifies the icon image source

- **Access** public

bool function TreeNode::addChild($treeNode) *[line 312]*

Function Parameters:

- *TreeNode* **$treeNode** reference to an instance of a TreeNode object

adds child TreeNode to current TreeNode

- **Access** public

void function TreeNode::collapse() *[line 459]*

collapse node

- **Access** public

void function TreeNode::collapseAll() *[line 470]*

collapses all children

- **Access** public

TreeNode function TreeNode::createChild([$textOrHTML = ''], [$expanded = false], [$imgSrc = '']) *[line 294]*

Function Parameters:

- *string* **$textOrHTML** string value of node
- *bool* **$expanded** expand node by default
- *string* **$imgSrc** path to icon image

creates a child TreeNode in the current TreeNode

- **Access** public

void function TreeNode::expand() *[line 433]*

expand node

- **Access** public

void function TreeNode::expandAll() *[line 444]*

expands all children

- **Access** public

TreeNode function TreeNode::findChildNode($id) *[line 261]*

Function Parameters:

- *string* **$id** id of TreeNode

returns TreeNode if node is found in Collection

- **Access** public

XMLEntityCollection function TreeNode::getAncestry() *[line 414]*

returns node ancestry

- **Access** public

TreeNode function TreeNode::getChild($id) *[line 345]*

Function Parameters:

- *string* **$id** id of node

returns child node by Id

- **Access** public

TreeNodeCollection function TreeNode::getChildren() *[line 364]*

returns array of sibling nodes

- **Access** public

TreeNode function TreeNode::getParent() *[line 403]*

get parent node

- **Access** public

XMLEntityCollection function TreeNode::getSiblings() *[line 375]*

returns array of sibling nodes

- **Access** public

bool function TreeNode::hasChild($id) *[line 325]*

Function Parameters:

- *string* **$id** id of node

returns true if child node exists

- **Access** public

bool function TreeNode::hasChildNode($id) *[line 231]*
Function Parameters:

- *string* **$id** id of TreeNode

returns true if node is found in Collection

- **Access** public

void function TreeNode::loadViewState(&$viewState, $viewState) *[line 151]*
Function Parameters:

- *object* **$viewState** session array
- *array* **&$viewState**

read view state from session

- **Access** public

void function TreeNode::requestProcessor($treeViewControlId, &$request, $request) *[line 181]*
Function Parameters:

- *string* **$treeViewControlId** TreeView Control Id
- *array* **$request** Request Parameter Collection
- *array* **&$request**

process the HTTP request array

- **Access** public

void function TreeNode::saveViewState(&$viewState, $viewState) *[line 209]*
Function Parameters:

- *object* **$viewState** session array
- *array* **&$viewState**

write view state to session

- **Access** public

void function TreeNode::setParent(&$parentNode, $treeNode) *[line 392]*
Function Parameters:

- *TreeNode* **$treeNode** reference to an instance of a TreeNode object
- *TreeNode* **&$parentNode**

set parent node

- **Access** public

Class TreeNodeCollection

[line 19]

Represents a Collection of TreeNode objects

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

bool function TreeNodeCollection::add($item) *[line 52]*

Function Parameters:

- *TreeNode* **$item**

add TreeNode to Collection

- **Access** public

bool function TreeNodeCollection::contains($id) *[line 84]*

Function Parameters:

- *string* **$id** TreeNode id

returns true if array item is found

- **Access** public

int function TreeNodeCollection::indexOf($id) *[line 103]*
Function Parameters:

- *string* **$id** TreeNode id

returns index if value is found in collection

- **Access** public

TreeNode function TreeNodeCollection::itemAt($index) *[line 72]*
Function Parameters:

- *int* **$index** index of TreeNode

return TreeNode at a specified index

- **Access** public

Class TreeView

[line 22]

Represents a TreeView Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** bool $showIcons: Specifies whether to show icons in tree
- **Property** bool $showRoot: Specifies whether to show the root node
- **Property** rootNode $rootNode: Contains the root node
- **Author** Darnell Shinbine

TreeView::$rootNode

TreeNode = null *[line 40]*

contains the root node

- **Access** protected

TreeView::$showIcons

bool = true *[line 28]*

specifies whether to show icons in tree, Default is true

- **Access** protected

TreeView::$showRoot

bool = true *[line 34]*

specifies whether to show the root node

- **Access** protected

TreeNode function TreeView::findTreeItem($id) *[line 130]*

Function Parameters:

- *string* **$id** id of node

traverses all tree nodes and returns true if a node is found

- **Access** public

DomObject function TreeView::getDomObject() *[line 145]*

returns a DomObject representing control

- **Access** public

DomObject function TreeView::getTreeNodeDomObject(&$treeNode, $treeNode) *[line 181]*

Function Parameters:

- *TreeNode* **$treeNode** TreeNode object
- *TreeNode* **&$treeNode**

returns DomObject for the TreeNode and all child TreeNodes

- **Access** protected

bool function TreeView::onLoad() *[line 328]*

called when control is loaded

- **Access** protected

void function TreeView::onLoadViewState(&$viewState, $viewState) *[line 314]*

Function Parameters:

- *object* **$viewState** session array
- *array* **&$viewState**

read view state from session

- **Access** protected

void function TreeView::onRequest(&$request) *[line 342]*

Function Parameters:

- *array* **&$request**

process the HTTP request array

- **Access** protected

void function TreeView::onSaveViewState(&$viewState, $viewState) *[line 360]*

Function Parameters:

- *object* **$viewState** session array
- *array* **&$viewState**

write view state to session

- **Access** protected

Class View

[line 24]

Represents a Page Control

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $contentType: Document content-type
- **Property** string $charset: Document character set
- **Property** string $template: Document template
- **Property** array $parameters: Document parameters
- **Property** MasterView $master: Document master page

- **Author** Darnell Shinbine

View::$charset

string = 'utf-8' *[line 36]*

specifies charset, default is utf-8

- **Access** protected

View::$contentType

string = 'text/plain' *[line 30]*

specifies content-Type, default is text/plain

- **Access** protected

View::$template

string = '' *[line 42]*

specifies view template

- **Access** protected

Constructor *void* function View::__construct($controlId) *[line 81]*
Function Parameters:

- *string* **$controlId** Control Id

sets the controlId and prepares the control attributes

- **Access** public

void function View::add($control, &$control) *[line 165]*
Function Parameters:

- *WebControlBase* **&$control** instance of a WebControl
- *WebControlBase* **$control**

adds child control to collection

- **Access** public

void function View::addHeader($header) *[line 288]*
Function Parameters:

- *string* **$header** header to add

add header

- **Access** public

void function View::assign($field, $value) *[line 178]*
Function Parameters:

- *string* **$field** var name
- *mixed* **$value** var value

assign template variables to the view

- **Access** public

string function View::fetch([$args = array()]) *[line 318]*

Function Parameters:

- *array* **$args** widget parameters

returns an html string

- **Access** public

void function View::fetchTemplate() *[line 441]*

return template

- **Access** protected

DomObject function View::getDomObject() *[line 306]*

returns a DomObject for rendering

- **Access** public

void function View::onInit() *[line 367]*

Event called when control is initiated

- **Access** protected

void function View::onLoad() *[line 381]*

Event called when page and all controls are loaded

- **Access** protected

void function View::onPost(&$request) *[line 411]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** protected

void function View::onRequest(&$request) *[line 396]*

Function Parameters:

- *array* **&$request** request data

Event called when request is processed

- **Access** protected

void function View::render([$args = array()]) *[line 349]*

Function Parameters:

- *array* **$args** widget parameters

This method will render the control to the html canvass

- **Access** public

void function View::sendHeaders() *[line 425]*

send headers

- **Access** protected

void function View::setData($data) *[line 269]*

Function Parameters:

- *string* **$data** raw data

set data

- **Access** public

void function View::setMaster($master) *[line 239]*

Function Parameters:

- *MasterView* **$master** master Page

assign master view to this view

- **Access** public

Class WebControlAttributeCollection

[line 19]

Represents a Collection of WebControl attributes

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

void function WebControlAttributeCollection::append($name, $value) *[line 53]*

Function Parameters:

- *string* **$name** name of attribute
- *string* **$value** value of attribute

append attribute of xml node overwrite attribute with same name

- **Access** public

Class WebControlBase

[line 33]

Provides a web-based server-side UI to manage the HTML that ends up on client machine by abstracting away the presentation logic and data access layer

- **Package** PHPRum
- **Sub-Package** Web
- **Property** string $controlId: Control Id
- **Property** bool $enableViewState: Specifies whether to enable view state preservation
- **Property** bool $visible: Specifies whether control is visible
- **Property** bool $escapeOutput: Specifies whether to escape the output
- **Property** string $ajaxCallback: Specifies the ajax callback uri (optional)
- **Property** string $ajaxEventHandler: Specifies the name of the ajax event handler javascript function (optional)
- **Property** string $ajaxHTTPRequest: Specifies the name of the ajax HTTPRequest object (optional)
- **Property** WebControlAttributeCollection $attributes: Collection of attributes
- **Property** WebControlCollection $controls: Collection of child controls
- **Property** WebControlBase $parent: parent control
- **Property** DataSet $dataSource: Reference to data-source

- **Property** mixed $value: Control value
- **Property** EventCollection $events: event collection
- **Author** Darnell Shinbine
- **Abstract Element**

WebControlBase::$ajaxCallback

string = '' [line 57]

specifies the ajax callback uri (optional)

- **Access** protected

WebControlBase::$ajaxEventHandler

string = '' [line 63]

specifies the name of the ajax event handler javascript function (optional)

- **Access** protected

WebControlBase::$ajaxHTTPRequest

string = '' [line 69]

specifies the name of the ajax HTTPRequest object (optional)

- **Access** protected

WebControlBase::$attributes

WebControlAttributeCollection = null *[line 75]*

collection of arbitrary attributes (for rendering only)

- **Access** protected

WebControlBase::$controls

WebControlCollection = null *[line 81]*

collection of child controls

- **Access** protected

WebControlBase::$dataSource

DataSet = null *[line 87]*

instance of the datasource

- **Access** protected

WebControlBase::$enableViewState

bool = true *[line 39]*

specifies whether the server control persists its view state, Default is true

- **Access** protected

WebControlBase::$escapeOutput

bool = true *[line 51]*

determines whether to escape the output

- **Access** protected

WebControlBase::$events

EventCollection = null *[line 93]*

event collection

- **Access** protected

WebControlBase::$visible

bool = true *[line 45]*

determines whether the control is visible, Default is true

- **Access** protected

Constructor *void* function WebControlBase::__construct($controlId) *[line 120]*
Function Parameters:

- *string* **$controlId** Control Id

sets the controlId and prepares the control attributes

- **Access** public

void function WebControlBase::addControl($control, &$control) *[line 375]*

Function Parameters:

- *WebControlBase* **&$control** instance of a WebControl
- *WebControlBase* **$control**

add a child control to collection

- **Access** protected

void function WebControlBase::bind(\\\$ds, $ds) *[line 330]*

Function Parameters:

- *\ArrayAccess* **$ds** data source to attach
- *ModelBase* **\\\$ds**

bind a data source to control

- **Access** public

DomObject function WebControlBase::createDomObject($name) *[line 891]*

Function Parameters:

- **$name**

returns the default DomObject

- **Access** protected

string function WebControlBase::escape($string) *[line 873]*
Function Parameters:

- *type* **$string** string to escape

escape

- **Access** protected

string function WebControlBase::fetch([$args = array()]) *[line 730]*
Function Parameters:

- *array* **$args** widget parameters

returns an html string

- **Access** public

WebControlBase function WebControlBase::findControl($controlId) *[line 494]*

Function Parameters:

- *string* **$controlId** control id

returns WebControl if control is found in Collection

- **Access** public

WebControlBase function WebControlBase::getControl($controlId) *[line 445]*

Function Parameters:

- *string* **$controlId** id of control

returns reference to child control

- **Access** public

DomObject function WebControlBase::getDomObject() *[line 776]*

returns a DomObject representing control

- **Abstract Element**
- **Access** public

string function WebControlBase::getHTMLControlId() *[line 754]*

creates controlId string for html

- **Access** public

string function WebControlBase::getHTMLControlIdString() *[line 765]*

creates controlId string for html

- **Access** public

WebControlBase function WebControlBase::getParentByType($type) *[line 526]*

Function Parameters:

- *string* **$type** type

returns 1st parent object by type

- **Access** public

string function WebControlBase::getParentString() *[line 551]*

returns string of parent id's seperated by .

- **Access** public

string function WebControlBase::getQueryString([$queryString = '']) *[line 911]*

Function Parameters:

- *string* **$queryString** query string

creates string of all vars currently in the request object seperated by an '&' used to preserve application state

- **Access** protected

string function WebControlBase::getRequestData() *[line 948]*

creates string of all vars currently in the request object seperated by an '&' used to preserve application state

- **Access** protected

void function WebControlBase::handlePostEvents(&$request) *[line 664]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** public

bool function WebControlBase::hasControl($controlId) *[line 465]*

Function Parameters:

- *string* **$controlId** control id

returns Webtrue if control is found in Collection

- **Access** public

void function WebControlBase::init() *[line 569]*

called when control is initiated

- **Access** public

bool function WebControlBase::isControl($controlId) *[line 433]*

Function Parameters:

- *string* **$controlId** control id

compares control id, returns true if control id matches

- **Access** public

void function WebControlBase::load(&$request) *[line 591]*

Function Parameters:

- *array* **&$request** request data

called when all controls are loaded

- **Access** public

void function WebControlBase::loadViewState(&$viewState) *[line 611]*

Function Parameters:

- *array* **&$viewState** session data

read view state from session

- **Access** public

void function WebControlBase::onDataBind() *[line 833]*

Event called when DataBind is called

- **Access** protected

void function WebControlBase::onInit() *[line 798]*

Event called when control is initiated

- **Access** protected

void function WebControlBase::onLoad(&$request) *[line 807]*

Function Parameters:

- *array* **&$request** request data

Event called when page and all controls are loaded

- **Access** protected

void function WebControlBase::onLoadViewState(&$viewState) *[line 790]*

Function Parameters:

- *array* **&$viewState** session data

Event called when view state is loaded

- **Access** protected

void function WebControlBase::onPost(&$request) *[line 825]*

Function Parameters:

- *array* **&$request** request data

handle post events

- **Access** protected

void function WebControlBase::onPreRender() *[line 841]*

Event called when control is ready for rendering

- **Access** protected

void function WebControlBase::onRequest(&$request) *[line 816]*

Function Parameters:

- *array* **&$request** request data

Event called when request is processed

- **Access** protected

void function WebControlBase::onSaveViewState(&$viewState) *[line 851]*

Function Parameters:

- *array* **&$viewState** session data

Event called when view state is written

- **Access** protected

void function WebControlBase::onUpdateAjax() *[line 859]*

Event called on ajax callback

- **Access** protected

void function WebControlBase::preRender() *[line 680]*

called before control is ready for rendering

- **Access** public

void function WebControlBase::refreshDataSource() *[line 355]*

refresh data source

- **Access** public

bool function WebControlBase::removeControl(&$webcontrol) *[line 394]*

Function Parameters:

- *WebControlBase* **&$webcontrol** instance of a WebControl

remove a child control

- **Access** public

void function WebControlBase::render([$args = array()]) *[line 743]*

Function Parameters:

- *array* **$args** widget parameters

This method will render the control to the html canvass

- **Access** public

bool function WebControlBase::replaceControl(&$webcontrol1, &$webcontrol2, &$oldWebControl, &$newWebControl) *[line 414]*

Function Parameters:

- *WebControlBase* **&$oldWebControl** instance of a WebControl
- *WebControlBase* **&$newWebControl** instance of a WebControl
- *WebControlBase* **&$webcontrol1**
- *WebControlBase* **&$webcontrol2**

replace a child control

- **Access** public

void function WebControlBase::requestProcessor(&$request) *[line 636]*

Function Parameters:

- *array* **&$request** request data

process the HTTP request array

- **Access** public

void function WebControlBase::saveViewState(&$viewState) *[line 700]*

Function Parameters:

- *array* **&$viewState** session data

write view state to session

- **Access** public

void function WebControlBase::setParent($control, &$control) *[line 311]*

Function Parameters:

- *WebControlBase* **&$control** instance of a WebControl
- *WebControlBase* **$control**

set parent control

- **Access** public

void function WebControlBase::updateAjax() *[line 652]*

updates the ui control asyncronously

- **Access** public

Class WebControlCollection

[line 19]

Represents a Collection of WebControl objects

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlCollection::__construct(&$parent, $parent) *[line 35]*

Function Parameters:

- *WebControlBase* **$parent** instance of a WebControlBase object
- *WebControlBase* **&$parent**

Constructor

- **Access** public

void function WebControlCollection::add($item) *[line 78]*

Function Parameters:

- *WebControlBase* **$item**

add WebControl to Collection

- **Access** public

bool function WebControlCollection::contains($controlId) *[line 155]*
Function Parameters:

- *string* **$controlId** control id

returns true if array item is found

- **Access** public

int function WebControlCollection::indexOf($controlId) *[line 174]*
Function Parameters:

- *string* **$controlId** control id

returns index if value is found in collection

- **Access** public

WebControlBase function WebControlCollection::itemAt($index) *[line 143]*
Function Parameters:

- *int* **$index** index of WebControlBase

return WebControl at a specified index

- **Access** public

bool function WebControlCollection::remove($item) *[line 131]*

Function Parameters:

- *WebControlBase* **$item**

remove item from collection

- **Access** public

void function WebControlCollection::replace($controlId, $item) *[line 100]*

Function Parameters:

- *string* **$controlId**
- *WebControlBase* **$item**

replace a WebControl in a Collection

- **Access** public

Class WebControlCreateEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlCreateEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class WebControlCreateEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlCreateEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebControlInitEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlInitEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class WebControlInitEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlInitEventHandler::__construct($callback) *[line 27]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebControlLoadEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlLoadEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class WebControlLoadEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlLoadEventHandler::__construct($callback) *[line 27]*

Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebControlPreRenderEvent

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlPreRenderEvent::__construct() *[line 26]*

Constructor

- **Access** public

Class WebControlPreRenderEventHandler

[line 19]

Provides event handling

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebControlPreRenderEventHandler::__construct($callback) *[line 27]*
Function Parameters:

- *string* **$callback** call back

Constructor

- **Access** public

Class WebServiceBase

[line 18]

This class handles all remote procedure calls for a web service

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine
- **Abstract Element**

WebServiceBase::$remoteProcedures

string = array() *[line 24]*

Contains an array of RPC method calls

- **Access** protected

WebServiceBase::$reserved_methods

array = array(
'__construct',
'__set',
'__get',
'getView',
'getRoles',
'getCacheId',
'configure',
'handle') *[line 36]*

array of reserved methods

- **Static**
- **Access** protected

WebServiceBase::$view

View = *[line 30]*

contains the View object

- **Access** protected

void function WebServiceBase::configure() *[line 81]*

configure the server

- **Access** protected

View function WebServiceBase::getView(\\\&$request, &$request) *[line 53]*

Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

return view component for rendering

- **Access** public

void function WebServiceBase::handle(\\\&$request, &$request) *[line 75]*
Function Parameters:

- *HTTPRequest* **&$request** HTTPRequest object
- *HTTPRequest* **\\\&$request**

this method will handle the web service request

- **Abstract Element**
- **Access** public

Class WebServiceMethod

[line 18]

This class represents a remote procedure call

- **Package** PHPRum
- **Sub-Package** Web
- **Author** Darnell Shinbine

Constructor *void* function WebServiceMethod::__construct($function) *[line 29]*

Function Parameters:

- *string* **$function** function callback

Constructor

- **Access** public

array function WebServiceMethod::getFunction() *[line 62]*

return the function callback

- **Access** public

array function WebServiceMethod::getParameters() *[line 72]*

return all parameters

- **Access** public

void function WebServiceMethod::setFunction($function) *[line 40]*

Function Parameters:

- *string* **$function** function callback

set function

- **Access** public

void function WebServiceMethod::setParameters($name, $type) *[line 52]*

Function Parameters:

- *string* **$name** parameter name
- *string* **$type** parameter type

set parameters

- **Access** public

domobject.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

textnode.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

xmlentity.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

xmlentitycollection.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

xmlexception.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

xmlparser.class.php

- **Package** PHPRum
- **Author** Darnell Shinbine
- **Copyright** Copyright (c) 2011
- **Filesource** Source Code for this file
- **License** see

Class DomObject

[line 23]

Represents an XHTML DOM element

- **Package** PHPRum
- **Sub-Package** XML
- **Property** string $nodeName: name of XHTML DOM entity
- **Property** string $nodeValue: value of XHTML DOM entity
- **Property** string $innerHtml: inner html
- **Property** XMLEntityCollection $childNodes: XHTML DOM entity children
- **Author** Darnell Shinbine

DomObject::$innerHtml

string = '' [line 29]

innerHtml value

- **Access** protected

XMLEntity function DomObject::getChildById($id) *[line 119]*
Function Parameters:

- *string* **$id** id

returns a reference to a child element by id

- **Access** public

string function DomObject::getEntityValue([$charset = 'utf-8']) *[line 100]*
Function Parameters:

- *string* **$charset** character set

return formatted entity value

- **Access** protected

string function DomObject::getXMLString([$indent = 0]) *[line 141]*
Function Parameters:

- **$indent**

return as XML string

- **Access** public

void function DomObject::render([$args = array()]) *[line 130]*

Function Parameters:

- *array* **$args** node parameters

output html node

- **Access** public

Class TextNode

[line 18]

Represents a TextNode XML entity

- **Package** PHPRum
- **Sub-Package** XML
- **Author** Darnell Shinbine

Constructor *void* function TextNode::__construct([$text = '']) *[line 27]*

Function Parameters:

- *string* **$text** Node text

Constructor

- **Access** public

bool function TextNode::addChild($node, &$node) *[line 127]*

Function Parameters:

- *DomObject* **&$node** instance of a DomObject
- *XMLEntity* **$node**

add a reference to a child element

- **Access** public

void function TextNode::appendAttribute($name, $value) *[line 94]*

Function Parameters:

- *string* **$name** name of attribute
- *string* **$value** value of attribute

append attribute of xml node overwrite attribute with same name

- **Access** public

string function TextNode::getAttribute($name) *[line 105]*

Function Parameters:

- *string* **$name** name of attribute

get attribute of xml node

- **Access** public

XMLEntity function TextNode::getChild($index) *[line 138]*

Function Parameters:

- *int* **$index** index of element

returns a reference to a child element by index

- **Access** public

XMLEntity function TextNode::getChildByAttribute($attribute, $value, $index) *[line 160]*

Function Parameters:

- *string* **$index** name of element
- **$attribute**
- **$value**

returns a reference to a child element by element attribute

- **Access** public

XMLEntity function TextNode::getChildByName($name) *[line 149]*
Function Parameters:

- *string* **$name** name of element

returns a reference to a child element by element name

- **Access** public

XMLEntityCollection function TextNode::getChildrenByName($name) *[line 171]*
Function Parameters:

- *string* **$name** name of elements

returns all children by name

- **Access** public

string function TextNode::getXMLString([$indent = 0]) *[line 204]*
Function Parameters:

- **$indent**

return XML string

- **Access** public

void function TextNode::removeAttribute($name) *[line 116]*
Function Parameters:

- *string* **$name** name of attribute

remove an attribute from an xml node

- **Access** public

bool function TextNode::removeChild(&$node) *[line 182]*
Function Parameters:

- *DomObject* **&$node** instance of a DomObject

remove a reference to a child element

- **Access** public

bool function TextNode::replaceChild(&$node1, &$node2) *[line 194]*
Function Parameters:

- *DomObject* **&$node1** instance of a DomObject
- *DomObject* **&$node2** instance of a DomObject

replace a reference to a child element

- **Access** public

void function TextNode::setAttribute($name, $value) *[line 81]*

Function Parameters:

- *string* **$name** name of attribute
- *string* **$value** value of attribute

set attribute

- **Access** public

Class XMLEntity

[line 27]

XMLEntity

Represents an XML entity

- **Package** PHPRum
- **Sub-Package** XML
- **Property** string $name: name of XML entity

- **Property** string $value: value of XML entity
- **Property** string $cdata: XML cdata
- **Property** XMLEntityAttributeCollection $attributes: XML entity attributes
- **Property** XMLEntityCollection $children: XML entity children
- **Author** Darnell Shinbine

XMLEntity::$attributes

XMLEntityAttributeCollection = array() *[line 51]*

entity attributes

- **Access** protected

XMLEntity::$cdata

string = '' *[line 45]*

cdata value

- **Access** protected

XMLEntity::$children

XMLEntityCollection = null *[line 57]*

entity children

- **Access** protected

XMLEntity::$forceCloseTag

bool = false *[line 63]*

set to force close tag

- **Access** protected

XMLEntity::$name

string = '' *[line 33]*

entity name

- **Access** protected

XMLEntity::$value

string = '' *[line 39]*

entity value

- **Access** protected

Constructor *void* function XMLEntity::__construct([$name = '']) *[line 73]*
Function Parameters:

- *string* **$name** Name of XML entity

Constructor

- **Access** public

bool function XMLEntity::addChild($node, &$node) *[line 301]*

Function Parameters:

- *XMLEntity* **&$node** instance of an XMLEntity
- *XMLEntity* **$node**

add a reference to a child element

- **Access** public

void function XMLEntity::appendAttribute($name, $value) *[line 263]*

Function Parameters:

- *string* **$name** name of attribute
- *string* **$value** value of attribute

append attribute of xml node overwrite attribute with same name

- **Access** public

string function XMLEntity::fetch([$args = array()]) *[line 500]*

Function Parameters:

- *array* **$args** node parameters

return XML node as string

- **Access** public

XMLEntity function XMLEntity::findChildByName($name) *[line 327]*
Function Parameters:

- *string* **$name** name of element

returns a reference to a child element by element name

- **Access** public

string function XMLEntity::getAttribute($name) *[line 279]*
Function Parameters:

- *string* **$name** name of attribute

get attribute of xml node

- **Access** public

XMLEntity function XMLEntity::getChild($index) *[line 312]*
Function Parameters:

- *int* **$index** index of element

returns a reference to a child element by index

- **Access** public

XMLEntity function XMLEntity::getChildByAttribute($attribute, $value, $index) *[line 360]*
Function Parameters:

- *string* **$index** name of element
- **$attribute**
- **$value**

returns a reference to a child element by element attribute

- **Access** public

XMLEntity function XMLEntity::getChildByName($name) *[line 343]*
Function Parameters:

- *string* **$name** name of element

returns a reference to a child element by element name

- **Access** public

XMLEntityCollection function XMLEntity::getChildrenByName($name) *[line 377]*

Function Parameters:

- *string* **$name** name of elements

returns all children by name

- **Access** public

string function XMLEntity::getEntityValue([$charset = 'utf-8']) *[line 234]*

Function Parameters:

- *string* **$charset** character set

get formatted entity value

- **Access** protected

string function XMLEntity::getXMLString([$indent = 0]) *[line 435]*

Function Parameters:

- **$indent**

return XML string

- **Access** public

void function XMLEntity::removeAttribute($name) *[line 290]*
Function Parameters:

- *string* **$name** name of attribute

remove an attribute from an xml node

- **Access** public

bool function XMLEntity::removeChild(&$node) *[line 394]*
Function Parameters:

- *XMLEntity* **&$node** instance of a XMLEntity

remove a reference to a child element

- **Access** public

bool function XMLEntity::replaceChild(&$node1, &$node2) *[line 415]*
Function Parameters:

- *XMLEntity* **&$node1** instance of a XMLEntity
- *XMLEntity* **&$node2** instance of a XMLEntity

replace a reference to a child element

- **Access** public

void function XMLEntity::setAttribute($name, $value) *[line 250]*

Function Parameters:

- *string* **$name** name of attribute
- *string* **$value** value of attribute

set entity attribute

- **Access** public

Class XMLEntityCollection

[line 19]

Represents a Collection of XMLEntity objects

- **Package** PHPRum
- **Sub-Package** XML
- **Author** Darnell Shinbine

bool function XMLEntityCollection::add($item) *[line 51]*

Function Parameters:

- *XMLEntity* **$item**

add XMLEntity to Collection

- **Access** public

bool function XMLEntityCollection::contains($entity_name) *[line 83]*
Function Parameters:

- *string* **$entity_name** name of entity

returns true if array item is found

- **Access** public

int function XMLEntityCollection::indexOf($entity_name) *[line 102]*
Function Parameters:

- *string* **$entity_name** name of entity

returns index if value is found in collection

- **Access** public

XMLEntity function XMLEntityCollection::itemAt($index) *[line 71]*
Function Parameters:

- *int* **$index** index of ActiveRecord

return XMLEntity at a specified index

- **Access** public

Class XMLException

[line 19]

Represents an XML Exception

- **Package** PHPRum
- **Sub-Package** XML
- **Author** Darnell Shinbine

Class XMLParser

[line 21]

Provides a way to parse xml documents into XMLEntity objects

- **Package** PHPRum

- **Sub-Package** XML
- **Property** string $error: parse error
- **Property** array $data: xml data
- **Author** Darnell Shinbine

XMLParser::$data

array = array() *[line 33]*

array of xml data

- **Access** protected

XMLParser::$error

string = '' *[line 27]*

parse error

- **Access** protected

XMLEntity function XMLParser::parse($xml, $attrs) *[line 63]*
Function Parameters:

- *string* **$xml** raw xml data
- *bool* **$attrs** specifies whether to return attributes in result

parse xml data

Appendix A - Class Trees

Package PHPRum

AppConfiguration

- AppConfiguration

ApplicationBase

- ApplicationBase
  - ConsoleApplicationBase
    - Deploy
    - Get
    - Make
    - Migrate
    - Test
  - WebApplicationBase

AppMessage

- AppMessage

AppMessageType

- AppMessageType

AppState

- AppState

Authentication

- Authentication
    - BasicAuthentication
    - FormsAuthentication
    - WebServiceAuthentication

AuthenticationStatus

- AuthenticationStatus

Build

- Build

CacheBase

- CacheBase
    - DBCache
    - FileCache
    - MEMCache

ColumnCompare

- ColumnCompare

ColumnSchema

- ColumnSchema

ControllerBase

- ControllerBase
    - PageControllerBase

CredentialBase

- CredentialBase
    - TableCredential
    - UserCredential

DataAdapter

- DataAdapter
    - DirDataAdapter
    - MSSQLDataAdapter
    - MySQLDataAdapter
    - MySQLiDataAdapter
    - PDODataAdapter
    - TextDataAdapter

DatabaseSchema

- DatabaseSchema

DataSetType

- DataSetType

EventBase

- EventBase
  - ApplicationRunEvent
  - FormAjaxPostEvent
  - FormPostEvent
  - GridViewChangeOrderEvent
  - GridViewColumnAjaxPostEvent
  - GridViewColumnPostEvent
  - GridViewFilterEvent
  - GridViewSortEvent
  - InputAjaxChangeEvent
  - InputAjaxPostEvent
  - InputChangeEvent
  - InputPostEvent
  - LoginFormSubmitEvent
  - MasterViewInitEvent
  - MasterViewLoadEvent
  - PageControllerCreatePageEvent
  - PagePostEvent
  - PageRequestEvent
  - ReportDetailsEvent
  - ReportFooterEvent
  - ReportGroupFooterEvent
  - ReportGroupHeaderEvent
  - ReportHeaderEvent
  - WebApplicationHandleRequestEvent
  - WebApplicationLoadStateEvent
  - WebApplicationSaveStateEvent
  - WebControlCreateEvent
  - WebControlInitEvent
  - WebControlLoadEvent
  - WebControlPreRenderEvent

EventHandlerBase

- EventHandlerBase
  - ApplicationRunEventHandler

    - FormAjaxPostEventHandler
    - FormPostEventHandler
    - GridViewChangeOrderEventHandler
    - GridViewColumnAjaxPostEventHandler
    - GridViewColumnPostEventHandler
    - GridViewFilterEventHandler
    - GridViewSortEventHandler
    - InputAjaxChangeEventHandler
    - InputAjaxPostEventHandler
    - InputChangeEventHandler
    - InputPostEventHandler
    - LoginFormSubmitEventHandler
    - MasterViewInitEventHandler
    - MasterViewLoadEventHandler
    - PageControllerCreatePageEventHandler
    - PagePostEventHandler
    - PageRequestEventHandler
    - ReportDetailsEventHandler
    - ReportFooterEventHandler
    - ReportGroupFooterEventHandler
    - ReportGroupHeaderEventHandler
    - ReportHeaderEventHandler
    - WebApplicationHandleRequestEventHandler
    - WebApplicationLoadStateEventHandler
    - WebApplicationSaveStateEventHandler
    - WebControlCreateEventHandler
    - WebControlInitEventHandler
    - WebControlLoadEventHandler
    - WebControlPreRenderEventHandler

ExceptionBase

- ExceptionBase
    - ActiveRecordException
    - ArgumentOutOfRangeException
    - BadMemberCallException
    - BadMethodCallException
    - CacheException
    - DataAdapterException
    - DatabaseException
    - IndexOutOfRangeException
    - InvalidArgumentException
    - InvalidOperationException

- IOException
    - DirectoryNotFoundException
    - DirectoryNotWritableException
    - FileLoadException
    - FileNotFoundException
    - FileNotWritableException
- LoggerException
- MethodNotImplementedException
- MissingArgumentException
- QueryException
- SchemaException
- TransactionException
- TranslatorException
- TypeMismatchException
- XMLException

File

- File

FileSystem

- FileSystem

Folder

- Folder

ForwardMethodType

- ForwardMethodType

GridViewColumn

- GridViewColumn
    - GridViewControlBase
        - GridViewButton
        - GridViewCheckBox
        - GridViewDropDownMenu
        - GridViewTextBox
    - GridViewImage
    - GridViewLink

HTMLTestReporter

- HtmlReporter
    - HTMLTestReporter

HTTPRequest

- HTTPRequest

HTTPResponse

- HTTPResponse

HTTPWebRequest

- HTTPWebRequest

HTTPWebResponse

- HTTPWebResponse

IMailClient

- IMailClient

IteratorBase

- IteratorBase
    - CollectionBase
        - ActiveRecordCollection
        - AppMessageCollection
        - EventArgs
        - EventCollection
        - EventHandlerCollection
        - GridViewColumnCollection
        - KeyCollection
        - StringCollection
        - TreeNodeCollection
        - ValidatorCollection
        - ValueCollection
        - WebControlCollection
        - XMLEntityCollection
    - DictionaryBase
        - StringDictionary
            - AppSettingsCollection
            - ListItemCollection
            - WebControlAttributeCollection

LoggerBase

- LoggerBase
    - DBLogger
    - FileLogger

MailMessage

- MailMessage

MakeBase

- MakeBase
    - Controller
    - FormController
    - GridViewController
    - JSONWebService
    - ListController
    - Model
    - PageController
    - RESTWebService
    - SOAPWebService

MigrationBase

- MigrationBase

MigrationCompare

- MigrationCompare

Migrations

- Migrations

ModelBase

- ModelBase
  - DataSet
  - FormModelBase
    - ActiveRecordBase

PHPMailClient

- PHPMailClient

QueryBuilderBase

- QueryBuilderBase
  - MSSQLQueryBuilder
  - MySQLiQueryBuilder
  - MySQLQueryBuilder
  - PDOQueryBuilder

RelationshipType

- RelationshipType

ReportGroup

- ReportGroup

Roles

- Roles

Rum

- Rum

SendMailClient

- SendMailClient

Session

- Session

SMTPClient

- SMTPClient

SSHClientBase

- SSHClientBase
  - DeploymentBase

StreamBase

- StreamBase
    - FileStream
    - SocketStream

TableSchema

- TableSchema

TestCaseBase

- TestCaseBase
    - ControllerTestCaseBase
    - MigrationsTestCaseBase
    - UnitTestCaseBase

Tester

- Tester

TestReporter

- TextReporter
    - TestReporter

Timer

- Timer

TransactionBase

- TransactionBase
  - MSSQLTransaction
  - MySQLiTransaction
  - MySQLTransaction
  - PDOTransaction

TranslatorBase

- TranslatorBase
  - DBTranslator
  - FileTranslator

TreeNode

- TreeNode

ValidatorBase

- ValidatorBase
  - BooleanValidator
  - DateTimeValidator
  - EnumValidator
  - FileSizeValidator
  - FileTypeValidator
  - IntegerValidator
  - LengthValidator
  - MatchValidator
  - NumericValidator
  - PatternValidator

        - EmailValidator
        - URLValidator

    - RangeValidator
    - RequiredValidator
    - UniqueValidator

WebControlBase

- WebControlBase
    - Fieldset
    - Form
        - LoginForm

    - GridView
    - InputBase
        - Button
        - CheckBox
        - DateSelector
        - DateTimeSelector
        - FileBrowser
        - HyperLink
        - ListControlBase
            - ListBox
                - DropDownList

            - RadioButtonList
                - CheckBoxList

        - RadioButton
        - RadioGroup
        - TextBox
            - TextArea

        - TimeSelector

    - Label
    - ReportView
    - TreeView
    - View

        - MasterView
        - Page

WebServiceBase

- WebServiceBase
    - JSONServiceBase
    - RESTWebServiceBase
    - SOAPWebServiceBase

WebServiceMethod

- WebServiceMethod

XMLEntity

- XMLEntity
    - DomObject
    - TextNode

XMLParser

- XMLParser

Index

specifies the number of warnings to dump