

# patError



# Contents

<a href="#">Package patErrorHandler Procedural Elements</a>	2
<a href="#">patErrorHandlerDebug.php</a>	2
<a href="#">Package patErrorHandler Classes</a>	3
<a href="#">Class patErrorHandlerDebug</a>	3
<a href="#">Method arghDebug</a>	3
<a href="#">Method niceDie</a>	4
<a href="#">Method schstDebug</a>	4
<a href="#">Package patSessoin Procedural Elements</a>	7
<a href="#">package-config.php</a>	7
<a href="#">Package patError Procedural Elements</a>	9
<a href="#">patError.php</a>	9
<a href="#">patErrorManager.php</a>	10
<a href="#">Define PATERRORMANAGER_ERROR_CALLBACK_NOT_CALLABLE</a>	10
<a href="#">Define PATERRORMANAGER_ERROR_ILLEGAL_MODE</a>	10
<a href="#">Define PATERRORMANAGER_ERROR_ILLEGAL_OPTIONS</a>	10
<a href="#">Package patError Classes</a>	11
<a href="#">Class patError</a>	11
<a href="#">Var \$args</a>	11
<a href="#">Var \$backtrace</a>	12
<a href="#">Var \$class</a>	12
<a href="#">Var \$code</a>	12
<a href="#">Var \$file</a>	12
<a href="#">Var \$function</a>	13
<a href="#">Var \$info</a>	13
<a href="#">Var \$level</a>	13
<a href="#">Var \$line</a>	13
<a href="#">Var \$message</a>	14
<a href="#">Var \$type</a>	14
<a href="#">Constructor construct</a>	14
<a href="#">Method getBacktrace</a>	15
<a href="#">Method getCode</a>	15
<a href="#">Method getFile</a>	15
<a href="#">Method getInfo</a>	15
<a href="#">Method getLevel</a>	16
<a href="#">Method getLine</a>	16
<a href="#">Method getMessage</a>	16
<a href="#">Class patErrorManager</a>	17
<a href="#">Var \$errorClass</a>	17
<a href="#">Var \$errorExpects</a>	17
<a href="#">Var \$errorHandling</a>	18
<a href="#">Var \$errorIgnores</a>	18
<a href="#">Var \$errorLevels</a>	18

<a href="#">Method addIgnore</a>	19
<a href="#">Method clearExpect</a>	19
<a href="#">Method clearIgnore</a>	19
<a href="#">Method getErrorHandling</a>	19
<a href="#">Method getExpect</a>	20
<a href="#">Method getIgnore</a>	20
<a href="#">Method handleErrorCallback</a>	20
<a href="#">Method handleErrorDie</a>	21
<a href="#">Method handleErrorEcho</a>	21
<a href="#">Method handleErrorException</a>	21
<a href="#">Method handleErrorIgnore</a>	22
<a href="#">Method handleErrorTrigger</a>	22
<a href="#">Method handleErrorVerbose</a>	23
<a href="#">Method isError</a>	23
<a href="#">Method popExpect</a>	23
<a href="#">Method pushExpect</a>	24
<a href="#">Method raise</a>	24
<a href="#">Method raiseError</a>	25
<a href="#">Method raiseNotice</a>	25
<a href="#">Method raiseWarning</a>	26
<a href="#">Method registerErrorLevel</a>	26
<a href="#">Method removeIgnore</a>	27
<a href="#">Method setErrorClass</a>	27
<a href="#">Method setErrorHandling</a>	27
<a href="#">Method translateErrorLevel</a>	28
<a href="#">example cli error.php</a>	30
<a href="#">Function foo</a>	30
<a href="#">example error.php</a>	31
<a href="#">example errormanager customlevel.php</a>	32
<a href="#">example exception.php</a>	33
<a href="#">example expect.php</a>	34
<a href="#">example ignore.php</a>	35
<a href="#">Class NoticeException</a>	35
<a href="#">package.php</a>	36
<b><a href="#">Package patTemplate Procedural Elements</a></b>	38
<a href="#">autopackage2.php</a>	38
<b><a href="#">Appendices</a></b>	39
<a href="#">Appendix A - Class Trees</a>	40
<a href="#">patSessoin</a>	40
<a href="#">patError</a>	40
<a href="#">patErrorManager</a>	40
<a href="#">patTemplate</a>	40
<a href="#">Appendix D - Todo List</a>	41



# Package patErrorManager Procedural Elements

## patErrorHandlerDebug.php

**custom patErrorManager handler for the callback error handling mode**

Provides the pat-teams favourite error handlers.

\$Id: patErrorHandlerDebug.php 45 2008-07-19 17:08:08Z schst \$

- **Package** patErrorManager
- **Sub-Package** Debug
- **Access** public

# Package patErrorManager Classes

## Class patErrorHandlerDebug

[line 29]

custom patErrorManager handler for the callback error handling mode

- **Package** patErrorManager
- **Sub-Package** Debug
- **License** LGPL
- **Link** <http://www.php-tools.net>
- **Version** 0.1
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>
- **Author** Sebastian 'The Argh' Mordziol < [argh@php-tools.net](mailto:argh@php-tools.net)>

*object error function* patErrorHandlerDebug::arghDebug(\$error) [line 158]

### **Function Parameters:**

- *object error* **\$error** object

### **Error handler that outputs pretty debugging HTML**

Displays:

- Error level
- Error Message
- Error info
- Error file
- Error line
- plus the call stack that lead to the error

The output has been inspired by Schst's debug, updated for a designer's eye.

- **Access** public
- **Static**
- **Author** Sebastian Mordziol < [argh@php-tools.net](mailto:argh@php-tools.net)>

function patErrorHandlerDebug::niceDie(\$error) [*line 37*]

**Function Parameters:**

- *string* **\$error** The error message to display

**niceDie - outputs a nicely formatted version of the traditional die()**

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

*object error* function patErrorHandlerDebug::schstDebug(\$error) [*line 80*]

**Function Parameters:**

- *object error* **\$error** object

**error handler that outputs nice debugging HTML**

Displays:

- Error level
- Error Message
- Error info
- Error file
- Error line
- plus the call stack that lead to the error

The output has been inspired by Derick Rethan's xDebug.

- **Access** public
- **TODO** console output (no HTML)
- **Static**
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>



# Package patSessoin Procedural Elements

package-config.php

**package-config for patTemplate**

Config to to build PEAR packages

\$Id\$

- **Package** patSessoin
- **Sub-Package** Tools
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>



# Package patError Procedural Elements

## patError.php

**patError** error object used by the **patFormsError** manager as error messages container for precise error management.

\$Id: patError.php 45 2008-07-19 17:08:08Z schst \$

- **Package** patError
- **Access** public

## patErrorManager.php

**patErrorManager** main error management class used by pat tools for the application-internal error management. Creates patError objects for any errors for precise error management.

\$Id: patErrorManager.php 46 2008-07-19 17:15:44Z schst \$

- **Package** patError

PATERRORMANAGER\_ERROR\_CALLBACK\_NOT\_CALLABLE = 2 *[line 20]*

**error definition: callback function does not exist.**

PATERRORMANAGER\_ERROR\_ILLEGAL\_MODE = 3 *[line 25]*

**error definition: illegal error handling mode.**

PATERRORMANAGER\_ERROR\_ILLEGAL\_OPTIONS = 1 *[line 15]*

**error definition: illegal options.**

# Package patError Classes

## Class patError

[line 27]

**patError** error object used by the **patFormsError** manager as error messages container for precise error management.

\$Id: patError.php 45 2008-07-19 17:08:08Z schst \$

- **Package** patError
- **License** LGPL
- **Link** <http://www.php-tools.net>
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>
- **Author** Sebastian Mordziol < [argh@php-tools.net](mailto:argh@php-tools.net)>
- **Version** 0.3
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>
- **Access** public

### patError::\$args

*mixed = array()* [line 109]

stores the arguments the method that the error occurred in had received.

- **Access** protected

### patError::\$backtrace

*mixed* = false [*line 118*]

stores the complete debug backtrace (if your PHP version has the `debug_backtrace` function)

- **Access** protected

#### **patError::\$class**

*mixed* = " [*line 93*]

stores the name of the class (if any) the error occurred in.

- **Access** protected

#### **patError::\$code**

*string* = null [*line 42*]

stores the code of the error

- **Access** protected

#### **patError::\$file**

*mixed* = " [*line 69*]

stores the filename of the file the error occurred in.

- **Access** protected

### **patError::\$function**

```
mixed = " [line 85]
```

**stores the name of the method the error occurred in**

- **Access** protected

### **patError::\$info**

```
mixed = " [line 61]
```

**additional info that is relevant for the developer of the script (e.g. if a database connect fails, the dsn used) and that the end-user should not see.**

- **Access** protected

### **patError::\$level**

```
string = null [line 34]
```

**stores the error level for this error**

- **Access** protected

### **patError::\$line**

```
mixed = 0 [line 77]
```

**stores the line number the error occurred in.**

- **Access** protected

#### patError::\$message

*mixed* = null [*line 51*]

**stores the error message - this is the message that can also be shown the user if need be.**

- **Access** protected

#### patError::\$type

*mixed* = " [*line 101*]

**stores the type of error, as it is listed in the error backtrace**

- **Access** protected

Constructor function patError::\_\_construct(\$level, \$code, \$msg, [\$info = null]) [*line 129*]

#### **Function Parameters:**

- *int* **\$level** The error level (use the PHP constants E\_ALL, E\_NOTICE etc.).
- *string* **\$code** The error code from the application
- *string* **\$msg** The error message
- *string* **\$info** Optional: The additional error information.

**constructor - used to set up the error with all needed error details.**

- **Access** public

*array* function `patError::getBacktrace()` [*line 229*]

**get the backtrace**

This is only possible, if `debug_backtrace()` is available.

- **Access** public
- **See** [patError::\\$backtrace](#)

*string|integer* function `patError::getCode()` [*line 216*]

**recieve error code**

- **Access** public
- **See** [patError::\\$code](#)

*string* function `patError::getFile()` [*line 242*]

**get the filename in which the error ocured**

This is only possible, if `debug_backtrace()` is available.

- **Access** public
- **See** [patError::\\$file](#)

*mixed* function `patError::getInfo()` [*line 205*]

**retrieves the additional error information (information usually only relevant for developers)**

- **Access** public
- **See** [patError::\\$info](#)

*int* function patError::getLevel() [*line 181*]

**returns the error level of the error - corresponds to the PHP error levels (E\_ALL, E\_NOTICE...)**

- **Access** public
- **See** [patError::\\$level](#)

*integer* function patError::getLine() [*line 255*]

**get the line number in which the error occurred**

This is only possible, if debug\_backtrace() is available.

- **Access** public
- **See** [patError::\\$line](#)

*string* function patError::getMessage() [*line 193*]

**retrieves the error message**

- **Access** public
- **See** [patError::\\$message](#)

# Class patErrorManager

[line 42]

**patErrorManager** main error management class used by pat tools for the application-internal error management. Creates patError objects for any errors for precise error management.

- **Package** patError
- **Link** <http://www.php-tools.net>
- **TODO** implement expectError() to ignore an error with a certain code only once.
- **TODO** implement ignoreError() to ignore errors with a certain code
- **License** LGPL
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>
- **Version** 0.3
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>
- **Static**

## patErrorManager::\$errorClass

*mixed* = 'patError' [line 67]

### error class names

Stored in a variable allows to change during runtime

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

## patErrorManager::\$errorExpects

*mixed* = array() [line 79]

### expects errors

Store error-codes that will be ignored once

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

## patErrorManager::\$errorHandling

```
mixed = array(  
    E_NOTICE => array( 'mode' => 'echo'  
) , E_WARNING => array( 'mode' => 'echo' ) , E_ERROR => array( 'mode' => 'die' ) ) [line 48]
```

**global definitions needed to keep track of things when calling the patErrorManager static methods.**

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

## patErrorManager::\$errorIgnores

```
mixed = array() [line 73]
```

### ignore errors

Store error-codes that will be ignored forever

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

## patErrorManager::\$errorLevels

```
mixed = array(  
    E_NOTICE => 'Notice',  
    E_WARNING => 'Warning',  
    E_ERROR => 'Error'  
) [line 58]
```

### available error levels

Stored in a variable to keep them flexible

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

*boolean* function patErrorManager::addIgnore(\$codes) [*line 363*]

**Function Parameters:**

- *mixed* **\$codes** either an array of error code or a single code that will be ignored in future

## **add error codes to be ingored**

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

*boolean* function patErrorManager::clearExpect() [*line 475*]

## **empty list of errors to be ignored**

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

*boolean* function patErrorManager::clearIgnore() [*line 420*]

## **empty list of errors to be ignored**

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

*array* function patErrorManager::getErrorHandling(\$level) [*line 313*]

**Function Parameters:**

- *int* **\$level** The error level to retrieve. This can be any of PHP's own error levels, e.g. E\_ALL, E\_NOTICE...

**retrieves the current error handling settings for the specified error level.**

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

*array* function patErrorManager::getExpect() [*line 464*]

**recieve all registered error codes that will be ignored**

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

*array* function patErrorManager::getIgnore() [*line 409*]

**recieve all registered error codes that will be ignored**

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

*object* function patErrorManager::handleErrorCallback(\$error, \$options) [*line 620*]

**Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

**handleError: callback forward error to custom handler**

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)

*object* function patErrorManager::handleErrorDie(\$error, \$options) [*line 561*]

**Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

### handleError: die display error-message and die

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)

*object* function patErrorManager::handleErrorEcho(\$error, \$options) [*line 504*]

**Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

### handleError: Echo display error message

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)

*null* function patErrorManager::handleErrorException(\$error, \$options) [*line 636*]

**Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

## handleError: throw an exception

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)
- **Throws** Exception

*object* function patErrorManager::handleErrorIgnore(\$error, \$options) [*line 490*]

### **Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

## handleError: Ignore Does nothing

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)

*object* function patErrorManager::handleErrorTrigger(\$error, \$options) [*line 590*]

### **Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

## handleError: trigger trigger php-error

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)

*object* function patErrorManager::handleErrorVerbose(\$error, \$options) [*line 531*]

**Function Parameters:**

- *object* **\$error** patError-Object
- *array* **\$options** options for handler

**handleError: Verbose** display verbose output for developing purpose

- **Static**
- **Access** public
- **See** [patErrorManager::raise\(\)](#)

*boolean* function patErrorManager::isError(\$object, &\$object) [*line 90*]

**Function Parameters:**

- *mixed* **&\$object**
- **\$object**

**method for checking whether the return value of a pat application method is a pat error object.**

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

*boolean* function patErrorManager::popExpect() [*line 448*]

**remove top of error-codes from stack**

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

*boolean* function patErrorManager::pushExpect(\$codes) [*line 433*]

**Function Parameters:**

- *mixed* **\$codes** either an array of error code or a single code that will be ignored in future

**add expected errors to stack**

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

*mixed* function patErrorManager::raise(\$level, \$code, \$msg, [\$info = null]) [*line 165*]

**Function Parameters:**

- *int* **\$level** The error level - use any of PHP's own error levels for this: E\_ERROR, E\_WARNING, E\_NOTICE, E\_USER\_ERROR, E\_USER\_WARNING, E\_USER\_NOTICE.
- *string* **\$code** The application-internal error code for this error
- *string* **\$msg** The error message, which may also be shown the user if need be.
- *mixed* **\$info** Optional: Additional error information (usually only developer-relevant information that the user should never see, like a database DSN).

**creates a new patError object given the specified information.**

- **Static**
- **Access** public
- **TODO** implement 'simple' mode that returns just false (BC for patConfiguration)
- **TODO** either remove HTML tags and entities from output or test for environment!!! in shell is ugly!
- **See** [patError](#)

object function patErrorManager::raiseError(\$code, \$msg, [\$info = null]) [line 114]

**Function Parameters:**

- **string \$code** The application-internal error code for this error
- **string \$msg** The error message, which may also be shown the user if need be.
- **mixed \$info** Optional: Additional error information (usually only developer-relevant information that the user should never see, like a database DSN).

wrapper for the [raise\(\)](#) method where you do not have to specify the error level - a [patError](#) object with error level E\_ERROR will be returned.

- **Static**
- **Access** public
- **See** [patError](#)
- **See** [patErrorManager::raise\(\)](#)

object function patErrorManager::raiseNotice(\$code, \$msg, [\$info = null]) [line 148]

**Function Parameters:**

- **string \$code** The application-internal error code for this error
- **string \$msg** The error message, which may also be shown the user if need be.
- **mixed \$info** Optional: Additional error information (usually only developer-relevant information that the user should never see, like a database DSN).

wrapper for the [raise\(\)](#) method where you do not have to specify the error level - a [patError](#) object with error level E\_NOTICE will be returned.

- **Static**
- **Access** public
- **See** [patError](#)
- **See** [patErrorManager::raise\(\)](#)

object function patErrorManager::raiseWarning(\$code, \$msg, [\$info = null]) [line 131]

**Function Parameters:**

- *string* **\$code** The application-internal error code for this error
- *string* **\$msg** The error message, which may also be shown the user if need be.
- *mixed* **\$info** Optional: Additional error information (usually only developer-relevant information that the user should never see, like a database DSN).

wrapper for the [raise\(\)](#) method where you do not have to specify the error level - a [patError](#) object with error level E\_WARNING will be returned.

- **Static**
- **Access** public
- **See** [patError](#)
- **See** [patErrorManager::raise\(\)](#)

boolean function patErrorManager::registerErrorLevel(\$level, \$name) [line 218]

**Function Parameters:**

- *integer* **\$level** error level
- *string* **\$name** human-readable name

### register a new error level

This allows you to add custom error levels to the built-in

- E\_NOTICE
- E\_WARNING
- E\_NOTICE

You may use this level in subsequent calls to [raise\(\)](#). Error handling will be set to 'ignore' for the new level, you may change it by using [setErrorHandling\(\)](#).

You could be using PHP's predefined constants for error levels or any other integer value.

- **Static**
- **Access** public
- **Link** <http://www.php.net/manual/en/function.error-reporting.php>
- **See** [patErrorManager::raise\(\)](#), [patErrorManager::setErrorHandling\(\)](#)

*boolean* function patErrorManager::removeIgnore(\$codes) [*line 382*]

**Function Parameters:**

- **\$codes**

## removeIgnore

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

*boolean* function patErrorManager::setErrorClass(\$name) [*line 345*]

**Function Parameters:**

- *string* **\$name** classname

## setErrorClass

In order to autoload this class, the filename containing that class must be named like the class itself; with an appending ".php". Although the file must be stored in the same directory as patErrorManager.php (this file)

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

*mixed* function patErrorManager::setErrorHandler(\$level, \$mode, [\$options = null]) [*line 253*]

**Function Parameters:**

- *int* **\$level** The error level for which to set the error handling
- *string* **\$mode** The mode to use for the error handling.
- *mixed* **\$options** Optional: Any options needed for the given mode.

sets the way the `patErrorManager` will handle the different error levels. Use this if you want to override the default settings.

Error handling modes:

- ignore
- trigger
- verbose
- echo
- callback
- die

You may also set the error handling for several modes at once using PHP's bit operations. Examples:

- `E_ALL` = Set the handling for all levels
- `E_ERROR | E_WARNING` = Set the handling for errors and warnings
- `E_ALL ^ E_ERROR` = Set the handling for all levels except errors

- **Static**
- **Access** public
- See [patErrorManager::getErrorHandling\(\)](#)

*string* function `patErrorManager::translateErrorLevel($level)` [line 327]

**Function Parameters:**

- *integer* **\$level** error level

### translate an error level

returns the human-readable name for an error level, e.g. `E_ERROR` will be translated to 'Error'.

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



## example\_cli\_error.php

### Simple example for patError and patErrorManager

\$Id: example\_cli\_error.php 43 2008-07-19 16:39:17Z schst \$

- **Package** patError
- **Sub-Package** Examples
- **Link** <http://www.php-tools.net>
- **License** LGPL,
- **Copyright** PHP Application Tools
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>

function foo() [*line 62*]

**foo function raises an error**

include\_once '[patErrorHandlerDebug.php](#)' [*line 23*]

**patErrorManagerDebug - just an example**

include\_once '[../patErrorManager.php](#)' [*line 18*]

**patErrorManager class**

## example\_error.php

### Simple example for patError and patErrorManager

\$Id: example\_error.php 43 2008-07-19 16:39:17Z schst \$

- **Package** patError
- **Sub-Package** Examples
- **Link** <http://www.php-tools.net>
- **License** LGPL,
- **Copyright** PHP Application Tools
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>

include\_once ['patErrorHandlerDebug.php'](#)*[line 22]*

### patErrorManagerDebug - just an example

include\_once ['./patErrorManager.php'](#)*[line 17]*

### patErrorManager class

# example\_errormanager\_customlevel.php

## patErrorManager example

This example demonstrates how to add custom error levels

\$Id: example\_errormanager\_customlevel.php 46 2008-07-19 17:15:44Z schst \$

- **Package** patError
- **Sub-Package** Examples
- **Link** <http://www.php-tools.net>
- **License** LGPL,
- **Author** Stephan Schmidt < [argh@php-tools.net](mailto:argh@php-tools.net)>
- **Access** public

include '[../patErrorManager.php](#)*[line 20]*

## patErrorManager class

## example\_exception.php

### Simple example for patError and patErrorManager

\$Id: example\_exception.php 36 2005-03-06 12:37:47Z schst \$

- **Package** patError
- **Sub-Package** Examples
- **Link** <http://www.php-tools.net>
- **License** LGPL,
- **Copyright** PHP Application Tools
- **Author** Stephan Schmidt <schst@php-tools.net

include\_once ['./patErrorManager.php'](#)*[line 17]*

### patErrorManager class

## example\_expect.php

### Simple example for patError and patErrorManager

\$Id: example\_expect.php 21 2004-04-17 20:27:33Z schst \$

- **Package** patError
- **Sub-Package** Examples
- **Link** <http://www.php-tools.net>
- **License** LGPL,
- **Copyright** PHP Application Tools
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>

include\_once ['./patErrorManager.php'](#)*[line 17]*

### patErrorManager class

## example\_ignore.php

### Simple example for patError and patErrorManager

\$Id: example\_ignore.php 21 2004-04-17 20:27:33Z schst \$

- **Package** patError
- **Sub-Package** Examples
- **Link** <http://www.php-tools.net>
- **License** LGPL,
- **Copyright** PHP Application Tools
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>

include\_once ['./patErrorManager.php'](#)*[line 17]*

### patErrorManager class

## Class NoticeException

*[line 22]*

### Exception class for notices

- **Package** patError
- **Sub-Package** Examples

# package.php

## package.xml generation file for patForms

\$Id: package.php 48 2008-07-19 17:29:01Z schst \$

- **Package** patError
- **Sub-Package** Tools
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>

include **\_\_FILE\_\_** *[line 21]*

require\_once **'PEAR/PackageFileManager/Svn.php'** *[line 19]*

require\_once **'PEAR/PackageFileManager2.php'** *[line 18]*

**uses PackageFileManager Version 2**



# Package patTemplate Procedural Elements

## autopackage2.php

### package.xml generation file for patTemplate

This file is executed by createSnaps.php to automatically create a package that can be installed via the PEAR installer.

\$Id: autopackage2.php 462 2007-06-12 21:15:34Z gerd \$

- **Package** patTemplate
- **Sub-Package** Tools
- **Author** Stephan Schmidt < [schst@php-tools.net](mailto:schst@php-tools.net)>
- **Author** gERD Schaufelberger < [gerd@php-tools.net](mailto:gerd@php-tools.net)>

include **\_\_FILE\_\_** *[line 23]*

require\_once **'PEAR/PackageFileManager/Svn.php'** *[line 21]*

require\_once **'PEAR/PackageFileManager2.php'** *[line 20]*

**uses PackageFileManager**

# Appendices

# Appendix A - Class Trees

Package patSessoIn

Package patError

## NoticeException

- Exception
  - [NoticeException](#)

## patError

- [patError](#)

## patErrorManager

- [patErrorManager](#)

Package patErrorManager

## patErrorHandlerDebug

- [patErrorHandlerDebug](#)

Package patTemplate

# Appendix D - Todo List

## In Package patErrorManager

In [patErrorHandlerDebug::schstDebug\(\)](#)

- console output (no HTML)

## In Package patError

In [patErrorManager](#)

- implement expectError() to ignore an error with a certain code only once.
- implement ignoreError() to ignore errors with a certain code

In [patErrorManager::raise\(\)](#)

- either remove HTML tags and entities from output or test for environment!!! in shell is ugly!
- implement 'simple' mode that returns just false (BC for patConfiguration)

# Index

## A

<a href="#">autopackage2.php</a> . . . . .	38
<i>package.xml generation file for patTemplate</i>	

## C

<a href="#">constructor patError::__construct()</a> . . . . .	14
<i>constructor - used to set up the error with all needed error details.</i>	

## E

<a href="#">example expect.php</a> . . . . .	34
<i>Simple example for patError and patErrorManager</i>	
<a href="#">example ignore.php</a> . . . . .	35
<i>Simple example for patError and patErrorManager</i>	
<a href="#">example exception.php</a> . . . . .	33
<i>Simple example for patError and patErrorManager</i>	
<a href="#">example errormanager_customlevel.php</a> . . . . .	32
<i>patErrorManager example</i>	
<a href="#">example error.php</a> . . . . .	31
<i>Simple example for patError and patErrorManager</i>	
<a href="#">example cli_error.php</a> . . . . .	30
<i>Simple example for patError and patErrorManager</i>	

## F

<a href="#">foo()</a> . . . . .	30
<i>foo function raises an error</i>	

## N

<a href="#">NoticeException</a> . . . . .	35
<i>Exception class for notices</i>	

## P

<a href="#">patErrorManager::getIgnore()</a> . . . . .	20
<i>recieve all registerd error codes that will be ignored</i>	
<a href="#">patErrorManager::getExpect()</a> . . . . .	20
<i>recieve all registerd error codes that will be ignored</i>	

<a href="#">patErrorManager::handleErrorCallback()</a>	20
<i>handleError: callback</i>	
<i>forward error to custom handler</i>	
<a href="#">patErrorManager::handleErrorDie()</a>	21
<i>handleError: die</i>	
<i>display error-message and die</i>	
<a href="#">patErrorManager::handleErrorException()</a>	21
<i>handleError: throw an exception</i>	
<a href="#">patErrorManager::handleErrorEcho()</a>	21
<i>handleError: Echo</i>	
<i>display error message</i>	
<a href="#">patErrorManager::getErrorHandling()</a>	19
<i>retrieves the current error handling settings for the specified error level.</i>	
<a href="#">patErrorManager::clearIgnore()</a>	19
<i>empty list of errors to be ignored</i>	
<a href="#">patErrorManager::\$errorHandling</a>	18
<i>global definitions needed to keep track of things when calling the patErrorManager</i>	
<i>static methods.</i>	
<a href="#">patErrorManager::\$errorExpects</a>	17
<i>expects errors</i>	
<a href="#">patErrorManager::\$errorIgnores</a>	18
<i>ignore errors</i>	
<a href="#">patErrorManager::\$errorLevels</a>	18
<i>available error levels</i>	
<a href="#">patErrorManager::clearExpect()</a>	19
<i>empty list of errors to be ignored</i>	
<a href="#">patErrorManager::addIgnore()</a>	19
<i>add error codes to be ingored</i>	
<a href="#">patErrorManager::handleErrorIgnore()</a>	22
<i>handleError: Ignore</i>	
<i>Does nothing</i>	
<a href="#">patErrorManager::handleErrorTrigger()</a>	22
<i>handleError: trigger</i>	
<i>trigger php-error</i>	
<a href="#">patErrorManager::removeIgnore()</a>	27
<i>removeIgnore</i>	
<a href="#">patErrorManager::registerErrorLevel()</a>	26
<i>register a new error level</i>	
<a href="#">patErrorManager::setErrorClass()</a>	27
<i>setErrorClass</i>	
<a href="#">patErrorManager::setErrorHandling()</a>	27
<i>sets the way the patErrorManager will handle teh different error levels. Use this</i>	
<i>if you want to override the default settings.</i>	
<a href="#">package.php</a>	36
<i>package.xml generation file for patForms</i>	
<a href="#">patErrorManager::translateErrorLevel()</a>	28
<i>translate an error level</i>	
<a href="#">patErrorManager::raiseWarning()</a>	26
<i>wrapper for the <a href="#">raise()</a> method where you do not have to specify the</i>	
<i>error level - a <a href="#">patError</a> object with error level E_WARNING will be returned.</i>	
<a href="#">patErrorManager::raiseNotice()</a>	25
<i>wrapper for the <a href="#">raise()</a> method where you do not have to specify the</i>	
<i>error level - a <a href="#">patError</a> object with error level E_NOTICE will be returned.</i>	

<a href="#">patErrorManager::isError()</a>	23
<i>method for checking whether the return value of a pat application method is a pat error object.</i>	
<a href="#">patErrorManager::handleErrorVerbose()</a>	23
<i>handleError: Verbose display verbose output for developing purpose</i>	
<a href="#">patErrorManager::popExpect()</a>	23
<i>remove top of error-codes from stack</i>	
<a href="#">patErrorManager::pushExpect()</a>	24
<i>add expected errors to stack</i>	
<a href="#">patErrorManager::raiseError()</a>	25
<i>wrapper for the <a href="#">raise()</a> method where you do not have to specify the error level - a <a href="#">patError</a> object with error level E_ERROR will be returned.</i>	
<a href="#">patErrorManager::raise()</a>	24
<i>creates a new patError object given the specified information.</i>	
<a href="#">patErrorManager::\$errorClass</a>	17
<i>error class names</i>	
<a href="#">patErrorManager</a>	17
<i>patErrorManager main error management class used by pat tools for the application-internal error management. Creates patError objects for any errors for precise error management.</i>	
<a href="#">PATERRORMANAGER_ERROR_ILLEGAL_OPTIONS</a>	10
<i>error definition: illegal options.</i>	
<a href="#">PATERRORMANAGER_ERROR_ILLEGAL_MODE</a>	10
<i>error definition: illegal error handling mode.</i>	
<a href="#">patError</a>	11
<i>patError error object used by the patFormsError manager as error messages container for precise error management.</i>	
<a href="#">patError::\$args</a>	11
<i>stores the arguments the method that the error occurred in had received.</i>	
<a href="#">patError::\$class</a>	12
<i>stores the name of the class (if any) the error occurred in.</i>	
<a href="#">patError::\$backtrace</a>	12
<i>stores the complete debug backtrace (if your PHP version has the debug_backtrace function)</i>	
<a href="#">PATERRORMANAGER_ERROR_CALLBACK_NOT_CALLABLE</a>	10
<i>error definition: callback function does not exist.</i>	
<a href="#">patErrorManager.php</a>	10
<i>patErrorManager main error management class used by pat tools for the application-internal error management. Creates patError objects for any errors for precise error management.</i>	
<a href="#">patErrorHandlerDebug::arghDebug()</a>	3
<i>Error handler that outputs pretty debugging HTML</i>	
<a href="#">patErrorHandlerDebug</a>	3
<i>custom patErrorManager handler for the callback error handling mode</i>	
<a href="#">patErrorHandlerDebug::niceDie()</a>	4
<i>niceDie - outputs a nicely formatted version of the traditional die()</i>	
<a href="#">patErrorHandlerDebug::schstDebug()</a>	4
<i>error handler that outputs nice debugging HTML</i>	
<a href="#">patError.php</a>	9
<i>patError error object used by the patFormsError manager as error messages container for precise error management.</i>	
<a href="#">package-config.php</a>	7

<i>package-config for patTemplate</i>	
<a href="#">patError::\$code</a>	12
<i>stores the code of the error</i>	
<a href="#">patError::\$file</a>	12
<i>stores the filename of the file the error occurred in.</i>	
<a href="#">patError::getFile()</a>	15
<i>get the filename in which the error occurred</i>	
<a href="#">patError::getCode()</a>	15
<i>recieve error code</i>	
<a href="#">patError::getInfo()</a>	15
<i>retrieves the additional error information (information usually only relevant for developers)</i>	
<a href="#">patError::getLevel()</a>	16
<i>returns the error level of the error - corresponds to the PHP error levels (E_ALL, E_NOTICE...)</i>	
<a href="#">patError::getMessage()</a>	16
<i>retrieves the error message</i>	
<a href="#">patError::getLine()</a>	16
<i>get the line number in which the error occurred</i>	
<a href="#">patError::getBacktrace()</a>	15
<i>get the backtrace</i>	
<a href="#">patError::\$type</a>	14
<i>stores the type of error, as it is listed in the error backtrace</i>	
<a href="#">patError::\$info</a>	13
<i>additional info that is relevant for the developer of the script (e.g. if a database connect fails, the dsn used) and that the end-user should not see.</i>	
<a href="#">patError::\$function</a>	13
<i>stores the name of the method the error occurred in</i>	
<a href="#">patError::\$level</a>	13
<i>stores the error level for this error</i>	
<a href="#">patError::\$line</a>	13
<i>stores the line number the error occurred in.</i>	
<a href="#">patError::\$message</a>	14
<i>stores the error message - this is the message that can also be shown the user if need be.</i>	
<a href="#">patErrorHandlerDebug.php</a>	2
<i>custom patErrorHandler handler for the callback error handling mode</i>	