

# phphooks\_class doc



# Contents

<a href="#">Package phphooks Procedural Elements</a>	2
<a href="#">phphooks.class.php</a>	2
<a href="#">Define PLUGINS_FOLDER</a>	2
<a href="#">Package phphooks Classes</a>	3
<a href="#">Class phphooks</a>	3
<a href="#">Var \$hooks</a>	3
<a href="#">Var \$plugins</a>	3
<a href="#">Method add_hook</a>	3
<a href="#">Method execute_hook</a>	4
<a href="#">Method filter_hook</a>	4
<a href="#">Method hook_exist</a>	4
<a href="#">Method load_plugins</a>	5
<a href="#">Method register_plugin</a>	5
<a href="#">Method set_hook</a>	5
<a href="#">Method set_hooks</a>	6
<a href="#">Method unset_hook</a>	6
<a href="#">Method unset_hooks</a>	7
<a href="#">Package default Procedural Elements</a>	9
<a href="#">DebugDemo.php</a>	9
<a href="#">Function display_workers</a>	9
<a href="#">Function row_color</a>	9
<a href="#">Appendices</a>	10
<a href="#">Appendix A - Class Trees</a>	11
<a href="#">default</a>	11
<a href="#">phphooks</a>	11



# Package phphooks Procedural Elements

## phphooks.class.php

- **Package** phphooks
- **Author** eric.wzy@gmail.com
- **Version** 1.0

PLUGINS\_FOLDER = 'plugins' *[line 11]*

# Package phphooks Classes

## Class phphooks *[line 13]*

- **Package** phphooks

### phphooks::\$hooks

*mixed = array() [line 25]*

### hooks data

### phphooks::\$plugins

*mixed = array() [line 19]*

### plugins data

*void function phphooks::add\_hook(\$tag, \$function, [\$priority = 10]) [line 112]*

#### **Function Parameters:**

- *string* **\$tag** The name of the hook.
- *string* **\$function** The function you wish to be called.
- *int* **\$priority** optional. Used to specify the order in which the functions associated with a particular action are executed.(range 0~20, 0 first call, 20 last call)

### attach custom function to hook

- Since 1.0

*option.* function phphooks::execute\_hook(\$tag, [\$args = ""]) [line 141]

**Function Parameters:**

- *string* **\$tag** The name of the hook.
- *array* **\$args** optional. The arguments the function accept (default none)

**execute all functions which are attached to hook, you can provide argument (or arguments via array)**

- Since 1.0

*array.* function phphooks::filter\_hook(\$tag, [\$args = ""]) [line 168]

**Function Parameters:**

- *string* **\$tag** The name of the hook.
- *array* **\$args** optional. The arguments the function accept to filter (default none)

**filter \$args and after modify, return it. (or arguments via array)**

- Since 1.0

*void* function phphooks::hook\_exist(\$tag) [line 128]

**Function Parameters:**

- *string* **\$tag** The name of the hook.

## check whether any function is attached to hook

- Since 1.0

void function phphooks::load\_plugins([\$from\_folder = PLUGINS\_FOLDER]) [line 82]

### **Function Parameters:**

- **var \$from\_folder** option. load plugins from folder, if no argument is supplied, a 'plugins/' constant will be used

## load plugins from specific folder, includes all \*.plugin.php files

- Since 1.0

void function phphooks::register\_plugin(\$plugin\_id, [\$data = "], \$plugin\_id.) [line 195]

### **Function Parameters:**

- **string \$plugin\_id**. The name of the plugin.
- **array \$data** optional. The data the plugin accessorial (default none)
- **\$plugin\_id**

## register plugin data in \$this->plugin

- Since 1.0

void function phphooks::set\_hook(\$tag) [line 34]

### **Function Parameters:**

- *string* **\$tag** The name of the hook.

**register hook name/tag, so plugin developers can attach functions to hooks**

- **Since 1.0**

*void* function phphooks::set\_hooks(\$tags, \$tag) [*line 45*]

**Function Parameters:**

- *array* **\$tag** The name of the hooks.
- **\$tags**

**register multiple hooks name/tag**

- **Since 1.0**

*void* function phphooks::unset\_hook(\$tag) [*line 58*]

**Function Parameters:**

- *string* **\$tag** The name of the hook.

**write hook off**

- **Since 1.0**



*void* function phphooks::unset\_hooks(\$tags, \$tag) [*line 69*]

***Function Parameters:***

- *array* **\$tag** The name of the hooks.
- **\$tags**

**write multiple hooks off**

- **Since 1.0**



# Package default Procedural Elements

## DebugDemo.php

- **Package** default

*void* function display\_workers() [*line 41*]

**Displays a table of the workers**

*string* function row\_color(\$i) [*line 24*]

**Function Parameters:**

- *int* \$i

**Returns 'white' for even numbers and 'yellow' for odd numbers**

# Appendices

# Appendix A - Class Trees

Package default

Package phphooks

## phphooks

- [phphooks](#)

# Index

## D

<a href="#">display_workers()</a> . . . . .	9
<a href="#">DebugDemo.php</a> . . . . .	9

## P

<a href="#">phphooks::register_plugin()</a> . . . . .	5
<i>register plugin data in \$this-&gt;plugin</i>	
<a href="#">phphooks::load_plugins()</a> . . . . .	5
<i>load plugins from specific folder, includes all *.plugin.php files</i>	
<a href="#">phphooks::set_hook()</a> . . . . .	5
<i>register hook name/tag, so plugin developers can attach functions to hooks</i>	
<a href="#">phphooks::set_hooks()</a> . . . . .	6
<i>register multiple hooks name/tag</i>	
<a href="#">phphooks::unset_hooks()</a> . . . . .	7
<i>write multiple hooks off</i>	
<a href="#">phphooks::unset_hook()</a> . . . . .	6
<i>write hook off</i>	
<a href="#">phphooks::hook_exist()</a> . . . . .	4
<i>check whether any function is attached to hook</i>	
<a href="#">phphooks::filter_hook()</a> . . . . .	4
<i>filter \$args and after modify, return it. (or arguments via array)</i>	
<a href="#">phphooks</a> . . . . .	3
<a href="#">PLUGINS FOLDER</a> . . . . .	2
<a href="#">phphooks::\$hooks</a> . . . . .	3
<i>hooks data</i>	
<a href="#">phphooks::\$plugins</a> . . . . .	3
<i>plugins data</i>	
<a href="#">phphooks::execute_hook()</a> . . . . .	4
<i>execute all functions which are attached to hook, you can provide argument (or arguments via array)</i>	
<a href="#">phphooks::add_hook()</a> . . . . .	3
<i>attach custom function to hook</i>	
<a href="#">phphooks.class.php</a> . . . . .	2

## R

<a href="#">row_color()</a> . . . . .	9
---------------------------------------	---