runkit

Introduction

The runkit extension provides means to modify constants, user-defined functions, and user-defined classes. It also provides for custom superglobal variables and embeddable sub-interpreters via sandboxing.

This package is meant as a feature added replacement for the » classkit package. When compiled with the *--enable-runkit=classkit* option to ./configure, it will export classkit compatible function definitions and constants.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

RUNKIT_IMPORT_FUNCTIONS (integer)
runkit_import() flag indicating that normal functions should be imported from the specified file.

RUNKIT_IMPORT_CLASS_METHODS (integer)
runkit_import() flag indicating that class methods should be imported from the specified file.

RUNKIT_IMPORT_CLASS_CONSTS (integer)
runkit_import() flag indicating that class constants should be imported from the specified file. Note that this flag is only meaningful in PHP versions 5.1.0 and above.

RUNKIT_IMPORT_CLASS_PROPS (integer)
runkit_import() flag indicating that class standard properties should be imported from the specified file.

RUNKIT_IMPORT_CLASSES (integer)
runkit_import() flag representing a bitwise OR of the **RUNKIT_IMPORT_CLASS_*** constants.

RUNKIT_IMPORT_OVERRIDE (integer)
runkit_import() flag indicating that if any of the imported functions, methods, constants, or properties already exist, they should be replaced with the new definitions. If this flag is not set, then any imported definitions which already exist will be discarded.

RUNKIT_ACC_PUBLIC (integer)
PHP 5 specific flag to runkit_method_add()

RUNKIT_ACC_PROTECTED (integer)
PHP 5 specific flag to runkit_method_add()

RUNKIT_ACC_PRIVATE (integer)
PHP 5 specific flag to runkit_method_add()

CLASSKIT_ACC_PUBLIC (integer)
PHP 5 specific flag to classkit_method_add() Only defined when classkit compatibility is enabled.

CLASSKIT_ACC_PROTECTED (integer)
PHP 5 specific flag to classkit_method_add() Only defined when classkit compatibility is enabled.

CLASSKIT_ACC_PRIVATE (integer)
PHP 5 specific flag to classkit_method_add() Only defined when classkit compatibility is enabled.

CLASSKIT_AGGREGATE_OVERRIDE (integer)
PHP 5 specific flag to classkit_import() Only defined when classkit compatibility is enabled.

RUNKIT_VERSION (string)
Defined to the current version of the runkit package.

CLASSKIT_VERSION (string)
Defined to the current version of the runkit package. Only defined when classkit compatibility is enabled.

Installing/Configuring

Requirements

Modifying Constants, Functions, Classes, and Methods works with all releases of PHP 4 and PHP 5. No special requirements are necessary.

Custom Superglobals are only available in PHP 4.2.0 or later.

Sandboxing requires PHP 5.1.0 or later, or PHP 5.0.0 with a special TSRM patch applied. Regardless of which version of PHP is in use it must be compiled with the *--enable-maintainer-zts* option. See the *README* file in the runkit package for additional information.

Installation

This » PECL extension is not bundled with PHP.

Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/runkit.

The DLL for this PECL extension may be downloaded from either the » PHP Downloads page or from » http://pecl4win.php.net/

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Runkit Configuration Options

Name	Default	Changeable	Changelog
runkit.superglobal	""	PHP_INI_PERDIR	
runkit.internal_override	"0"	PHP_INI_SYSTEM	

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

runkit.superglobal string

Comma-separated list of variable names to be treated as superglobals. This value should be set in the systemwide php.ini file, but may work in perdir configuration contexts depending on your SAPI.

Example #1 - Custom Superglobals with runkit.superglobal=_FOO,_BAR in php.ini

```
<?php
function show_values() {
 echo "Foo is $_FOO\n";
 echo "Bar is $_BAR\n";
 echo "Baz is $_BAZ\n";
}

$_FOO = 'foo';
$_BAR = 'bar';
$_BAZ = 'baz';

/* Displays foo and bar, but not baz */
show_values();
?>
```

Resource Types

This extension has no resource types defined.

runkit Functions

Runkit_Sandbox

Runkit_Sandbox -- Runkit Sandbox Class -- PHP Virtual Machine

Description

Instantiating the Runkit_Sandbox class creates a new thread with its own scope and program stack. Using a set of options passed to the constructor, this environment may be restricted to a subset of what the primary interpreter can do and provide a safer environment for executing user supplied code.

Note
Sandbox support (required for runkit_lint(), runkit_lint_file(), and the Runkit_Sandbox class) is only available as of PHP 5.1.0 or specially patched versions of PHP 5.0, and requires that thread safety be enabled. See the *README* file included in the runkit package for more information.

Constructor

void **Runkit_Sandbox::__construct** ([array `$options`])

`options` is an associative array containing any combination of the special ini options listed below.

`safe_mode`
: If the outer script which is instantiating the Runkit_Sandbox class is configured with *safe_mode* = *off*, then safe_mode may be turned on for the sandbox environment. This setting can not be used to disable *safe_mode* when it's already enabled in the outer script.

`safe_mode_gid`
: If the outer script which is instantiating the Runkit_Sandbox class is configured with *safe_mode_gid* = *on*, then safe_mode_gid may be turned off for the sandbox environment. This setting can not be used to enable *safe_mode_gid* when it's already disabled in the outer script.

`safe_mode_include_dir`
: If the outer script which is instantiating the Runkit_Sandbox class is configured with a *safe_mode_include_dir*, then a new safe_mode_include_dir may be set for sandbox environments below the currently defined value. safe_mode_include_dir may also be cleared to indicate that the bypass feature is disabled. If safe_mode_include_dir was blank in the outer script, but safe_mode was not enabled, then any arbitrary safe_mode_include_dir may be set while turning safe_mode on.

`open_basedir`

open_basedir may be set to any path below the current setting of *open_basedir*. If *open_basedir* is not set within the global scope, then it is assumed to be the root directory and may be set to any location.

allow_url_fopen
Like *safe_mode*, this setting can only be made more restrictive, in this case by setting it to **FALSE** when it is previously set to **TRUE**

disable_functions
Comma separated list of functions to disable within the sandbox sub-interpreter. This list need not contain the names of the currently disabled functions, they will remain disabled whether listed here or not.

disable_classes
Comma separated list of classes to disable within the sandbox sub-interpreter. This list need not contain the names of the currently disabled classes, they will remain disabled whether listed here or not.

runkit.superglobal
Comma separated list of variables to be treated as superglobals within the sandbox sub-interpreter. These variables will be used in addition to any variables defined internally or through the global runkit.superglobal setting.

runkit.internal_override
Ini option *runkit.internal_override* may be disabled (but not re-enabled) within sandboxes.

Example #2 - Instantiating a restricted sandbox

```
<?php
$options = array(
 'safe_mode'=>true,
 'open_basedir'=>'/var/www/users/jdoe/',
 'allow_url_fopen'=>'false',
 'disable_functions'=>'exec,shell_exec,passthru,system',
 'disable_classes'=>'myAppClass');
$sandbox = new Runkit_Sandbox($options);
/* Non-protected ini settings may set normally */
$sandbox->ini_set('html_errors',true);
?>
```

Accessing Variables

All variables in the global scope of the sandbox environment are accessible as properties of the sandbox object. The first thing to note is that because of the way memory between these two threads is managed, object and resource variables can not currently be exchanged between interpreters. Additionally, all arrays are deep copied and any references will be lost. This also means that references between interpreters are not possible.

Example #3 - Working with variables in a sandbox

```
<?php
$sandbox = new Runkit_Sandbox();

$sandbox->foo = 'bar';
$sandbox->eval('echo "$foo\n"; $bar = $foo . "baz";');
echo "{$sandbox->bar}\n";
if (isset($sandbox->foo)) unset($sandbox->foo);
$sandbox->eval('var_dump(isset($foo));');
?>
```

The above example will output:

```
bar
barbaz
bool(false)
```

Calling PHP Functions

Any function defined within the sandbox may be called as a method on the sandbox object. This also includes a few pseudo-function language constructs: eval(), **include()**, **include_once()**, **require()**, **require_once()**, echo(), print(), die(), and exit().

Example #4 - Calling sandbox functions

```
<?php
$sandbox = new Runkit_Sandbox();

echo $sandbox->str_replace('a','f','abc');
?>
```

The above example will output:

```
fbc
```

When passing arguments to a sandbox function, the arguments are taken from the outer instance of PHP. If you wish to pass arguments from the sandbox's scope, be sure to access them as properties of the sandbox object as illustrated above.

Example #5 - Passing arguments to sandbox functions

```
<?php
$sandbox = new Runkit_Sandbox();

$foo = 'bar';
$sandbox->foo = 'baz';
echo $sandbox->str_replace('a',$foo,'a');
echo $sandbox->str_replace('a',$sandbox->foo,'a');
?>
```

The above example will output:

```
bar
baz
```

Changing Sandbox Settings

As of runkit version 0.5, certain Sandbox settings may be modified on the fly using ArrayAccess syntax. Some settings, such as *`active`* are read-only and meant to provide status information. Other settings, such as *`output_handler`* may be set and read much like a normal array offset. Future settings may be write-only, however no such settings currently exist.

Sandbox Settings / Status Indicators

Setting	Type	Purpose	Default
active	Boolean (Read Only)	**TRUE** if the Sandbox is still in a usable state, **FALSE** if the request is in bailout due to a call to die(), exit(), or because of a fatal error condition.	**TRUE** (Initial)
output_handler	Callback	When set to a valid callback, all output generated by the Sandbox instance will be processed through the named function. Sandbox output handlers follow the same calling conventions as the system-wide output handler.	None
parent_access	Boolean	May the sandbox use instances of the Runkit_Sandbox_Parent class? Must be enabled for other Runkit_Sandbox_Parent related settings to work.	**FALSE**
parent_read	Boolean	May the sandbox read variables in its parent's context?	**FALSE**

parent_write	Boolean	May the sandbox modify variables in its parent's context?	**FALSE**
parent_eval	Boolean	May the sandbox evaluate arbitrary code in its parent's context? *DANGEROUS*	**FALSE**
parent_include	Boolean	May the sandbox include php code files in its parent's context? *DANGEROUS*	**FALSE**
parent_echo	Boolean	May the sandbox echo data in its parent's context effectively bypassing its own output_handler?	**FALSE**
parent_call	Boolean	May the sandbox call functions in its parent's context?	**FALSE**
parent_die	Boolean	May the sandbox kill its own parent? (And thus itself)	**FALSE**
parent_scope	Integer	What scope will parental property access look at? 0 == Global scope, 1 == Calling scope, 2 == Scope preceeding calling scope, 3 == The scope before that, etc..., etc...	*0* (Global)
parent_scope	String	When *parent_scope* is set to a string value, it refers to a named array variable in the global scope. If the named variable does not exist at the time of access it will be created as an empty array. If the variable exists but it	

		not an array, a dummy array will be created containing a reference to the named global variable.	

Runkit_Sandbox_Parent

Runkit_Sandbox_Parent -- Runkit Anti-Sandbox Class

Description

void **Runkit_Sandbox_Parent::__construct** (void)

Instantiating the Runkit_Sandbox_Parent class from within a sandbox environment created from the Runkit_Sandbox class provides some (controlled) means for a sandbox child to access its parent.

Note
Sandbox support (required for runkit_lint(), runkit_lint_file(), and the Runkit_Sandbox class) is only available as of PHP 5.1.0 or specially patched versions of PHP 5.0, and requires that thread safety be enabled. See the *README* file included in the runkit package for more information.

In order for any of the Runkit_Sandbox_Parent features to function. Support must be enabled on a per-sandbox basis by enabling the *parent_access* flag from the parent's context.

Example #6 - Working with variables in a sandbox

```
<?php
$sandbox = new Runkit_Sandbox();
$sandbox['parent_access'] = true;
?>
```

Accessing the Parent's Variables

Just as with sandbox variable access, a sandbox parent's variables may be read from and written to as properties of the Runkit_Sandbox_Parent class. Read access to parental variables may be enabled with the *parent_read* setting (in addition to the base *parent_access* setting). Write access, in turn, is enabled through the *parent_write* setting.

Unlike sandbox child variable access, the variable scope is not limited to globals only. By setting the *parent_scope* setting to an appropriate integer value, other scopes in the active call stack may be inspected instead. A value of 0 (Default) will direct variable access at the global scope. 1 will point variable access at whatever variable scope was active at the time the current block of sandbox code was executed. Higher values progress back through the functions that called the functions that led to the sandbox executing code that tried to access its own parent's variables.

Example #7 - Accessing parental variables

```
<?php
$php = new Runkit_Sandbox();
$php['parent_access'] = true;
$php['parent_read'] = true;

$test = "Global";

$php->eval('$PARENT = new Runkit_Sandbox_Parent;');

$php['parent_scope'] = 0;
one();

$php['parent_scope'] = 1;
one();

$php['parent_scope'] = 2;
one();

$php['parent_scope'] = 3;
one();

$php['parent_scope'] = 4;
one();

$php['parent_scope'] = 5;
one();

function one() {
   $test = "one()";
   two();
}

function two() {
   $test = "two()";
   three();
}

function three() {
   $test = "three()";
   $GLOBALS['php']->eval('var_dump($PARENT->test);');
}
?>
```

The above example will output:

```
string(6) "Global"
string(7) "three()"
string(5) "two()"
string(5) "one()"
string(6) "Global"
string(6) "Global"
```

Calling the Parent's Functions

Just as with sandbox access, a sandbox may access its parents functions providing that

the proper settings have been enabled. Enabling *parent_call* will allow the sandbox to call all functions available to the parent scope. Language constructs are each controlled by their own setting: print() and echo() are enabled with *parent_echo*. die() and exit() are enabled with *parent_die*. eval() is enabled with *parent_eval* while **include()**, **include_once()**, **require()**, and **require_once()** are enabled through *parent_include*.

runkit_class_adopt

runkit_class_adopt -- Convert a base class to an inherited class, add ancestral methods when appropriate

Description

bool **runkit_class_adopt** (string `$classname`, string `$parentname`)

Parameters

`classname`
Name of class to be adopted

`parentname`
Parent class which child class is extending

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #8 - A runkit_class_adopt() example

```
<?php
class myParent {
 function parentFunc() {
   echo "Parent Function Output\n";
 }
}

class myChild {
}

runkit_class_adopt('myChild','myParent');
myChild::parentFunc();
?>
```

The above example will output:

```
Parent Function Output
```

See Also

- runkit_class_emancipate()

runkit_class_emancipate

runkit_class_emancipate -- Convert an inherited class to a base class, removes any method whose scope is ancestral

Description

bool **runkit_class_emancipate** (string `$classname`)

Parameters

`classname`
Name of class to emancipate

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #9 - A runkit_class_emancipate() example

```
<?php
class myParent {
 function parentFunc () {
   echo "Parent Function Output\n";
 }
}
class myChild extends myParent {
}

myChild::parentFunc();
runkit_class_emancipate('myChild');
myChild::parentFunc();
?>
```

The above example will output:

```
Parent Function Output
Fatal error: Call to undefined function:  parentFunc() in example.php on
line 12
```

See Also

- runkit_class_adopt()

runkit_constant_add

runkit_constant_add -- Similar to define(), but allows defining in class definitions as well

Description

bool **runkit_constant_add** (string `$constname`, mixed `$value`)

Parameters

`constname`
: Name of constant to declare. Either a string to indicate a global constant, or *classname::constname* to indicate a class constant.

`value`
: NULL, Bool, Long, Double, String, or Resource value to store in the new constant.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- define()
- runkit_constant_redefine()
- runkit_constant_remove()

runkit_constant_redefine

runkit_constant_redefine -- Redefine an already defined constant

Description

bool **runkit_constant_redefine** (string `$constname`, mixed `$newvalue`)

Parameters

`constname`
: Constant to redefine. Either string indicating global constant, or *classname::constname* indicating class constant.

`newvalue`
: New value to assign to constant.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- runkit_constant_add()
- runkit_constant_remove()

runkit_constant_remove

runkit_constant_remove -- Remove/Delete an already defined constant

Description

bool **runkit_constant_remove** (string `$constname`)

Parameters

`constname`
: Name of constant to remove. Either a string indicating a global constant, or *classname::constname* indicating a class constant.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- define()
- runkit_constant_add()
- runkit_constant_redefine()

runkit_function_add

runkit_function_add -- Add a new function, similar to create_function()

Description

bool **runkit_function_add** (string `$funcname`, string `$arglist`, string `$code`)

Parameters

`funcname`
Name of function to be created

`arglist`
Comma separated argument list

`code`
Code making up the function

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #10 - A runkit_function_add() example

```
<?php
runkit_function_add('testme','$a,$b','echo "The value of a is $a\n"; echo
"The value of b is $b\n";');
testme(1,2);
?>
```

The above example will output:

```
The value of a is 1
The value of b is 2
```

See Also

- create_function()

- runkit_function_redefine()
- runkit_function_copy()
- runkit_function_rename()
- runkit_function_remove()
- runkit_method_add()

runkit_function_copy

runkit_function_copy -- Copy a function to a new function name

Description

bool **runkit_function_copy** (string `$funcname`, string `$targetname`)

Parameters

`funcname`
Name of existing function

`targetname`
Name of new function to copy definition to

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #11 - A runkit_function_copy() example

```
<?php
function original() {
 echo "In a function\n";
}
runkit_function_copy('original','duplicate');
original();
duplicate();
?>
```

The above example will output:

```
In a function
In a function
```

See Also

- runkit_function_add()
- runkit_function_redefine()

- runkit_function_rename()
- runkit_function_remove()

runkit_function_redefine

runkit_function_redefine -- Replace a function definition with a new implementation

Description

bool **runkit_function_redefine** (string `$funcname`, string `$arglist`, string `$code`)

Note
By default, only userspace functions may be removed, renamed, or modified. In order to override internal functions, you must enable the *runkit.internal_override* setting in *php.ini*.

Parameters

`funcname`
: Name of function to redefine

`arglist`
: New list of arguments to be accepted by function

`code`
: New code implementation

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #12 - A runkit_function_redefine() example

```
<?php
function testme() {
 echo "Original Testme Implementation\n";
}
testme();
runkit_function_redefine('testme','','echo "New Testme Implementation\n";');
testme();
?>
```

The above example will output:

```
Original Testme Implementation
New Testme Implementation
```

See Also

- runkit_function_add()
- runkit_function_copy()
- runkit_function_rename()
- runkit_function_remove()

runkit_function_remove

runkit_function_remove -- Remove a function definition

Description

bool **runkit_function_remove** (string `$funcname`)

Note
By default, only userspace functions may be removed, renamed, or modified. In order to override internal functions, you must enable the *runkit.internal_override* setting in *php.ini*.

Parameters

`funcname`
: Name of function to be deleted

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- runkit_function_add()
- runkit_function_copy()
- runkit_function_redefine()
- runkit_function_rename()

runkit_function_rename

runkit_function_rename -- Change a function's name

Description

bool **runkit_function_rename** (string `$funcname`, string `$newname`)

Note
By default, only userspace functions may be removed, renamed, or modified. In order to override internal functions, you must enable the *runkit.internal_override* setting in *php.ini*.

Parameters

`funcname`
: Current function name

`newname`
: New function name

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- runkit_function_add()
- runkit_function_copy()
- runkit_function_redefine()
- runkit_function_remove()

runkit_import

runkit_import -- Process a PHP file importing function and class definitions, overwriting where appropriate

Description

bool **runkit_import** (string `$filename` [, int `$flags`])

Similar to **include()** however any code residing outside of a function or class is simply ignored. Additionally, depending on the value of *`flags`*, any functions or classes which already exist in the currently running environment will be automatically overwritten by their new definitions.

Parameters

`filename`
: Filename to import function and class definitions from

`flags`
: Bitwise OR of the **RUNKIT_IMPORT_*** family of constants.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

runkit_lint_file

runkit_lint_file -- Check the PHP syntax of the specified file

Description

bool **runkit_lint_file** (string `$filename`)

The runkit_lint_file() function performs a syntax (lint) check on the specified filename testing for scripting errors. This is similar to using php -l from the commandline.

Note
Sandbox support (required for runkit_lint(), runkit_lint_file(), and the Runkit_Sandbox class) is only available as of PHP 5.1.0 or specially patched versions of PHP 5.0, and requires that thread safety be enabled. See the *README* file included in the runkit package for more information.

Parameters

`filename`
: File containing PHP Code to be lint checked

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- runkit_lint()

runkit_lint

runkit_lint -- Check the PHP syntax of the specified php code

Description

bool **runkit_lint** (string `$code`)

The runkit_lint() function performs a syntax (lint) check on the specified php code testing for scripting errors. This is similar to using *php -l* from the command line except runkit_lint() accepts actual code rather than a filename.

Note
Sandbox support (required for runkit_lint(), runkit_lint_file(), and the Runkit_Sandbox class) is only available as of PHP 5.1.0 or specially patched versions of PHP 5.0, and requires that thread safety be enabled. See the *README* file included in the runkit package for more information.

Parameters

code
: PHP Code to be lint checked

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- runkit_lint_file()

runkit_method_add

runkit_method_add -- Dynamically adds a new method to a given class

Description

bool **runkit_method_add** (string `$classname`, string `$methodname`, string `$args`, string `$code` [, int `$flags`])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`classname`
: The class to which this method will be added

`methodname`
: The name of the method to add

`args`
: Comma-delimited list of arguments for the newly-created method

`code`
: The code to be evaluated when *`methodname`* is called

`flags`
: The type of method to create, can be **RUNKIT_ACC_PUBLIC**, **RUNKIT_ACC_PROTECTED** or **RUNKIT_ACC_PRIVATE**

Note
This parameter is only used as of PHP 5, because, prior to this, all methods were public.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #13 - runkit_method_add() example

```
<?php
class Example {
   function foo() {
       echo "foo!\n";
   }
}

// create an Example object
$e = new Example();

// Add a new public method
runkit_method_add(
   'Example',
   'add',
   '$num1, $num2',
   'return $num1 + $num2;',
   RUNKIT_ACC_PUBLIC
);

// add 12 + 4
echo $e->add(12, 4);
?>
```

The above example will output:

```
16
```

See Also

- runkit_method_copy()
- runkit_method_redefine()
- runkit_method_remove()
- runkit_method_rename()
- runkit_function_add()

runkit_method_copy

runkit_method_copy -- Copies a method from class to another

Description

bool **runkit_method_copy** (string `$dClass`, string `$dMethod`, string `$sClass` [, string `$sMethod`])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`dClass`
: Destination class for copied method

`dMethod`
: Destination method name

`sClass`
: Source class of the method to copy

`sMethod`
: Name of the method to copy from the source class. If this parameter is omitted, the value of `dMethod` is assumed.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #14 - runkit_method_copy() example

```
<?php
class Foo {
   function example() {
       return "foo!\n";
   }
```

```
}

class Bar {
   // initially, no methods
}

// copy the example() method from the Foo class to the Bar class, as baz()
runkit_method_copy('Bar', 'baz', 'Foo', 'example');

// output copied function
echo Bar::baz();
?>
```

The above example will output:

```
foo!
```

See Also

- runkit_method_add()
- runkit_method_redefine()
- runkit_method_remove()
- runkit_method_rename()
- runkit_function_copy()

runkit_method_redefine

runkit_method_redefine -- Dynamically changes the code of the given method

Description

bool **runkit_method_redefine** (string `$classname`, string `$methodname`, string `$args`, string `$code` [, int `$flags`])

Note
This function cannot be used to manipulate the currently running (or chained) method.

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`classname`
: The class in which to redefine the method

`methodname`
: The name of the method to redefine

`args`
: Comma-delimited list of arguments for the redefined method

`code`
: The new code to be evaluated when *`methodname`* is called

`flags`
: The redefined method can be **RUNKIT_ACC_PUBLIC**, **RUNKIT_ACC_PROTECTED** or **RUNKIT_ACC_PRIVATE**

Note
This parameter is only used as of PHP 5, because, prior to this, all methods were public.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #15 - runkit_method_redefine() example

```
<?php
class Example {
   function foo() {
       return "foo!\n";
   }
}

// create an Example object
$e = new Example();

// output Example::foo() (before redefine)
echo "Before: " . $e->foo();

// Redefine the 'foo' method
runkit_method_redefine(
   'Example',
   'foo',
   '',
   'return "bar!\n";',
   RUNKIT_ACC_PUBLIC
);

// output Example::foo() (after redefine)
echo "After: " . $e->foo();
?>
```

The above example will output:

```
Before: foo!
After: bar!
```

See Also

- runkit_method_add()
- runkit_method_copy()
- runkit_method_remove()
- runkit_method_rename()
- runkit_function_redefine()

runkit_method_remove

runkit_method_remove -- Dynamically removes the given method

Description

bool **runkit_method_remove** (string `$classname`, string `$methodname`)

Note
This function cannot be used to manipulate the currently running (or chained) method.

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`classname`
The class in which to remove the method

`methodname`
The name of the method to remove

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #16 - runkit_method_remove() example

```
<?php
class Example {
   function foo() {
       return "foo!\n";
   }

   function bar() {
       return "bar!\n";
```

```
    }
}

// Remove the 'foo' method
runkit_method_remove(
    'Example',
    'foo'
);

echo implode(' ', get_class_methods('Example'));

?>
```

The above example will output:

```
bar
```

See Also

- runkit_method_add()
- runkit_method_copy()
- runkit_method_redefine()
- runkit_method_rename()
- runkit_function_remove()

runkit_method_rename

runkit_method_rename -- Dynamically changes the name of the given method

Description

bool **runkit_method_rename** (string `$classname`, string `$methodname`, string `$newname`)

Note
This function cannot be used to manipulate the currently running (or chained) method.

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`classname`
The class in which to rename the method

`methodname`
The name of the method to rename

`newname`
The new name to give to the renamed method

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #17 - runkit_method_rename() **example**

```
<?php
class Example {
   function foo() {
       return "foo!\n";
```

```
    }
}

// Rename the 'foo' method to 'bar'
runkit_method_rename(
    'Example',
    'foo',
    'bar'
);

// output renamed function
echo Example::bar();
?>
```

The above example will output:

```
foo!
```

See Also

- runkit_method_add()
- runkit_method_copy()
- runkit_method_redefine()
- runkit_method_remove()
- runkit_function_rename()

runkit_return_value_used

runkit_return_value_used -- Determines if the current functions return value will be used

Description

bool **runkit_return_value_used** (void)

Return Values

Returns **TRUE** if the function's return value is used by the calling scope, otherwise **FALSE**

Examples

Example #18 - runkit_return_value_used() **example**

```
<?php
function foo() {
 var_dump(runkit_return_value_used());
}

foo();
$f = foo();
?>
```

The above example will output:

```
bool(false)
bool(true)
```

runkit_sandbox_output_handler

runkit_sandbox_output_handler -- Specify a function to capture and/or process output from a runkit sandbox

Description

mixed **runkit_sandbox_output_handler** (object `$sandbox` [, mixed `$callback`])

Ordinarily, anything output (such as with echo() or print()) will be output as though it were printed from the parent's scope. Using runkit_sandbox_output_handler() however, output generated by the sandbox (including errors), can be captured by a function outside of the sandbox.

Note
Sandbox support (required for runkit_lint(), runkit_lint_file(), and the Runkit_Sandbox class) is only available as of PHP 5.1.0 or specially patched versions of PHP 5.0, and requires that thread safety be enabled. See the *README* file included in the runkit package for more information.

Note
Deprecated As of runkit version 0.5, this function is deprecated and is scheduled to be removed from the package prior to a 1.0 release. The output handler for a given Runkit_Sandbox instance may be read/set using the array offset syntax shown on the Runkit_Sandbox class definition page.

Parameters

`sandbox`
: Object instance of Runkit_Sandbox class on which to set output handling.

`callback`
: Name of a function which expects one parameter. Output generated by *`sandbox`* will be passed to this callback. Anything returned by the callback will be displayed normally. If this parameter is not passed then output handling will not be changed. If a non-truth value is passed, output handling will be disabled and will revert to direct display.

Return Values

Returns the name of the previously defined output handler callback, or **FALSE** if no handler was previously defined.

Examples

Example #19 - Feeding output to a variable

```
<?php
function capture_output($str) {
 $GLOBALS['sandbox_output'] .= $str;

 return '';
}

$sandbox_output = '';

$php = new Runkit_Sandbox();
runkit_sandbox_output_handler($php, 'capture_output');
$php->echo("Hello\n");
$php->eval('var_dump("Excuse me");');
$php->die("I lost myself.");
unset($php);

echo "Sandbox Complete\n\n";
echo $sandbox_output;
?>
```

The above example will output:

```
Sandbox Complete

Hello
string(9) "Excuse me"
I lost myself.
```

runkit_superglobals

runkit_superglobals -- Return numerically indexed array of registered superglobals

Description

array **runkit_superglobals** (void)

Return Values

Returns a numerically indexed array of the currently registered superglobals. i.e. _GET, _POST, _REQUEST, _COOKIE, _SESSION, _SERVER, _ENV, _FILES

See Also

- Variable Scope