

patSpiderizer



Contents

Package patSpiderizer Procedural Elements	2
patSpiderizer.php	2
Function checkInstallation	2
Function displayErrors	2
example.php	3
Function addVars	3
patSpiderizerAdmin.php	4
Define UPDATE_DBURL	4
Define UPDATE_DBVERSIONURL	4
xmlrpc.inc	5
Function iso8601_decode	5
Function iso8601_encode	5
Function xmlrpcDecode	5
Function xmlrpcEncode	5
Function xmlrpc_cd	6
Function xmlrpc_dh	6
Function xmlrpc_ee	6
Function xmlrpc_entity_decode	6
Function xmlrpc_lookup_entity	6
Function xmlrpc_se	6
Package patSpiderizer Classes	8
Class patExtras	8
Var \$errorCodes	8
Var \$errors	8
Var \$systemVars	9
Var \$updateIntervals	9
Method addError	9
Method displayMessage	9
Method getErrorCodes	10
Method getErrorMessages	10
Method getErrors	10
Method getLastError	10
Method getLastErrorCode	11
Method getLastErrorMessage	11
Method process	11
Method readConfig	11
Method reportErrors	12
Method setAdminInterface	12
Method setTemplate	12
Method setXMLRpcClient	12
Method writeConfig	13
Class patSpiderizer	13
Var \$amountErrors	13

Var \$config	14
Var \$vars	14
Constructor patSpiderizer	14
Method runCheck	14
Method setAdminMail	14
Method setDebugMode	15
Method setExitPage	15
Method setIp	16
Method setOption	16
Method setRootPath	16
Method setUseragent	17
Class patSpiderizer Admin	17
Var \$adminfile	17
Var \$config	18
Var \$extras	18
Var \$options	18
Var \$systemVars	18
Var \$template	18
Var \$vars	18
Method connectBlock	18
Method connectBlocks	18
Method connectLink	19
Method connectLinks	19
Method deleteLink	19
Method deleteTextBlock	19
Method displayEnginesDeletelp	19
Method displayEnginesDeleteUseragent	19
Method displayEnginesEditIp	19
Method displayEnginesEditUseragent	19
Method displayTemplatesView	19
Method getAppInfo	19
Method getOptions	20
Method getRandomConfirmTexts	20
Method setAdminfile	20
Method setConfigReference	20
Method setExtrasReference	20
Method setTemplateReference	21
Method start	21
Class patXMLRpcClient	21
Var \$client	22
Var \$errors	22
Constructor patXMLRpcClient	22
Method checkForUpdate	22
Method getProjectList	22
Method getServerVersion	23
Method getSupportedApps	23
Class xmlrpcmsg	23
Var \$debug	23
Var \$methodname	24

Var \$params	24
Var \$payload	24
Constructor xmlrpcmsg	24
Method addParam	24
Method createPayload	24
Method getNumParams	24
Method getParam	24
Method method	24
Method parseResponse	24
Method parseResponseFile	24
Method serialize	25
Method xml_footer	25
Method xml_header	25
Class xmlrpcresp	25
Var \$fn	25
Var \$fs	25
Var \$hdrs	25
Var \$xv	25
Constructor xmlrpcresp	25
Method faultCode	25
Method faultString	25
Method serialize	25
Method value	25
Class xmlrpcval	26
Var \$me	26
Var \$mytype	26
Constructor xmlrpcval	26
Method addArray	26
Method addScalar	26
Method addStruct	26
Method arraymem	27
Method arraysize	27
Method dump	27
Method getval	27
Method kindOf	27
Method scalartyp	27
Method scalarval	27
Method serialize	27
Method serializedata	27
Method serializeval	27
Method structeach	27
Method structmem	27
Method structreset	27
Class xmlrpc_client	28
Var \$cert	28
Var \$certpass	28
Var \$debug	28
Var \$errno	28
Var \$errstring	28

Var \$password	28
Var \$path	28
Var \$port	28
Var \$server	28
Var \$username	28
Constructor xmlrpc_client	28
Method send	29
Method sendPayloadHTTP10	29
Method sendPayloadHTTPS	29
Method setCertificate	29
Method setCredentials	29
Method setDebug	30
Package patConfiguration Classes	31
Class patConfiguration	31
Var \$conf	31
Var \$data	31
Var \$extensions	31
Var \$nsStack	32
Var \$path	32
Var \$valDepth	32
Var \$valStack	32
Var \$xmlSpecialchars	32
Method addExtension	32
Method appendData	33
Method characterData	33
Method clearConfigValue	33
Method createParser	33
Method endElement	34
Method externalEntity	34
Method getConfigValue	34
Method parseConfigFile	34
Method parseXMLFile	35
Method replaceXMLSpecialchars	35
Method setConfigDir	35
Method setConfigValue	35
Method setConfigValues	36
Method setIncludeDir	36
Method startElement	37
Method writeConfigFile	37
Package patTemplate Procedural Elements	39
patTemplate.php	39
Define patTEMPLATE_TAG_END	39
Define patTEMPLATE_TAG_START	39
Define patTEMPLATE_TYPE_CONDITION	39
Define patTEMPLATE_TYPE_ODDEVEN	39
Define patTEMPLATE_TYPE_SIMPLECONDITION	39
Define patTEMPLATE_TYPE_STANDARD	40
Package patTemplate Classes	41

Class patTemplate	41
Constructor patTemplate	41
Method addGlobalVar	42
Method addGlobalVars	42
Method addRows	43
Method addTemplate	43
Method addTemplates	43
Method addVar	44
Method addVars	44
Method clearAllTemplates	45
Method clearAttribute	45
Method clearTemplate	45
Method displayParsedTemplate	46
Method dump	46
Method exists	46
Method getAttribute	47
Method getParsedTemplate	47
Method getVar	48
Method parseTemplate	48
Method readTemplatesFromFile	48
Method setAttribute	49
Method setAttributes	49
Method setBasedir	50
Method setTags	50
Method setType	51
Appendices	52
Appendix A - Class Trees	53
patSpiderizer	53
patConfiguration	54
patTemplate	54

Package patSpiderizer Procedural Elements

patSpiderizer.php

- **Package** patSpiderizer

function checkInstallation() [*line 68*]
function displayErrors(\$errors) [*line 147*]

Function Parameters:

- **\$errors**

require_once ["../include/xmlrpc.inc"](#) [*line 36*]

require_once ["../include/patXMLRpcClient.php"](#) [*line 35*]

require_once ["../include/patExtras.php"](#) [*line 34*]

require_once ["../include/patConfiguration.php"](#) [*line 32*]

require_once ["../include/patTemplate.php"](#) [*line 33*]

require_once ["../include/patSpiderizerAdmin.php"](#) [*line 31*]

example.php

- **Package** patSpiderizer

function addVars(\$exitpage) *[line 92]*

Function Parameters:

- **\$exitpage**

include **\$rootpath."/include/patSpiderizer.php"** *[line 33]*

include **\$rootpath."/include/patConfiguration.php"** *[line 29]*

patSpiderizerAdmin.php

- **Package** patSpiderizer

UPDATE_DBURL = "http://www.php-tools.de/updates/patSpiderizer/db-v2.upd" *[line 25]*
sets the url of the search engine database update server's database file

UPDATE_DBVERSIONURL = "http://www.php-tools.de/updates/patSpiderizer/dbversion-v2.txt" *[line 19]*
sets the url of the search engine database update server's version file

xmlrpc.inc

- **Package** patSpiderizer

function iso8601_decode(\$idate, [\$utc = 0]) [*line 981*]

Function Parameters:

- **\$idate**
- **\$utc**

function iso8601_encode(\$timet, [\$utc = 0]) [*line 956*]

Function Parameters:

- **\$timet**
- **\$utc**

function xmlrpcDecode(\$xmlrpc_val) [*line 1001*]

Function Parameters:

- **\$xmlrpc_val**

xmlrpc_decode takes a message in PHP xmlrpc object format and * tranlates it into
native PHP types. *
Libby (dan@libby.com) * author: Dan *

function xmlrpcEncode(\$php_val) [*line 1039*]

Function Parameters:

- **\$php_val**

xmlrpc_encode takes native php types and encodes them into * xmlrpc PHP object

format. * BUG: All sequential arrays are turned into structs. I don't *
know of a good way to determine if an array is sequential * only.
* feature creep -- could support more types
via optional type * argument.
* author: Dan Libby (dan@libby.com) *

function xmlrpc_cd(\$parser, \$data) [line 314]

Function Parameters:

- **\$parser**
- **\$data**

function xmlrpc_dh(\$parser, \$data) [line 339]

Function Parameters:

- **\$parser**
- **\$data**

function xmlrpc_ee(\$parser, \$name) [line 214]

Function Parameters:

- **\$parser**
- **\$name**

function xmlrpc_entity_decode(\$string) [line 119]

Function Parameters:

- **\$string**

function xmlrpc_lookup_entity(\$ent) [line 139]

Function Parameters:

- **\$ent**

function xmlrpc_se(\$parser, \$name, \$attrs) [line 149]

Function Parameters:

- **\$parser**
- **\$name**
- **\$attrs**

Package patSpiderizer Classes

Class patExtras *[line 11]*

pat Administration Extras

\$Id: patExtras.php 2 2004-03-02 23:07:44Z argh \$

- **Package** patSpiderizer
- **Author** Sebastian Mordziol < argh@php-tools.de>, Stephan Schmidt <schst@php-tools.de>
- **Version** 0.11
- **Access** public

patExtras::\$errorCodes

```
mixed = array(
    1   => "Could not load configuration file.",
    2   => "Could not write configuration file.",

    10  => "Please enter your name.",
    11  => "Please enter your email adress.",
    15  => "Please enter a detailed description of the bug.",

    100 => "PHP's xml extension is not installed. No XML-RPC available.",
    101 => "PHP's libcurl extension is not installed. No XML-RPC available.",
    102 => "Got no or invalid response from the XML-RPC server."
) [line 13]
```

patExtras::\$errors

```
array = array() [line 67]
```

array to store all errors

patExtras::\$systemVars

```
array = array(
    "appname" => "patExtras",
    "version" => 0.11,
    "date" => "2002-02-04",
    "author" => array( "Stephan Schmidt", "Sebastian Mordziol" ) ) [line 55]
```

information about patExtras

patExtras::\$updateIntervals

```
array = array(
    array(
        "value" => 0,
        "title" => "Always"
    ), array("value"=>86400,"title"=>"Once per day"), array("value"=>604800,"title"=>"Once per
week"), array("value"=>-1,"title"=>"Never")) [line 31]
```

intervals for the "check for updates selector"

```
function patExtras::addError($code, [$message = ""], $msg) [line 435]
```

Function Parameters:

- *int* **\$code** error code
- *string* **\$msg** error message
- **\$message**

add an error

- **Access public**

```
function patExtras::displayMessage($message) [line 160]
```

Function Parameters:

- **\$message**

Displays a message (success, general notes, whatever...)

- **Access** public

array function patExtras::getErrorCodes() [*line 499*]
return code of all errors

- **Access** public

array function patExtras::getErrorMessages() [*line 514*]
return message of all errors

- **Access** public

array function patExtras::getErrors() [*line 488*]
return code and message of all errors

- **Access** public

array function patExtras::getLastError() [*line 455*]
return code and message of the last error

- **Access** public

int function patExtras::getLastErrorCode() [*line 466*]
return code of the last error

- **Access public**

string function patExtras::getLastErrorMessage() [*line 477*]
return message of the last error

- **Access public**

function patExtras::process(\$vars) [*line 108*]
Function Parameters:

- **\$vars**

run the extras administration interface

- **Access public**

function patExtras::readConfig() [*line 383*]
read the extras configuration file (csv formatted)

- **Access public**

boolean function patExtras::reportErrors() [*line 529*]

report (display) errors

- **Access** public

function patExtras::setAdminInterface(&\$interface) [*line 87*]

Function Parameters:

- *object patXMLRenderer* **&\$interface** administration interface

set reference to Randy

- **Access** public

function patExtras::setTemplate(&\$template) [*line 76*]

Function Parameters:

- *object patTemplate* **&\$template** template object used for rendering

set the template object, that should be used for rendering

- **Access** public

function patExtras::setXMLRpcClient(&\$client) [*line 98*]

Function Parameters:

- [*object patXMLRpcClient*](#) **&\$client** patXMLRpc client

set the XMLRpc Client object needed to retrieve data from the php-tools website

- **Access** public

function patExtras::writeConfig() [*line 409*]

write the extras configuration file (csv formatted)

- **Access** public

Class patSpiderizer

[*line 25*]

patSpiderizer search engine cloaking script

- **Package** patSpiderizer
- **Author** Sebastian 'The Argh' Mordziol < argh@php-tools.de>
- **Version** 2.0.8
- **Access** public

patSpiderizer::\$amountErrors

string = 0 [*line 45*]

Used to keep track of the amount of errors encountered

patSpiderizer::\$config

```
string = [line 31]
```

Used to store the configuration object needed to retrieve search engine data

patSpiderizer::\$vars

```
string = array( "version" => "2.0.8",  
              "debug"   => "off"  
            ) [line 37]
```

Used to store all needed variables

Constructor function patSpiderizer::patSpiderizer(&\$config, \$locationID) [line 53]

Function Parameters:

- *string* **&\$config** reference to the configuration object used to retrieve data
- **\$locationID**

patSpiderizer constructor, stores vital information for later use.

- **Access** public

function patSpiderizer::runCheck() [line 150]

runs the search engine check with the current useragent and IP, and displays spider page or sends a header to the exit page according to the check result.

- **Access** public

function patSpiderizer::setAdminMail(\$email) [line 116]

Function Parameters:

- *string* **\$email** the email address to set

sets the email address of the admin of the script - use this if you want to override the default address from the main config file and specify custom admins for the different locations of the patSpiderizer script.

- **Access public**

function patSpiderizer::setDebugMode(\$state) [*line 128*]

Function Parameters:

- *string* **\$state** The mode to set - "on" activates it. "off" is default.

can be used to make patSpiderizer go into debug mode to make a dump of all his internal variables and checking process

- **See** debug()
- **Access public**

function patSpiderizer::setExitPage(\$url) [*line 105*]

Function Parameters:

- *string* **\$url** the complete http url (or page name if it is in the same directory than the script) to jump to

sets the name of the page that normal users should be directed to (in case no search engine has been recognized)

- **Access public**

function patSpiderizer::setIp(\$ip) [*line 93*]

Function Parameters:

- *string* **\$ip** the ip to cross-check the search engine database for

sets the ip used to recognize a search engine from if the option is enabled.

- **Access** public

function patSpiderizer::setOption(\$option, \$state) [*line 140*]

Function Parameters:

- *string* **\$option** the name of the option to set
- *string* **\$state** the state to set that option to (on|off)

set a patSpiderizer option - used this to override any default option as needed, which are: ipcheck (on|off) / logging (on|off) / errormailings (on|off).

- **Access** public

function patSpiderizer::setRootPath(\$rootPath) [*line 82*]

Function Parameters:

- *string* **\$rootPath** the root path to the spiderizer files.

sets the useragent string used to recognize a search engine with

- **Access** public

function patSpiderizer::setUseragent(\$useragent) [*line 71*]

Function Parameters:

- *string* **\$useragent** the useragent to cross-check the search engine database for

sets the useragent string used to recognize a search engine with

- **Access** public

Class patSpiderizer_Admin

[*line 35*]

Administration Interface for the patSpiderizer search engine cloaking script

- **Package** patSpiderizer
- **Author** Sebastian 'The Argh' Mordziol < argh@php-tools.de>
- **Version** 2.0.8
- **Access** public

patSpiderizer_Admin::\$adminfile

string = [*line 74*]

Used to store the name of the administration script

patSpiderizer_Admin::\$config

```
object = [line 56]
```

Used to store the template object needed for the administration's output

patSpiderizer_Admin::\$extras

```
object = [line 62]
```

Used to store the extras object used for miscellaneous functions.

patSpiderizer_Admin::\$options

```
array = [line 80]
```

Used to store diverse internal options

patSpiderizer_Admin::\$systemVars

```
array = array(  
    "appName" => "patSpiderizer",  
    "appVersion" => 2.08  
) [line 42]
```

information about current application and version

patSpiderizer_Admin::\$template

```
object = [line 50]
```

Used to store the configuration object needed to retrieve search engine data

patSpiderizer_Admin::\$vars

```
array = [line 68]
```

Used to store post and get variables

```
function patSpiderizer_Admin::connectBlock($id, $bid) [line 3320]
```

Function Parameters:

- **\$id**
- **\$bid**

```
function patSpiderizer_Admin::connectBlocks($id, $bids) [line 3339]
```

Function Parameters:

- **\$id**
- **\$bids**

function patSpiderizer_Admin::connectLink(\$id, \$lid) [*line 3376*]

Function Parameters:

- **\$id**
- **\$lid**

function patSpiderizer_Admin::connectLinks(\$id, \$lids) [*line 3395*]

Function Parameters:

- **\$id**
- **\$lids**

function patSpiderizer_Admin::deleteLink(\$id) [*line 1122*]

Function Parameters:

- **\$id**

function patSpiderizer_Admin::deleteTextBlock(\$id) [*line 1070*]

Function Parameters:

- **\$id**

function patSpiderizer_Admin::displayEnginesDeletelp() [*line 2682*]

function patSpiderizer_Admin::displayEnginesDeleteUseragent() [*line 2626*]

function patSpiderizer_Admin::displayEnginesEditlp() [*line 2573*]

function patSpiderizer_Admin::displayEnginesEditUseragent() [*line 2517*]

function patSpiderizer_Admin::displayTemplatesView() [*line 3950*]

function patSpiderizer_Admin::getApplInfo(\$info) [*line 530*]

Function Parameters:

- **\$info**

Returns information about this application like version, name and such.

function patSpiderizer_Admin::getOptions() [*line 539*]

Retrieves needed options and stores them in the options variable.

function patSpiderizer_Admin::getRandomConfirmTexts() [*line 979*]

function patSpiderizer_Admin::setAdminfile(\$file) [*line 110*]

Function Parameters:

- *string* **\$file** the filename without paths of the main administration script

sets the name of the main administration script

- **Access** public

function patSpiderizer_Admin::setConfigReference(&\$config) [*line 99*]

Function Parameters:

- *string* **&\$config** the configuration object reference

sets a reference to the configuration object needed for the administration data management

- **Access** public

function patSpiderizer_Admin::setExtrasReference(&\$extras, \$file) [*line 121*]

Function Parameters:

- *string* **\$file** the filename without paths of the main administration script
- **&\$extras**

sets the name of the main administration script

- **Access** public

function patSpiderizer_Admin::setTemplateReference(&\$template) [*line 88*]

Function Parameters:

- *string* **&\$template** the template object reference

sets a reference to the template object needed for the administration output

- **Access** public

function patSpiderizer_Admin::start() [*line 149*]

triggers the start of the administration tool - determines what course of action to take.

- **Access** public

Class patXMLRpcClient

[*line 10*]

class patXMLRpcClient

- **Package** patSpiderizer
- **Author** Gerd Schaufelberger < gerd@php-tools.de>, Stephan Schmidt <schst@php-tools.de>
- **Version** 0.1

patXMLRpcClient::\$client

mixed = [line 13]

patXMLRpcClient::\$errors

mixed = array() [line 15]

Constructor function patXMLRpcClient::patXMLRpcClient() [line 25]

constructor

create client

- **Access** public

mixed function patXMLRpcClient::checkForUpdate(\$project, \$version) [line 90]

Function Parameters:

- **\$project**
- **\$version**

checkForUpdate

returns information for asked projects

- **Access** public

mixed function patXMLRpcClient::getProjectList() [line 71]

getProjectList

returns information for all available Projects

- **Access** public

float function patXMLRpcClient::getServerVersion() [*line 132*]
get version of the XML-RPC server

- **Access** public

mixed function patXMLRpcClient::getSupportedApps() [*line 113*]
get apps that support patExtras

- **Access** public

Class xmlrpcmsg [*line 560*]

- **Package** patSpiderizer

xmlrpcmsg::\$debug

mixed = 0 [*line 564*]

xmlrpcmsg::\$methodname

mixed = [line 562]

xmlrpcmsg::\$params

mixed = array() [line 563]

xmlrpcmsg::\$payload

mixed = [line 561]

Constructor function xmlrpcmsg::xmlrpcmsg(\$meth, [\$pars = 0]) *[line 566]*

Function Parameters:

- **\$meth**
- **\$pars**

function xmlrpcmsg::addParam(\$par) *[line 610]*

Function Parameters:

- **\$par**

function xmlrpcmsg::createPayload() *[line 582]*

function xmlrpcmsg::getNumParams() *[line 612]*

function xmlrpcmsg::getParam(\$i) *[line 611]*

Function Parameters:

- **\$i**

function xmlrpcmsg::method([\$meth = ""]) *[line 598]*

Function Parameters:

- **\$meth**

function xmlrpcmsg::parseResponse([\$data = ""]) *[line 623]*

Function Parameters:

- **\$data**

function xmlrpcmsg::parseResponseFile(\$fp) *[line 614]*

Function Parameters:

- **\$fp**

function xmlrpcmsg::serialize() [*line 605*]
function xmlrpcmsg::xml_footer() [*line 578*]
function xmlrpcmsg::xml_header() [*line 574*]

Class xmlrpcresp [*line 507*]

- **Package** patSpiderizer

xmlrpcresp::\$fn

mixed = [*line 509*]

xmlrpcresp::\$fs

mixed = [*line 510*]

xmlrpcresp::\$hdrs

mixed = [*line 511*]

xmlrpcresp::\$xv

mixed = [*line 508*]

Constructor function xmlrpcresp::xmlrpcresp(\$val, [\$fcode = 0], [\$fstr = ""]) [*line 513*]

Function Parameters:

- **\$val**
- **\$fcode**
- **\$fstr**

function xmlrpcresp::faultCode() [*line 524*]
function xmlrpcresp::faultString() [*line 531*]
function xmlrpcresp::serialize() [*line 534*]
function xmlrpcresp::value() [*line 532*]

Class xmlrpcval

[line 715]

- **Package** patSpiderizer

xmlrpcval::\$me

mixed = array() [line 716]

xmlrpcval::\$mytype

mixed = 0 [line 717]

Constructor function xmlrpcval::xmlrpcval([\$val = -1], [\$type = ""]) [line 719]

Function Parameters:

- **\$val**
- **\$type**

function xmlrpcval::addArray(\$vals) [line 771]

Function Parameters:

- **\$vals**

function xmlrpcval::addScalar(\$val, [\$type = "string"]) [line 735]

Function Parameters:

- **\$val**
- **\$type**

function xmlrpcval::addStruct(\$vals) [line 784]

Function Parameters:

- **\$vals**

function xmlrpcval::arraymem(\$m) [*line 943*]

Function Parameters:

- **\$m**

function xmlrpcval::arraysize() [*line 948*]

function xmlrpcval::dump(\$ar) [*line 796*]

Function Parameters:

- **\$ar**

function xmlrpcval::getval() [*line 897*]

function xmlrpcval::kindOf() [*line 807*]

function xmlrpcval::scalartyp() [*line 934*]

function xmlrpcval::scalarval() [*line 927*]

function xmlrpcval::serialize() [*line 868*]

function xmlrpcval::serializedata(\$typ, \$val) [*line 823*]

Function Parameters:

- **\$typ**
- **\$val**

function xmlrpcval::serializeval(\$o) [*line 872*]

Function Parameters:

- **\$o**

function xmlrpcval::structeach() [*line 893*]

function xmlrpcval::structmem(\$m) [*line 884*]

Function Parameters:

- **\$m**

function xmlrpcval::structreset() [*line 889*]

Class xmlrpc_client

- **Package** patSpiderizer

xmlrpc_client::\$cert

mixed = "" [line 362]

xmlrpc_client::\$certpass

mixed = "" [line 363]

xmlrpc_client::\$debug

mixed = 0 [line 359]

xmlrpc_client::\$errno

mixed = [line 357]

xmlrpc_client::\$errstring

mixed = [line 358]

xmlrpc_client::\$password

mixed = "" [line 361]

xmlrpc_client::\$path

mixed = [line 354]

xmlrpc_client::\$port

mixed = [line 356]

xmlrpc_client::\$server

mixed = [line 355]

xmlrpc_client::\$username

mixed = "" [line 360]

Constructor function xmlrpc_client::xmlrpc_client(\$path, \$server, [\$port = 0]) [line 365]

Function Parameters:

- **\$path**
- **\$server**
- **\$port**

function xmlrpc_client::send(\$msg, [\$timeout = 0], [\$method = 'http']) *[line 387]*

Function Parameters:

- **\$msg**
- **\$timeout**
- **\$method**

function xmlrpc_client::sendPayloadHTTP10(\$msg, \$server, \$port, [\$timeout = 0], [\$username = ""], [\$password = ""]) *[line 405]*

Function Parameters:

- **\$msg**
- **\$server**
- **\$port**
- **\$timeout**
- **\$username**
- **\$password**

function xmlrpc_client::sendPayloadHTTPS(\$msg, \$server, \$port, [\$timeout = 0], [\$username = ""], [\$password = ""], [\$cert = ""], [\$certpass = ""]) *[line 446]*

Function Parameters:

- **\$msg**
- **\$server**
- **\$port**
- **\$timeout**
- **\$username**
- **\$password**
- **\$cert**
- **\$certpass**

function xmlrpc_client::setCertificate(\$cert, \$certpass) *[line 382]*

Function Parameters:

- **\$cert**
- **\$certpass**

function xmlrpc_client::setCredentials(\$u, \$p) *[line 377]*

Function Parameters:

- **\$u**

- **\$p**

function xmlrpc_client::setDebug(\$in) [*line 369*]

Function Parameters:

- **\$in**

Package patConfiguration Classes

Class patConfiguration

[line 12]

patConfiguration Class to read XML config files \$Id: patConfiguration.php 2 2004-03-02 23:07:44Z argh \$

- **Package** patConfiguration
- **Author** Stephan Schmidt < schst@php-tools.de>
- **Version** \$Revision: 2 \$
- **Access** public

patConfiguration::\$conf

array = array() [line 35]

array that stores configuration

patConfiguration::\$data

string = "" [line 65]

current CDATA found

patConfiguration::\$extensions

array = array() [line 41]

array that stores extensions

patConfiguration::\$nsStack

```
array = array() [line 47]
```

stack of the namespaces

patConfiguration::\$path

```
array = array() [line 29]
```

current path as array

patConfiguration::\$valDepth

```
int = 1 [line 59]
```

current depth of the stored values, i.e. array depth

patConfiguration::\$valStack

```
array = array() [line 53]
```

stack of values

patConfiguration::\$xmlSpecialchars

```
array = array(  
    "&"    => "&",  
    "'"    => "&apos;",  
    "\"    => \"\"\",  
    "<"    => "<",  
    ">"    => ">"  
    ) [line 18]
```

table used for translation of xml special chars

```
function patConfiguration::addExtension(&$ext, [$ns = ""]) [line 361]
```

Function Parameters:

- *object* **patConfigExtension** **&\$ext** extension that should be added
- *string* **\$ns** namespace for this extension (if differs from default ns)

add an extension

- **Access** public

function patConfiguration::appendData(\$data) [*line 592*]

Function Parameters:

- *mixed* **\$data** data to be appended

append Data ti the current data

function patConfiguration::characterData(\$parser, \$data) [*line 532*]

Function Parameters:

- *int* **\$parser** resource id of the current parser
- *string* **\$data** character data, that was found

handle character data if the character data was found between tags using namespaces, the appropriate namesapce handler will be called

function patConfiguration::clearConfigValue([\$path = ""]) [*line 787*]

Function Parameters:

- *string* **\$path** path, where the value is stored

clears a config value if no path is given, the complete config will be cleared

- **Access** public

object function patConfiguration::createParser() [*line 868*]

create a parser

function patConfiguration::endElement(\$parser, \$name) [line 463]

Function Parameters:

- *int* **\$parser** resource id of the current parser
- *string* **\$name** name of the element

handle end element handler if the end element contains a namespace calls the appropriate handler

function patConfiguration::externalEntity(\$parser, \$openEntityNames, \$base, \$systemId, \$publicId) [line 827]

Function Parameters:

- **\$parser**
- **\$openEntityNames**
- **\$base**
- **\$systemId**
- **\$publicId**

mixed function patConfiguration::getConfigValue([\$path = ""]) [line 718]

Function Parameters:

- *string* **\$path** path, where the value is stored

returns a configuration value if no path is given, all config values will be returned in an array

- **Access public**

function patConfiguration::parseConfigFile(\$file, [\$mode = "w"]) [line 96]

Function Parameters:

- *string* **\$file** name of the configuration file
- *string* **\$mode** mode of the parsing ("w" = overwrite old config, "a" = append to config)

parse a configuration file

- **Access** public

function patConfiguration::parseXMLFile(\$file) [*line 843*]

Function Parameters:

- *string* **\$file** filename, without dirname

parse an external xml file

string function patConfiguration::replaceXMLSpecialchars(\$string, [\$table = array()]) [*line 891*]

Function Parameters:

- *string* **\$string** string, where special chars should be replaced
- *array* **\$table** table used for replacing

replace XML special chars

function patConfiguration::setConfigDir(\$configDir) [*line 73*]

Function Parameters:

- *string* **\$configDir** name of the directory

set the directory, where all xml config files are stored

- **Access** public

function patConfiguration::setConfigValue(\$path, \$value, [\$type = "leave"]) [*line 760*]

Function Parameters:

- *string* **\$path** path, where the value will be stored
- *mixed* **\$value** value to store
- **\$type**

set a config value

*

- **Access** public

function patConfiguration::setConfigValues(\$values) [*line 771*]

Function Parameters:

- *array* **\$values** assoc array containg paths and values

sets several config values

- **Access** public

function patConfiguration::setIncludeDir(\$includeDir) [*line 84*]

Function Parameters:

- *string* **\$includeDir** name of the directory

set the directory, where all extensions are stored

- **Access** public

function patConfiguration::startElement(\$parser, \$name, \$attributes) [*line 379*]

Function Parameters:

- *int* **\$parser** resource id of the current parser
- *string* **\$name** name of the element
- *array* **\$attributes** array containing all attributes of the element

handle start element handler if the start element contains a namespace calls the appropriate handler

function patConfiguration::writeConfigFile(\$filename, [\$format = "xml"], [\$options = array()]) [*line 116*]

Function Parameters:

- *string* **\$filename** name of the configfile
- *string* **\$format** format of the config file (xml or php)
- *array* **\$options** available options for php: varname => anyString ; available options for xml: mode => pretty

write a configfile format may be php or xml

- **Access** public

Package patTemplate Procedural Elements

patTemplate.php

- **Package** patTemplate

patTEMPLATE_TAG_END = "}" *[line 15]*

Variable suffix

- **Access** public

patTEMPLATE_TAG_START = "{" *[line 8]*

Variable prefix

- **Access** public

patTEMPLATE_TYPE_CONDITION = "CONDITION" *[line 31]*

Template type Condition

patTEMPLATE_TYPE_ODDEVEN = "ODDEVEN" *[line 26]*

Template type OddEven

patTEMPLATE_TYPE_SIMPLECONDITION = "SIMPLECONDITION" *[line 36]*

Template type SimpleCondition

patTEMPLATE_TYPE_STANDARD = "STANDARD" *[line 21]*

Template type Standard

Package patTemplate Classes

Class patTemplate

[line 49]

Easy-to-use but powerful template engine

Features include: several templates in one file, automatic repetitions, global variables, alternating lists, conditions, and much more

- **Package** patTemplate
- **Version** 2.4 (\$Id: patTemplate.php 2 2004-03-02 23:07:44Z argh \$)
- **Author** Stephan Schmidt < schst@php-tools.de>
- **Access** public

Constructor function patTemplate::patTemplate([\$type = "html"]) [line 63]

Function Parameters:

- *string* **\$type** type of output you want to generate.

Constructor

Create new patTemplate object You can choose between two outputs you want to generate: html (default) or tex (LaTeX). When "tex" is used the patTemplate markings used for variables are changed as LaTeX makes use of the default patTemplate markings. You can also change the markings later by calling setTags();

- **Access** public

function patTemplate::addGlobalVar(\$name, \$value) [*line 888*]

Function Parameters:

- *string* **\$name** name of the global variable
- *string* **\$value** value of the variable

Adds a global variable

Global variables are valid in all templates of this object

- **See** [patTemplate::addGlobalVars\(\)](#), [patTemplate::addVar\(\)](#), [patTemplate::addVars\(\)](#), [patTemplate::addRows\(\)](#)
- **Access** public

function patTemplate::addGlobalVars(\$variables, [\$prefix = ""]) [*line 904*]

Function Parameters:

- *array* **\$variables** array containing the variables
- *string* **\$prefix** prefix for variable names

Adds several global variables

Global variables are valid in all templates of this object \$variables is an associative array, containing name/value pairs of the variables

- **See** [patTemplate::addGlobalVar\(\)](#), [patTemplate::addVar\(\)](#), [patTemplate::addVars\(\)](#), [patTemplate::addRows\(\)](#)
- **Access** public

function patTemplate::addRows(\$template, \$rows, [\$prefix = ""]) [*line 846*]

Function Parameters:

- *string* **\$template** name of the template
- *array* **\$rows** array containing associative arrays with variable/value pairs
- *string* **\$prefix** prefix for all variable names

Adds several rows of variables to a template

Each Template can have an unlimited amount of its own variables. Can be used to add a database result as variables to a template.

- See [patTemplate::addVar\(\)](#), [patTemplate::addVars\(\)](#), [patTemplate::addGlobalVar\(\)](#), [patTemplate::addGlobalVars\(\)](#)
- **Access** public

function patTemplate::addTemplate(\$name, \$filename) [*line 183*]

Function Parameters:

- *string* **\$name** name of the template
- *string* **\$filename** filename of the sourcetemplate

Add a template

Adds a plain text/html to the template engine. The file has to be in the directory that has been set using `setBaseDir`.

- See `setBaseDir()`, [patTemplate::addTemplates\(\)](#)
- **Deprecated** 2.4 2001/11/05
- **Access** public

function patTemplate::addTemplates(\$templates) [*line 203*]

Function Parameters:

- *array* **\$templates** associative Array with name/filename pairs

Adds several templates

Adds several templates to the template engine using an associative array. Names of the templates are stored in the keys, filenames are the values. The templates have to be in the directory set by `setBaseDir()`.

- **See** `setBaseDir()`, [patTemplate::addTemplate\(\)](#)
- **Deprecated** 2.4 2001/11/05
- **Access** public

```
function patTemplate::addVar($template, $name, $value) [line 788]
```

Function Parameters:

- *string* **\$template** name of the template
- *string* **\$name** name of the variables
- *mixed* **\$value** value of the variable

Adds a variable to a template

Each Template can have an unlimited amount of its own variables

- **See** [patTemplate::addVars\(\)](#), [patTemplate::addRows\(\)](#), [patTemplate::addGlobalVar\(\)](#), [patTemplate::addGlobalVars\(\)](#)
- **Access** public

```
function patTemplate::addVars($template, $variables, [$prefix = ""]) [line 820]
```

Function Parameters:

- *string* **\$template** name of the template
- *array* **\$variables** associative array of the variables
- *string* **\$prefix** prefix for all variable names

Adds several variables to a template

Each Template can have an unlimited amount of its own variables. \$variables has to be an associative array containing variable/value pairs

- See [patTemplate::addVar\(\)](#), [patTemplate::addRows\(\)](#), [patTemplate::addGlobalVar\(\)](#), [patTemplate::addGlobalVars\(\)](#)
- Access public

function patTemplate::clearAllTemplates() [*line 1409*]

clears all templates

- Access public

function patTemplate::clearAttribute(\$template, \$attribute) [*line 356*]

Function Parameters:

- *string* **\$template** name of the template
- *string* **\$attribute** name of the attribute

Clears an attribute of a template

supported attributes: visibilty, loop, parse, unusedvars

- See [patTemplate::setAttribute\(\)](#), [patTemplate::setAttributes\(\)](#), [patTemplate::getAttribute\(\)](#)
- Access public

function patTemplate::clearTemplate(\$name) [*line 1395*]

Function Parameters:

- *string* **\$name** name of the template

clears a parsed Template

parsed Content, variables and the loop attribute are cleared

- **Access** public

function patTemplate::displayParsedTemplate([\$name = ""]) [*line 1301*]

Function Parameters:

- *string* **\$name** name of the template

displays a parsed Template

If the template has not been loaded, it will be loaded.

- **See** [patTemplate::getParsedTemplate\(\)](#)
- **Access** public

function patTemplate::dump() [*line 1547*]

displays useful information about all templates

returns content, variables, attributes and unused variables

- **Access** public

bool function patTemplate::exists(\$name) [*line 160*]

Function Parameters:

- *string* **\$name** name of the template

Check if a template exists

- **Access** public

mixed function patTemplate::getAttribute(\$template, \$attribute) [*line 338*]

Function Parameters:

- *string* **\$template** name of the template
- *string* **\$attribute** name of the attribute

Gets an attribute of a template

supported attributes: visibilty, loop, parse, unusedvars

- See [patTemplate::setAttribute\(\)](#), [patTemplate::setAttributes\(\)](#), [patTemplate::clearAttribute\(\)](#)
- **Access** public

string function patTemplate::getParsedTemplate([\$name = ""]) [*line 1245*]

Function Parameters:

- *string* **\$name** name of the template

returns a parsed Template

If the template already has been parsed, it just returns the parsed template. If the template has not been loaded, it will be loaded.

- See [patTemplate::displayParsedTemplate\(\)](#)
- Access public

mixed function patTemplate::getVar(\$template, \$var, \$index) *[line 1514]*

Function Parameters:

- *string* **\$template** name of the template
- *string* **\$var** name of the variable
- *integer* **\$index** no of repetition

get the value of a variable

function patTemplate::parseTemplate(\$template, [\$mode = "w"]) *[line 987]*

Function Parameters:

- *string* **\$template** name of the template
- *string* **\$mode** mode for the parsing

parses a template

Parses a template and stores the parsed content. mode can be "w" for write (delete already parsed content) or "a" for append (appends the new parsed content to the already parsed content)

- See [parseStandardTemplate\(\)](#), [parseliterativeTemplate\(\)](#)
- Access public

function patTemplate::readTemplatesFromFile(\$file) *[line 394]*

Function Parameters:

- *string* **\$file** filename

Parses several templates from one patTemplate file

Templates can be separated using Tags. The file has to be located in the directory that has been set using `setBaseDir`.

- See [patTemplate::setBasedir\(\)](#)
- Access public

function patTemplate::setAttribute(\$template, \$attribute, \$value) [*line 293*]

Function Parameters:

- *string* **\$template** name of the template
- *string* **\$attribute** name of the attribute
- *mixed* **\$value** value of the attribute

Sets an attribute of a template

supported attributes: visibility, loop, parse, unusedvars

- See [patTemplate::setAttributes\(\)](#), [patTemplate::getAttribute\(\)](#), [patTemplate::clearAttribute\(\)](#)
- Access public

function patTemplate::setAttributes(\$template, \$attributes) [*line 312*]

Function Parameters:

- *string* **\$template** name of the template
- *array* **\$attributes** attribute/value pairs

Sets several attribute of a template

`$attributes` has to be an associative array containing attribute/value pairs. supported attributes: visibility, loop, parse, unusedvars

- See [patTemplate::setAttribute\(\)](#), [patTemplate::getAttribute\(\)](#), [patTemplate::clearAttribute\(\)](#)
- Access public

function patTemplate::setBasedir(\$basedir) [*line 148*]

Function Parameters:

- *string* **\$basedir** directory of the templates

Set template directory

Sets the directory where the template are stored. By default the engine looks in the directory where the original file is stored.

- Access public

function patTemplate::setTags([\$start = patTEMPLATE_TAG_START], [\$end = patTEMPLATE_TAG_END]) [*line 131*]

Function Parameters:

- *string* **\$start** start tag
- *string* **\$end** end tag

Set template tags

Sets the start and end tags of template variables

- Access public

function patTemplate::setType([\$type = ""]) [*line 107*]

Function Parameters:

- *string* **\$type** predefined template type, like "html" or "tex"

Set template type

select a predefined template type

- **Access** public

Appendices

Appendix A - Class Trees

Package patSpiderizer

patExtras

- [patExtras](#)

patSpiderizer

- [patSpiderizer](#)

patSpiderizer_Admin

- [patSpiderizer_Admin](#)

patXMLRpcClient

- [patXMLRpcClient](#)

xmlrpcmsg

- [xmlrpcmsg](#)

xmlrpcresp

- [xmlrpcresp](#)

xmlrpcval

- [xmlrpcval](#)

xmlrpc_client

- [xmlrpc_client](#)

Package patConfiguration

patConfiguration

- [patConfiguration](#)

Package patTemplate

patTemplate

- [patTemplate](#)

Index

A

[addVars\(\)](#) 3

C

[constructor xmlrpcval::xmlrpcval\(\)](#) 26

[constructor xmlrpc_client::xmlrpc_client\(\)](#) 28

[constructor patTemplate::patTemplate\(\)](#) 41

Constructor

[constructor xmlrpcresp::xmlrpcresp\(\)](#) 25

[constructor xmlrpcmsg::xmlrpcmsg\(\)](#) 24

[constructor patSpiderizer::patSpiderizer\(\)](#) 14

patSpiderizer constructor, stores vital information for later use.

[constructor patXMLRpcClient::patXMLRpcClient\(\)](#) 22

constructor

[checkInstallation\(\)](#) 2

D

[displayErrors\(\)](#) 2

E

[example.php](#) 3

I

[iso8601_encode\(\)](#) 5

[iso8601_decode\(\)](#) 5

P

[patConfiguration::getConfigValue\(\)](#) 34

returns a configuration value

if no path is given, all config values will be returned in an array

[patConfiguration::parseConfigFile\(\)](#) 34

parse a configuration file

[patConfiguration::externalEntity\(\)](#) 34

[patConfiguration::endElement\(\)](#) 34

handle end element

<i>if the end element contains a namespace calls the appropriate handler</i>	
patConfiguration::clearConfigValue()	33
<i>clears a config value</i>	
<i>if no path is given, the complete config will be cleared</i>	
patConfiguration::createParser()	33
<i>create a parser</i>	
patConfiguration::parseXMLFile()	35
<i>parse an external xml file</i>	
patConfiguration::replaceXMLSpecialchars()	35
<i>replace XML special chars</i>	
patConfiguration::startElement()	37
<i>handle start element</i>	
<i>if the start element contains a namespace calls the appropriate handler</i>	
patConfiguration::writeConfigFile()	37
<i>write a configfile</i>	
<i>format may be php or xml</i>	
patConfiguration::setIncludeDir()	36
<i>set the directory, where all extensions are stored</i>	
patConfiguration::setConfigValues()	36
<i>sets several config values</i>	
patConfiguration::setConfigDir()	35
<i>set the directory, where all xml config files are stored</i>	
patConfiguration::setConfigValue()	35
<i>set a config value</i>	
patConfiguration::characterData()	33
<i>handle character data</i>	
<i>if the character data was found between tags using namespaces, the appropriate namespace handler will be called</i>	
patConfiguration::appendData()	33
<i>append Data ti the current data</i>	
patConfiguration	31
<i>patConfiguration</i>	
<i>Class to read XML config files</i>	
<i>\$Id: patConfiguration.php 2 2004-03-02 23:07:44Z argh \$</i>	
patConfiguration::\$conf	31
<i>array that stores configuration</i>	
patXMLRpcClient::getSupportedApps()	23
<i>get apps that support patExtras</i>	
patXMLRpcClient::getServerVersion()	23
<i>get version of the XML-RPC server</i>	
patXMLRpcClient::checkForUpdate()	22
<i>checkForUpdate</i>	
patXMLRpcClient::getProjectList()	22
<i>getProjectList</i>	
patConfiguration::\$data	31
<i>current CDATA found</i>	
patConfiguration::\$extensions	31
<i>array that stores extensions</i>	
patConfiguration::\$xmlSpecialchars	32
<i>table used for translation of xml special chars</i>	
patConfiguration::addExtension()	32
<i>add an extension</i>	
patConfiguration::\$valStack	32

<i>stack of values</i>	
patConfiguration::\$valDepth	32
<i>current depth of the stored values, i.e. array depth</i>	
patConfiguration::\$nsStack	32
<i>stack of the namespaces</i>	
patConfiguration::\$path	32
<i>current path as array</i>	
patTemplate.php	39
patTEMPLATE_TAG_END	39
<i>Variable suffix</i>	
patTemplate::getAttribute()	47
<i>Gets an attribute of a template</i>	
patTemplate::getParsedTemplate()	47
<i>returns a parsed Template</i>	
patTemplate::exists()	46
<i>Check if a template exists</i>	
patTemplate::dump()	46
<i>displays useful information about all templates</i>	
patTemplate::clearTemplate()	45
<i>clears a parsed Template</i>	
patTemplate::displayParsedTemplate()	46
<i>displays a parsed Template</i>	
patTemplate::getVar()	48
<i>get the value of a variable</i>	
patTemplate::parseTemplate()	48
<i>parses a template</i>	
patTemplate::setTags()	50
<i>Set template tags</i>	
patTemplate::setType()	51
<i>Set template type</i>	
patTemplate::setBasedir()	50
<i>Set template directory</i>	
patTemplate::setAttributes()	49
<i>Sets several attribute of a template</i>	
patTemplate::readTemplatesFromFile()	48
<i>Parses several templates from one patTemplate file</i>	
patTemplate::setAttribute()	49
<i>Sets an attribute of a template</i>	
patTemplate::clearAttribute()	45
<i>Clears an attribute of a template</i>	
patTemplate::clearAllTemplates()	45
<i>clears all templates</i>	
patTEMPLATE_TYPE_STANDARD	40
<i>Template type Standard</i>	
patTemplate	41
<i>Easy-to-use but powerful template engine</i>	
patTEMPLATE_TYPE_SIMPLECONDITION	39
<i>Template type SimpleCondition</i>	
patTEMPLATE_TYPE_ODDEVEN	39
<i>Template type OddEven</i>	
patTEMPLATE_TAG_START	39
<i>Variable prefix</i>	
patTEMPLATE_TYPE_CONDITION	39

<i>Template type Condition</i>	
patTemplate::addGlobalVar()	42
<i>Adds a global variable</i>	
patTemplate::addGlobalVars()	42
<i>Adds several global variables</i>	
patTemplate::addVar()	44
<i>Adds a variable to a template</i>	
patTemplate::addVars()	44
<i>Adds several variables to a template</i>	
patTemplate::addTemplates()	43
<i>Adds several templates</i>	
patTemplate::addTemplate()	43
<i>Add a template</i>	
patTemplate::addRows()	43
<i>Adds several rows of variables to a template</i>	
patXMLRpcClient::\$errors	22
patXMLRpcClient::\$client	22
patExtras::writeConfig()	13
<i>write the extras configuration file (csv formatted)</i>	
patSpiderizer	13
<i>patSpiderizer search engine cloaking script</i>	
patExtras::setXMLRpcClient()	12
<i>set the XMLRpc Client object needed to retrieve data from the php-tools website</i>	
patExtras::setTemplate()	12
<i>set the template object, that should be used for rendering</i>	
patExtras::reportErrors()	12
<i>report (display) errors</i>	
patExtras::setAdminInterface()	12
<i>set reference to Randy</i>	
patSpiderizer::\$amountErrors	13
<i>Used to keep track of the amount of errors encountered</i>	
patSpiderizer::\$config	14
<i>Used to store the configuration object needed to retrieve search engine data</i>	
patSpiderizer::setExitPage()	15
<i>sets the name of the page that normal users should be directed to (in case no search engine has been recognized)</i>	
patSpiderizer::setIp()	16
<i>sets the ip used to recognize a search engine from if the option is enabled.</i>	
patSpiderizer::setDebugMode()	15
<i>can be used to make patSpiderizer go into debug mode to make a dump of all his internal variables and checking process</i>	
patSpiderizer::setAdminMail()	14
<i>sets the email adress of the admin of the script - use this if you want to override the default adress from the main config file and specify custom admins for the different locations of the patSpiderizer script.</i>	
patSpiderizer::\$vars	14
<i>Used to store all needed variables</i>	
patSpiderizer::runCheck()	14
<i>runs the search engine check with the current useragent and IP, and displays spider page or sends a header to the exit page according to the check result.</i>	
patExtras::readConfig()	11
<i>read the extras configuration file (csv formatted)</i>	
patExtras::process()	11

<i>run the extras administration interface</i>	9
patExtras::\$systemVars	9
<i>information about patExtras</i>	
patExtras::\$updateIntervals	9
<i>intervals for the "check for updates selector"</i>	
patExtras::\$errors	8
<i>array to store all errors</i>	
patExtras::\$errorCodes	8
patSpiderizerAdmin.php	4
patExtras	8
<i>pat Administration Extras</i>	
patExtras::addError()	9
<i>add an error</i>	
patExtras::displayMessage()	9
<i>Displays a message (success, general notes, whatever...)</i>	
patExtras::getLastErrorCode()	11
<i>return code of the last error</i>	
patExtras::getLastErrorMessage()	11
<i>return message of the last error</i>	
patExtras::getLastError()	10
<i>return code and message of the last error</i>	
patExtras::getErrors()	10
<i>return code and message of all errors</i>	
patExtras::getErrorCodes()	10
<i>return code of all errors</i>	
patExtras::getErrorMessages()	10
<i>return message of all errors</i>	
patSpiderizer::setOption()	16
<i>set a patSpiderizer option - used this to override any default option as needed, which are: ipcheck (on off) / logging (on off) / errormailings (on off).</i>	
patSpiderizer::setRootPath()	16
<i>sets the useragent string used to recognize a search engine with</i>	
patSpiderizer Admin::displayTemplatesView()	19
patSpiderizer Admin::getAppInfo()	19
<i>Returns information about this application like version, name and such.</i>	
patSpiderizer Admin::displayEnginesEditUseragent()	19
patSpiderizer Admin::displayEnginesEditIp()	19
patSpiderizer Admin::displayEnginesDeleteIp()	19
patSpiderizer Admin::displayEnginesDeleteUseragent()	19
patSpiderizer Admin::getOptions()	20
<i>Retrieves needed options and stores them in the options variable.</i>	
patSpiderizer Admin::getRandomConfirmTexts()	20
patSpiderizer Admin::start()	21
<i>triggers the start of the administration tool - determines what course of action to take.</i>	
patXMLRpcClient	21
<i>class patXMLRpcClient</i>	
patSpiderizer Admin::setTemplateReference()	21
<i>sets a reference to the template object needed for the administration output</i>	
patSpiderizer Admin::setExtrasReference()	20
<i>sets the name of the main administration script</i>	
patSpiderizer Admin::setAdminfile()	20
<i>sets the name of the main administration script</i>	
patSpiderizer Admin::setConfigReference()	20

	<i>sets a reference to the configuration object needed for the administration data management</i>	
patSpiderizer Admin::deleteTextBlock()		19
patSpiderizer Admin::deleteLink()		19
patSpiderizer Admin::\$extras		18
	<i>Used to store the extras object used for miscellaneous functions.</i>	
patSpiderizer Admin::\$options		18
	<i>Used to store diverse internal options</i>	
patSpiderizer Admin::\$config		18
	<i>Used to store the template object needed for the administration's output</i>	
patSpiderizer Admin::\$adminfile		17
	<i>Used to store the name of the administration script</i>	
patSpiderizer::setUseragent()		17
	<i>sets the useragent string used to recognize a search engine with</i>	
patSpiderizer Admin		17
	<i>Administration Interface for the patSpiderizer search engine cloaking script</i>	
patSpiderizer Admin::\$systemVars		18
	<i>information about current application and version</i>	
patSpiderizer Admin::\$template		18
	<i>Used to store the configuration object needed to retrieve search engine data</i>	
patSpiderizer Admin::connectLink()		19
patSpiderizer Admin::connectLinks()		19
patSpiderizer Admin::connectBlocks()		18
patSpiderizer Admin::connectBlock()		18
patSpiderizer Admin::\$vars		18
	<i>Used to store post and get variables</i>	
patSpiderizer.php		2

U

UPDATE_DBVERSIONURL		4
	<i>sets the url of the search engine database update server's version file</i>	
UPDATE_DBURL		4
	<i>sets the url of the search engine database update server's database file</i>	

X

xmlrpcval::serializedata()		27
xmlrpcval::serialize()		27
xmlrpcval::scalarval()		27
xmlrpcval::serializeval()		27
xmlrpcval::structeach()		27
xmlrpcval::structreset()		27
xmlrpcval::structmem()		27
xmlrpcval::scalartyp()		27
xmlrpcval::kindOf()		27
xmlrpcval::addStruct()		26
xmlrpcval::addScalar()		26
xmlrpcval::addArray()		26
xmlrpcval::arraymem()		27
xmlrpcval::arraysize()		27
xmlrpcval::getval()		27

xmlrpcval::dump()	27
xmlrpc_client	28
xmlrpc_client::\$cert	28
xmlrpc_client::sendPayloadHTTP10()	29
xmlrpc_client::send()	29
xmlrpc_client::\$username	28
xmlrpc_client::sendPayloadHTTPS()	29
xmlrpc_client::setCertificate()	29
xmlrpc_client::setDebug()	30
xmlrpc_client::setCredentials()	29
xmlrpc_client::\$server	28
xmlrpc_client::\$port	28
xmlrpc_client::\$debug	28
xmlrpc_client::\$certpass	28
xmlrpc_client::\$errno	28
xmlrpc_client::\$errstring	28
xmlrpc_client::\$path	28
xmlrpc_client::\$password	28
xmlrpcval::\$mytype	26
xmlrpcval::\$me	26
xmlrpcmsg::\$params	24
xmlrpcmsg::\$methodname	24
xmlrpcmsg::\$debug	23
xmlrpcmsg::\$payload	24
xmlrpcmsg::addParam()	24
xmlrpcmsg::getNumParams()	24
xmlrpcmsg::createPayload()	24
xmlrpcmsg	23
xmlrpc_se()	6
xmlrpc_cd()	6
xmlrpcEncode()	5

xmlrpcDecode()	5

xmlrpc_dh()	6
xmlrpc_ee()	6
xmlrpc_lookup_entity()	6
xmlrpc_entity_decode()	6
xmlrpcmsg::getParam()	24
xmlrpcmsg::method()	24
xmlrpcresp::faultCode()	25
xmlrpcresp::\$xv	25
xmlrpcresp::\$hdrs	25
xmlrpcresp::faultString()	25
xmlrpcresp::serialize()	25
xmlrpcval	26
xmlrpcresp::value()	25
xmlrpcresp::\$fs	25
xmlrpcresp::\$fn	25
xmlrpcmsg::parseResponseFile()	24
xmlrpcmsg::parseResponse()	24
xmlrpcmsg::serialize()	25
xmlrpcmsg::xml_footer()	25

xmlrpcresp	25
xmlrpcmsg::xml_header()	25
xmlrpc.inc	5