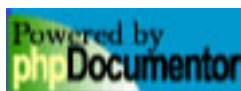


patMessenger



Contents

| | |
|--|----|
| Package patMessenger Procedural Elements | 2 |
| dump_shared_mem.php | 2 |
| messenger.php | 3 |
| Function signoff | 3 |
| Package patMessenger Classes | 4 |
| Class patMessenger | 4 |
| Var \$neededVars | 4 |
| Var \$semKey | 4 |
| Var \$shm | 4 |
| Var \$shmKey | 5 |
| Var \$shmSize | 5 |
| Var \$tmpI | 5 |
| Var \$uid | 5 |
| Var \$varKeys | 5 |
| Var \$vars | 5 |
| Method process | 5 |
| Package patSHMC Procedural Elements | 8 |
| patSHMC.php | 8 |
| Define patEXCLUSIVE_LOCK | 8 |
| Define patRELEASE_LOCK | 8 |
| Define patSHARED_LOCK | 8 |
| Package patSHMC Classes | 9 |
| Class patSHMC | 9 |
| Constructor patSHMC | 9 |
| Method detach | 9 |
| Method get_var | 10 |
| Method lock | 10 |
| Method put_var | 10 |
| Method remove | 11 |
| Method remove_var | 11 |
| Package patTemplate Procedural Elements | 13 |
| patTemplate.php | 13 |
| Define patTEMPLATE_TAG_END | 13 |
| Define patTEMPLATE_TAG_START | 13 |
| Define patTEMPLATE_TYPE_CONDITION | 13 |
| Define patTEMPLATE_TYPE_ODDEVEN | 13 |
| Define patTEMPLATE_TYPE_SIMPLECONDITION | 13 |
| Define patTEMPLATE_TYPE_STANDARD | 14 |
| Package patTemplate Classes | 15 |
| Class patTemplate | 15 |
| Constructor patTemplate | 15 |
| Method addGlobalVar | 16 |

| | |
|--|----|
| Method addGlobalVars | 16 |
| Method addRows | 17 |
| Method addTemplate | 17 |
| Method addTemplates | 17 |
| Method addVar | 18 |
| Method addVars | 18 |
| Method clearAllTemplates | 19 |
| Method clearAttribute | 19 |
| Method clearTemplate | 19 |
| Method displayParsedTemplate | 20 |
| Method dump | 20 |
| Method exists | 20 |
| Method getAttribute | 21 |
| Method getParsedTemplate | 21 |
| Method getVar | 22 |
| Method parseTemplate | 22 |
| Method readTemplatesFromFile | 22 |
| Method setAttribute | 23 |
| Method setAttributes | 23 |
| Method setBasedir | 24 |
| Method setTags | 24 |
| Method setType | 25 |
| Appendices | 26 |
| Appendix A - Class Trees | 27 |
| patMessenger | 27 |
| patSHMC | 27 |
| patTemplate | 27 |
| Appendix D - Todo List | 28 |

Package patMessenger Procedural Elements

dump_shared_mem.php

- **Package** patMessenger

include ["include/patSHMC.php"](#)*[line 6]*

Ausgabe der Informationen im Shared Memory

messenger.php

- **Package** patMessenger

require_once ["include/patTemplate.php"](#)*[line 9]*

require_once ["include/patSHMC.php"](#)*[line 8]*

require_once ["include/patMessenger.php"](#)*[line 7]*

Beispiel für einen Messenger mit PHP4

function signoff() *[line 23]*

Diese Funktion wird aufgerufen, wenn das Skript beendet wird.

Falls der User einfach das Fenster schließt, sollte er trotzdem abgemeldet werden.

Kann manchmal ein paar Sekunden dauern, bis der User abgemeldet wird.

Package patMessenger Classes

Class patMessenger

[line 10]

Messenger Klasse zum Artikel "Interprozesskommunikation mit PHP4 und Shared Memory" im PHP Magazin 01/2002 Diese Klasse soll lediglich zeigen, wie Shared Memory mit PHP4 zur Interprozesskommunikation genutzt werden kann.

Es ist KEINE umfassende Messenger Applikation und nicht auf Performance optimiert.

- **Package** patMessenger
- **Version** 0.1
- **Author** Stephan Schmidt < schst@php-tools.de>

patMessenger::\$neededVars

array = array("action", "name", "recipient", "message", "uid") [line 61]

Name der globalen Variablen, die importiert werden müssen

patMessenger::\$semKey

int = 1977 [line 28]

Key der Semaphore zum Blocken des SHM Segments

patMessenger::\$shm

mixed = [line 34]

Schnittstelle zum Shared Memory Segment

patMessenger::\$shmKey

```
int = 2105 [line 22]
```

Key des Shared Memory Segments

patMessenger::\$shmSize

```
int = 50000 [line 16]
```

Größe des Shared Memory Segments in bytes

patMessenger::\$tmpl

```
mixed = [line 40]
```

patTemplate Objekt für die Ausgabe

patMessenger::\$uid

```
int = [line 46]
```

User ID des aktuellen Benutzers

patMessenger::\$varKeys

```
array = array(  
    "users" => 1,  
    "messages" => 2  
) [line 52]
```

Map, um Variablennamen in Integers zu übersetzen, da Shared Memory Funktionen nur Int Keys akzeptieren

patMessenger::\$vars

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

Array, in dem die importierten Variablen gespeichert werden

function patMessenger::process() [line 76]

**dispatch Function, die alles weitere erledigt
Standardbetrieb keine weiteren Funktionen**

**Ausser dieser Funktion müssen im
aufgerufen werden.**

- **Access public**

Package patSHMC Procedural Elements

patSHMC.php

- **Package** patSHMC

patEXCLUSIVE_LOCK = 2 [*line 4*]
patRELEASE_LOCK = 3 [*line 5*]
patSHARED_LOCK = 1 [*line 3*]

Package patSHMC Classes

Class patSHMC

[line 16]

patSHMC - Shared Memory Connectivity This class can be used to easily handle shared memory including semaphore locking.

- **Package** patSHMC
- **TODO** Shared Locking (using a second semaphore), Error Handling
- **Author** Stephan Schmidt < schst@php-tools.de>
- **Version** 0.1

Constructor function patSHMC::patSHMC(\$shm_key, \$shm_size, \$sem_key, [\$perm = 438]) *[line 28]*

Function Parameters:

- *integer* **\$shm_key** unique key for the shared memory
- *integer* **\$shm_size** size of the shared memory in bytes
- *integer* **\$sem_key** unique key for the semaphore that is used for locking
- *integer* **\$perm** permissions

create a new shared memory connectivity

function patSHMC::detach() *[line 104]*

disconnect from the shared memory

- **Access** public

mixed function patSHMC::get_var(\$key) [*line 83*]

Function Parameters:

- *integer* **\$key** unique key for the variable

read a variable from the shared memory

- **Access** public

function patSHMC::lock(\$operation) [*line 50*]

Function Parameters:

- *integer* **\$operation** 2 = exclusive lock, 3 = free lock

lock or unlock the shared memory

- **Access** public

function patSHMC::put_var(\$key, \$value) [*line 71*]

Function Parameters:

- *integer* **\$key** unique key for the variable
- *mixed* **\$value** value of the variable

write a variable to the shared memory

- **Access** public

function patSHMC::remove() [*line 114*]
remove (clear) shared memory

- **Access** public

function patSHMC::remove_var(\$key) [*line 94*]
Function Parameters:

- *integer* **\$key** unique key for the variable

remove a variable from the shared memory

- **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 2004-03-08 19:49:36Z schst \$)
- **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: `visibilty`, `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 a associative arrays containing attribute/value pairs. supported attributes: `visibilty`, `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 patMessenger

patMessenger

- [patMessenger](#)

Package patSHMC

patSHMC

- [patSHMC](#)

Package patTemplate

patTemplate

- [patTemplate](#)

Appendix D - Todo List

In Package patSHMC

In [patSHMC](#):

- Shared Locking (using a second semaphore), Error Handling

Index

C

| | |
|--|----|
| constructor patTemplate::patTemplate() | 15 |
| <i>Constructor</i> | |
| constructor patSHMC::patSHMC() | 9 |
| <i>create a new shared memory connectivity</i> | |

D

| | |
|---|---|
| dump_shared_mem.php | 2 |
|---|---|

M

| | |
|---|---|
| messenger.php | 3 |
|---|---|

P

| | |
|--|----|
| patTemplate::addVars() | 18 |
| <i>Adds several variables to a template</i> | |
| patTemplate::addVar() | 18 |
| <i>Adds a variable to a template</i> | |
| patTemplate::clearAllTemplates() | 19 |
| <i>clears all templates</i> | |
| patTemplate::clearAttribute() | 19 |
| <i>Clears an attribute of a template</i> | |
| patTemplate::clearTemplate() | 19 |
| <i>clears a parsed Template</i> | |
| patTemplate::addTemplates() | 17 |
| <i>Adds several templates</i> | |
| patTemplate::addTemplate() | 17 |
| <i>Add a template</i> | |
| patTemplate | 15 |
| <i>Easy-to-use but powerful template engine</i> | |
| patTEMPLATE_TYPE_STANDARD | 14 |
| <i>Template type Standard</i> | |
| patTemplate::addGlobalVar() | 16 |
| <i>Adds a global variable</i> | |
| patTemplate::addGlobalVars() | 16 |
| <i>Adds several global variables</i> | |
| patTemplate::addRows() | 17 |
| <i>Adds several rows of variables to a template</i> | |
| patTemplate::displayParsedTemplate() | 20 |
| <i>displays a parsed Template</i> | |

| | |
|---|----|
| patTemplate::dump() | 20 |
| <i>displays useful information about all templates</i> | |
| patTemplate::setAttributes() | 23 |
| <i>Sets several attribute of a template</i> | |
| patTemplate::setAttribute() | 23 |
| <i>Sets an attribute of a template</i> | |
| patTemplate::setBasedir() | 24 |
| <i>Set template directory</i> | |
| patTemplate::setTags() | 24 |
| <i>Set template tags</i> | |
| patTemplate::setType() | 25 |
| <i>Set template type</i> | |
| patTemplate::readTemplatesFromFile() | 22 |
| <i>Parses several templates from one patTemplate file</i> | |
| patTemplate::parseTemplate() | 22 |
| <i>parses a template</i> | |
| patTemplate::exists() | 20 |
| <i>Check if a template exists</i> | |
| patTemplate::getAttribute() | 21 |
| <i>Gets an attribute of a template</i> | |
| patTemplate::getParsedTemplate() | 21 |
| <i>returns a parsed Template</i> | |
| patTemplate::getVar() | 22 |
| <i>get the value of a variable</i> | |
| patTEMPLATE_TYPE_SIMPLECONDITION | 13 |
| <i>Template type SimpleCondition</i> | |
| patTEMPLATE_TYPE_ODDEVEN | 13 |
| <i>Template type OddEven</i> | |
| patMessenger::\$vars | 5 |
| <i>Array, in dem die importierten Variablen gespeichert werden</i> | |
| patMessenger::\$varKeys | 5 |
| <i>Map, um Variablennamen in Integers zu übersetzen, da Shared Memory Funktionen nur Int Keys akzeptieren</i> | |
| patMessenger::process() | 5 |
| <i>dispatch Function, die alles weitere erledigt</i> | |
| <i>Ausser dieser Funktion müssen im Standardbetrieb keine weiteren Funktionen aufgerufen werden.</i> | |
| patSHMC.php | 8 |
| patEXCLUSIVE_LOCK | 8 |
| patMessenger::\$uid | 5 |
| <i>User ID des aktuellen Benutzers</i> | |
| patMessenger::\$tpl | 5 |
| <i>patTemplate Objekt für die Ausgabe</i> | |
| patMessenger::\$semKey | 4 |
| <i>Key der Semaphore zum Blocken des SHM Segments</i> | |
| patMessenger::\$neededVars | 4 |
| <i>Name der globalen Variablen, die importiert werden müssen</i> | |
| patMessenger::\$shm | 4 |
| <i>Schnittstelle zum Shared Memory Segment</i> | |
| patMessenger::\$shmKey | 5 |
| <i>Key des Shared Memory Segments</i> | |
| patMessenger::\$shmSize | 5 |
| <i>Größe des Shared Memory Segments in bytes</i> | |

| | |
|--|----|
| patRELEASE_LOCK | 8 |
| patSHARED_LOCK | 8 |
| patTemplate.php | 13 |
| patSHMC::remove_var() | 11 |
| <i>remove a variable from the shared memory</i> | |
| patTEMPLATE_TAG_END | 13 |
| <i>Variable suffix</i> | |
| patTEMPLATE_TAG_START | 13 |
| <i>Variable prefix</i> | |
| patTEMPLATE_TYPE_CONDITION | 13 |
| <i>Template type Condition</i> | |
| patSHMC::remove() | 11 |
| <i>remove (clear) shared memory</i> | |
| patSHMC::put_var() | 10 |
| <i>write a variable to the shared memory</i> | |
| patSHMC | 9 |
| <i>patSHMC - Shared Memory Connectivity</i> | |
| <i>This class can be used to easily handle shared memory including semaphore locking.</i> | |
| patSHMC::detach() | 9 |
| <i>disconnect from the shared memory</i> | |
| patSHMC::get_var() | 10 |
| <i>read a variable from the shared memory</i> | |
| patSHMC::lock() | 10 |
| <i>lock or unlock the shared memory</i> | |
| patMessenger | 4 |
| <i>Messenger Klasse zum Artikel "Interprozesskommunikation mit PHP4 und Shared Memory" im PHP Magazin 01/2002</i> | |
| <i>Diese Klasse soll lediglich zeigen, wie Shared Memory mit PHP4 zur Interprozesskommunikation genutzt werden kann.</i> | |

S

| | |
|--|---|
| signoff() | 3 |
| <i>Diese Funktion wird aufgerufen, wenn das Skript beendet wird.</i> | |