# Module

wbs:suite:modul:module

## Interne Struktur

Alle Module werden von der Klasse „core/modul/wbs_modul“ abgeleitet.

#### Eigenschaften

```
$name:string
$wbs:wbsfw
$modul_path
$content:array
$parameter:array
$is_loaded:boolean
$wildcards:array
$tables:array
$templates:array
$variables:array
```

#### Funktionen

```
__construct(name, path, wbs :
\de\blessen\wbsfw\wbsfw):de\blessen\wbsfw\core\modul\wbs_modul
db():de\blessen\wbsfw\core\db\myDB
wbs():de\blessen\wbsfw\wbsfw
getModulPath():string
loadFrontend():void
loadBackend():void
loadBackendFunctions():void
isLoaded():bool
handshake():void
createModulTables():void
setParameter(key : string, value : string):void
getParameter(key : string):mixed
ParameterExists(key : string):bool
setContent(key : string, value : string):void
getContent(key : string):mixed
hasContent(key : string):bool
getName():string
getTables()
loadTables()
getWildcards()
```

```
loadWildcards()
loadTemplates()
```

## Modul anlegen

### Kurzfassung

```
$modul_name = 'modul_name';
$modul_pfad = "/modul/modul_name/";
Create "/modul/modul_name/modul_name.php";
class modul_name extends core\modul\wbs_modul {}
[Create "/modul/modul_name/modul_name_constants.php";]
[Create "/modul/modul_name/woerterbuch.de.php";]
[Create "/modul/modul_name/woerterbuch.en.php";]
Create "/modul/modul_name/modul_name_loader.php";
```

### Festlegen eines Modulnamens

Der Modulname muß klein geschrieben sein und darf keine Leerzeichen enthalten.

```
$modul_name = 'modul_name';
```

### Entscheidung Internes Modul oder Modul

Interne Modul sind Systemmodule, die jeder Anwendung zur Verfügung stehen. Ein Beispiel ist install

Interne Module haben das Modulverzeichnis

```
$modul_pfad = "/modul_intern/modul_name/";
```

Alle anderen Module kommen in das Modulverzeichnis

```
$modul_pfad = "/modul/modul_name/";
```

### Modulklasse anlegen

Die Klasse wird mit dem selben Namen wie das Verzeichnis als Ableitung von

```
core\modul\wbs_modul
```

angelegt.

Beispiel: in der Datei

```
"/modul/modul_name/modul_name.php"
```

```
class modul_name extends core\modul\wbs_modul {}
```

**Basis Eigenschaften des Moduls:**

Das Modul wird mit seinem Namen, seinem Pfad $modul_pfad ( Siehe 1) und einer Instanz des wbsfw angelegt.

```
public function __construct($name,$path,wbsfw\wbsfw $wbs){}
```

Funktionen wbs_modul

```
__construct(name, path, wbs :
\de\blessen\wbsfw\wbsfw):de\blessen\wbsfw\core\modul\wbs_modul
db():de\blessen\wbsfw\core\db\myDB
wbs():de\blessen\wbsfw\wbsfw
getModulPath():string
loadFrontend():void
loadBackend():void
isLoaded():bool
handshake():void
setParameter(key : string, value : string):void
getParameter(key : string):mixed
ParameterExists(key : string):bool
setContent(key : string, value : string):void
getContent(key : string):mixed
hasContent(key : string):bool
getName():string
```

**Einbindung**

Handshake: Globale Module werden mit Komma getrennt in die Settings eingetragen, und dann geladen, dabei wird die Funktion handshake() aufgerufen, mit der zu einem sehr frühen Zeitpunkt in die Anwendung eingegriffen werden kann: (Z.B. Settings ändern)

$app_modules = explode(',', $pp['ta_module']); foreach ($app_modules as $modulname) {

```
  if ($wbs->modul()->debug) shout('App Module ' . $modulname);
  $modulname = trim($modulname);
  if (!$modulname) continue;
  $failure = $wbs->modul()->loadModul($modulname);
  if (($failure)) {
      shout('Could not load Modul ' . $modulname . ' ' . $failure);
  } else {
      $wbs->modul()->get($modulname)->handshake();
  }
```

}

Nach den globalen Modulen, werden die Module der Seite geladen, ansonsten werden die Module bei Einbindung geladen.

#### Laden der Module $wbs->modul()->loadModul()

Das Laden der Module geschieht in der Klasse modul()

Zuerst wird automatisiert der Pfad ermittelt: [modul,module_intern,module]

(Der Pfad module wird verschwinden, wenn die Module darin konvertiert wurden)

Gibt es Modulkonstanten, werden diese eingebunden, Konstanten müssen in der Datei „modul_name_constants.php“ liegen:

if(file_exists($modul_path.$modul_name.'_constants.php')) {

```
  require_once($modul_path.$modul_name.'_constants.php');
```

}

Gibt es ein Wörterbuch für die Übersetzung, wird es eingebunden:

switch ($this→wbs→getLanguage()) {

```
  case 'en':
      $woerter_buch = $modul_path . 'woerterbuch.en.php';
      break;
  case 'de':
  default:
      $woerter_buch = $modul_path . 'woerterbuch.de.php';
      break;
```

}

Die Datei $modul_name.'_loader.php' muß vorhanden sein, in ihr wird eine Instanz des Modul erzeugt, die dann in die Modulcollection eingehängt werden kann.

# Benutzung

Das CMS besteht aus Modulen. Für die Benutzung müssen sie in mehreren Stufen eingebunden werden.

##### Menü der Verwaltung

In der app_definition können Menüs oder Menüpunkte ein- und ausgeblendet werden.

#### Modul laden

In jeder angelegten Seite können die zu ladenden Module angegeben werden.

#### Modul einbinden

Ein Modul wird über Wildcards im Inhaltsbereich eingebunden.

## Wildcards

Wildcards sind Platzhalter, die durch festgelegte Inhalte im Inhaltsbereich ersetzt werden, um daraus die Seite aufzubauen. Sie haben folgende Syntax:

```
{{wbs}{module}MODUL}
{{wbs}{module}MODUL(param1=1, param2=1, ...)}
{{wbs}{module}MODUL[funktion]}
```

Dieser modulunabhängige Typ kann auch in Templates angewendet werden:

```
{{wbs}variable}
```

Beispiele:

```
{{wbs}{module}baumschule}
{{wbs}{module}cms(typ=menu, parent=0, level=1, depth=2)}
{{wbs}{module}baumschule[detail]}
{{wbs}title}
```