

```
<?php
$interpreterClass = "string: classname";

if (!class_exists("string: classname")) {
    class string: classname {
        var $title = "string: descriptive title";
        var $parsefile = "string: interpreter file name";
        var $version = "string: version";
        var $guid = "string: unique guid";
        var $ajaxcall = boolean: truefalse;
        var $phpvars = boolean: truefalse;
        var $bodyfunc = boolean: truefalse;
        var $metascript = boolean: truefalse;

        // get content title
        function getView($value = array()) {
            return string;
        }

        // get ajax function
        function getAJAX($value = array()) {
            return string;
        }

        // get ajax calls
        function getAJAXCall($value = array()) {
            return string;
        }

        // edit contents
        function getEdit($value = array()) {
            return string;
        }

        // return save data
        function getSave() {
            return string;
        }

        // parse module specific php variables
        function getHeader($value = array(), $mid) {
            return string;
        }

        // parse meta information
        function getMetaScript($value = array()) {
            return string;
        }

        // parse body tag function
        function getBodyFunction($value = array()) {
            return string;
        }

        // parse contents
        function getContent($value = array()) {
            return string;
        }

    } // class classname
} // if
?>
```

## WSP Interpreter Dokumentation - WSP 4.0 - Stand 2010-05-17

```
var $title = "string: descriptive title";
```

Die Variable `$title` gibt als String den Titel der Klasse zur Anzeige im Dropdown bei Auswahl des Interpreters sowie in der Listendarstellung bei der Auflistung aller Contentelemente zu einem Contentbereich zurück.

```
var $parsefile = "string: interpreter file name";
```

Die Variable `$parsefile` gibt als String den Namen der PHP-Datei zurück, die der Interpreter verwendet.

```
var $version = "string: version";
```

Die Variable `$version` gibt als String die Version des Interpreters zurück. Das Format der Versionierung ist A.B.C. Die Versionierung kann maximal 3 Versionsbereiche umfassen (z. B. 3.1.51, nicht jedoch 3.1.5.1), die Parameter B und C sind dabei optional. Bei der Installation kann nur ein gleich oder höher versioniertes Modul ein älteres überschreiben.

```
var $guid = "string: unique guid";
```

Die Variable `$guid` ist eine einmalige GUID, die den Interpreter identifiziert. Die GUID ist nach dem Prinzip 8-4-4-4-12 aufgebaut und darf nur Hexadezimalwerte enthalten. Die Kleinschreibung der Zeichen ist vorgeschrieben.

```
var $ajaxcall = boolean: true|false;
```

Die Variable `$ajaxcall` gibt zurück, ob der Interpreter die Funktionen `getAJAX()` und `getAJAXCall()` verwendet.

```
var $phpvars = boolean: true|false;
```

Die Variable `$phpvars` gibt zurück, ob der Interpreter die Funktion `getHeader()` verwendet.

```
var $bodyfunc = boolean: true|false;
```

Die Variable `$bodyfunc` gibt zurück, ob der Interpreter die Funktion `getBodyFunction()` verwendet.

```
var $metascript = boolean: true|false;
```

Die Variable `$metascript` gibt zurück, ob der Interpreter die Funktion `getMetaScript()` verwendet.

```
var $htmlmode deprecated
var $maxfields deprecated
var $classname deprecated
```

Die in einigen Interpretern (WSP≤3.5) verwendeten Variablen `$htmlmode`, `$maxfields` und `$classname` sollten nicht mehr verwendet werden.

### Funktion `getView()`

```
// get content title
function getView($value = array()) {
    return $value['desc'];
} // getView()
```

#### Beispielfunktion

`$value` enthält ein serialisiertes Array mit den Werten des Feldes `valuefields` aus der Tabelle `contents`

Grundsätzlich ist die Verarbeitung bzw. Rückgabe aller Werte aus `$value` möglich. Die Rückgabe muss als String erfolgen und sollte - aus optischen Gründen - nicht länger als 80 Zeichen sein.

return `string`

### Funktion *getAJAX()*

```
// parse contents
function getAJAX($value = array()) {
    ..
} // getAJAX()
```

*\$value* enthält ein serialisiertes Array mit den Werten des Feldes *valuefields* aus der Tabelle *contents*

Die Funktion *getAJAX()* kann AJAX-Funktionen enthalten, auf die beim Arbeiten im Interpreter zurückgegriffen werden kann. Die erforderliche Syntax ist entsprechend der XAJAX-Syntax.

return *string*

*Die Funktion getAJAX() steht noch nicht zur Verfügung.*

*Funktion getAJAXCall()*

```
// parse contents  
function getAJAX($value = array()) {  
    ..  
} // getAJAXCall()
```

*\$value* enthält ein serialisiertes Array mit den Werten des Feldes *valuefields* aus der Tabelle *contents*

Die Funktion *getAJAXCall()* kann JavaScript-Aufrufe zu den Funktionen aus *getAJAX()* enthalten, auf die beim Arbeiten im Interpreter zurückgegriffen werden kann. Die erforderliche Syntax ist entsprechend der XAJAX-Syntax.

return *string*

*Die Funktion getAJAXCall() steht noch nicht zur Verfügung.*

### Funktion *getEdit()*

```
// edit contents
function getEdit($value = array()) {
    ..
}
```

*\$value* enthält ein serialisiertes Array mit den Werten des Feldes *valuefields* aus der Tabelle *contents*

Zwischen den Beginn der Funktion und dem Aufruf der einleitenden Funktion zur Abbildung der Inhalte kann beliebiger PHP-Code eingefügt werden, der auf *\$value* zurückgreift.

```
..
ob_start();
?>
..
```

#### Einleitende Funktion zur Abbildung des Contents

Der Bereich nach *ob\_start()* kann für jegliche Inhalte (JavaScript, HTML, PHP) genutzt werden. Die Eröffnung des eigentlichen Eingabebereiches erfolgt mit:

```
..
<fieldset>
    <legend>Name</legend>
    ..

```

Optional kann im *legend*-Tag mit dem Aufruf der Funktion *showOpenerCloser(id, open|closed)* gearbeitet werden, die das entsprechende Fieldset offen oder geschlossen hält. Der zu öffnende bzw. zu schließende Bereich muss dann von einem *span*-Tag umlaufen werden.

```
..
<fieldset>
    <legend>Name <?php echo showOpenerCloser('xxxxxx','open'); ?></legend>
    <span id="xxxxxx">
    ..

```

Der Aufbau und die Gestaltung des Inhalts ist frei. WSP benutzt zur optimalen Einpassung ein 4er Raster mit je 25% Breite zur Darstellung des Inhalts.

```
..
<table border="0" width="100%" cellspacing="0" cellpadding="3">
    <tr>
        <td width="25%">Zelle 1</td>
        <td width="25%">Zelle 2</td>
        <td width="25%">Zelle 3</td>
        <td width="25%">Zelle 4</td>
    </tr>
</table>
..
```

Beispiel für einen WSP-orientierten Contentaufbau

## WSP Interpreter Dokumentation - WSP 4.0 - Stand 2010-05-17

Eingabe- und Rückgabewerte sind in folgender Struktur zu gestalten:

```
..  
<input type="text" name="field[xxx]" id="field_xxx" value="<?php echo $value['xxx']; ?>" />  
..
```

Dabei ist zur Kompatibilität auf die Benennung der Feldnamen auf `field[xxx]` zu beachten, da die Funktion `getSave()` mit den Daten aus dem Array `$_POST['field']` arbeitet.

Standardmäßig wird auf die hinterlegte Datenbankverbindung von WSP zurückgegriffen, so dass sie in einem Interpreter alle Tabellen der WSP-Datenbank ansprechen können. Sollten Sie eine andere Datenbankverbindung definiert haben, so wird diese Datenbankverbindung innerhalb des Interpreters verwendet, sie können dann ohne hart codierte Datenbankverbindungen nicht auf die Standard-Verbindung zurückgreifen.

Beendet wird die Funktion `getEdit()` durch den Aufruf der Übernahme-Funktion.

```
..  
<?php  
$edit = ob_get_contents();  
ob_end_clean();  
return $edit;  
} // getEdit()
```

return `string`

### *Funktion `getSave()`*

Die Funktion `getSave()` verarbeitet alle übergebenen Werte aus der Funktion `getEdit()`. Standardmäßig werden alle übergebenen POST-Variablen in einem Array erfasst und als serialisiertes Array die SQL-Anweisung zum Speichern in der Datenbank zurückgegeben.

```
// return save data
function getSave() {
    $value = array();
    foreach ($_POST['field'] as $key => $val):
        $value[$key] = $val;
    endforeach;
    return serialize($value);
} // getSave()
```

*Beispielfunktion für Standardübergaben*

Sollten Radio-, Checkbox- oder mehrere Auswahl-Werte übergeben werden, ist die Funktion entsprechend anzupassen.

return `string`

### Funktion `getHeader()`

```
// parse module specific php variables
function getHeader($value, $mid) {
    $parser = '';
    ..
}
```

`$value` enthält ein serialisiertes Array mit den Werten des Feldes `valuefields` aus der Tabelle `contents`

Die Funktion `getHeader()` gibt an den Parser zurück, was während des Veröffentlichens der Menüpunktes in die Datei geschrieben wird. Beispielwerte für Rückgaben sind zum Beispiel die normale Ausgabe der Werte aus der Datenbank:

```
$parser.= $value['xxx'];
```

Ebenfalls kann PHP-Code in der Ausgabe verwendet werden. Hier ist zu beachten, das der Code mit Slashes maskiert werden muss.

```
$parser.= "<?php include(\$_SERVER['DOCUMENT_ROOT'].\\"/\".$value['xxx'].\".php\"); ?>";
```

Beendet wird die Funktion `getHeader()` mit der Rückgabe der Parser-Werte:

```
..
return $parser;
} // getHeader()
```

return `string`

### Funktion `getMetaScript()`

```
// parse meta information
function getMetaScript($value) {
    $parser = '';
    ..
}
```

`$value` enthält ein serialisiertes Array mit den Werten des Feldes `valuefields` aus der Tabelle `contents`

Die Funktion `getMetaScript()` gibt an den Parser zurück, was während des Veröffentlichens der Menüpunktes in die Datei geschrieben wird. Beispielwerte für Rückgaben sind zum Beispiel die normale Ausgabe der Werte aus der Datenbank:

```
$parser.= $value['xxx'];
```

Ebenfalls kann PHP-Code in der Ausgabe verwendet werden. Hier ist zu beachten, das der Code mit Slashes maskiert werden muss.

```
$parser.= "<?php include(\$_SERVER['DOCUMENT_ROOT'].\\"/\".$value['xxx'].\".php\"); ?>";
```

Beendet wird die Funktion `getMetaScript()` mit der Rückgabe der Parser-Werte:

```
..
return $parser;
} // getMetaScript()
```

return `string`

### Funktion `getBodyFunction()`

```
// parse body tag function
function getBodyFunction($value) {
    $parser = '';
    ..
}
```

`$value` enthält ein serialisiertes Array mit den Werten des Feldes `valuefields` aus der Tabelle `contents`

Die Funktion `getBodyFunction()` gibt an den Parser zurück, was während des Veröffentlichens der Menüpunktes in die Datei geschrieben wird. Beispielwerte für Rückgaben sind zum Beispiel die normale Ausgabe der Werte aus der Datenbank:

```
$parser.= $value['xxx'];
```

Ebenfalls kann PHP-Code in der Ausgabe verwendet werden. Hier ist zu beachten, das der Code mit Slashes maskiert werden muss.

```
$parser.= "<?php include(\$_SERVER['DOCUMENT_ROOT'].\\"/\".$value['xxx'].\".php\"); ?>";
```

Beendet wird die Funktion `getBodyFunction()` mit der Rückgabe der Parser-Werte:

```
..
return $parser;
} // getBodyFunction()
```

return `string`

### Funktion `getContent()`

```
// parse contents
function getContent($value) {
    $parser = '';
    ..
}
```

`$value` enthält ein serialisiertes Array mit den Werten des Feldes `valuefields` aus der Tabelle `contents`

Die Funktion `getContent()` gibt an den Parser zurück, was während des Veröffentlichens der Menüpunktes in die Datei geschrieben wird. Beispielwerte für Rückgaben sind zum Beispiel die normale Ausgabe der Werte aus der Datenbank:

```
$parser.= $value['xxx'];
```

Ebenfalls kann PHP-Code in der Ausgabe verwendet werden. Hier ist zu beachten, das der Code mit Slashes maskiert werden muss.

```
$parser.= "<?php include(\$_SERVER['DOCUMENT_ROOT'].\\"/\".$value['xxx'].".php\"); ?>";
```

Beendet wird die Funktion `getContent()` mit der Rückgabe der Parser-Werte:

```
..
return $parser;
} // getContent()
```

return `string`