TemplateEngine2

Obexer Christoph

14. April 2013

Inhaltsverzeichnis

1 TemplateEngine2 Allgemein

1.1 TemplateEngine Version 1 Kompatibilität

Die syntax der TemplateEngine v1 wird vollständig verstanden - mit einer einzigen Außnahme:
Die {ELSE} Direktive muss nun als {IF:ELSE} geschrieben werden, Grund dafür ist das das {ELSE} als normale Template Variable verstanden werden kann und in Zukunft nicht mehr eindeutig sein wird(inline FOREACH).
Die methode `getTemplatePath` gibt set Version 2.0 nur noch den TemplatePath Anteil zurück, und nicht mehr RootPath / TemplatePath!

1.2 Features

- Plugin Interface - alles wird durch Plugins erledigt!
- Escape Method Interface - Plugins können auf Escape Methoden zurückgreifen um die Ausgabe zu filtern,...
- Ein paar Methoden um spezielle Aufgaben zu erleichtern(User Messages, addCSS, addJS, setPage-Title,...)
- Debugging Funktionen um Debug-Informationen in den Browser zu bekommen.
- Da alles als Plugin realisiert ist kann auch alles deaktiviert werden - auch die Skalar Ersetzung :)

2 TemplateEngine2 API

2.1 User API

2.1.1 Inst()

Die Inst Methode gibt ein Objekt der TemplateEngine zurück, das kann verwendet werden um die Tipparbeit zu verringern.

2.1.2 clear()

Mit der `clear` Methode werden alle gesetzten Variablen gelöscht und automatische Variablen wieder zurückgesetzt.

2.1.3 setBaseTemplatePath($path)

Setzt den Pfad des basis Templates. Relativ zum RootPath und zum Browser. Dieser Pfad wird verwendet wenn ein Template eine .tpl-Datei verwenden will die nicht existiert, die TemplateEngine versucht dann die .tpl-Datei im Basis-Template zu finden. Seit Version 2.1.0.

2.1.4 setTemplatePath($path)

Setzt den Pfad in dem nach Templates gesucht werden soll. Relativ zum RootPath und zum Browser. Der TemplatePath steht Templates als {TEMPLATE_PATH} zur Verfügung.

2.1.5 getTemplatePath()

Gibt den zuvor mit setTemplatePath gesetzten Teil zurück, die Version 1.0 hat hier den Pfad mit dem RootPath Anteil geliefert!

2.1.6 setRootPath($path)

Setzt den Pfad der dem Browser als RootPath zur Applikation dient, diese Variable ist als {ROOT_PATH} in den Templates verfügbar.

2.1.7 getRootPath()

Gibt den zuvor mit setRootPath gesetzten Pfad zurück.

2.1.8 output($basetemplate, $havingSession)

Verarbeitet das Template mit dem in $basetemplate übergebenen Namen (gesucht wird es im TemplatePath) und schickt das Ergebnis zum Browser. Der $havingSession Parameter gibt an ob vor der Verarbeitung nur TE_static_setup oder auch TE_setup aufgerufen werden soll.
$havingSession = true -> TE_static_setup, TE_setup aufrufen.
$havingSession = false -> TE_static_setup aufrufen.

2.1.9 processTemplate($basetemplate, $havingSession)

Verarbeitet das Template mit dem in $basetemplate übergebenen Namen (gesucht wird es im TemplatePath) und gibt den resultierenden Content als String zurück. Der $havingSession Parameter gibt an ob vor der Verarbeitung nur TE_static_setup oder auch TE_setup aufgerufen werden soll.
$havingSession = true -> TE_static_setup, TE_setup aufrufen.
$havingSession = false -> TE_static_setup aufrufen.

2.1.10 set($name, $value)

Setzt die Variable mit dem Namen $name auf den Wert $value, die Eingebauten Direktiven verstehen hier nur Skalare und einfache Arrays. Die Erweiterbarkeit der TemplateEngine2 macht es hier allerdings auch möglich beliebige Datenstrukturen zu verwalten und zu verwenden.

2.1.11 get($name, $default = null)

Ermittelt den Wert einer Variablen, sollte der Wert noch nicht bekannt sein wird der wert des 2. Parameters($default) zurückgegeben. Diese Methode arbeitet nur auf Basis des globalen Kontext und sollte deswegen nicht in einem Template-Plugin verwendet werden, dafür wurde lookupVar implementiert.

2.1.12 delete($name)

Löscht den wert der für die Variable $name gespeichert wurde.

2.1.13 Error($error)

Diese Methode fügt dem standardmäßig vorhandenem Array TE_ERRORS eine Fehlermeldung hinzu. Diese Fehlermeldungen sind dazu gedacht dem User angezeigt zu werden.

2.1.14 Warning($warning)

Diese Methode fügt dem standardmäßig vorhandenem Array TE_WARNINGS eine Warnmeldung hinzu. Diese Warnmeldungen sind dazu gedacht dem User angezeigt zu werden.

2.1.15 Info($info)

Diese Methode fügt dem standardmäßig vorhandenem Array TE_INFOS eine Infomeldung hinzu. Diese Infomeldungen sind dazu gedacht dem User angezeigt zu werden.

2.1.16 setTitle($title)

Diese Methode setzt den Titel der Seite (genauer gesagt die Variable `PAGE_TITLE`).

2.1.17 header($html)

Mit `header` wird der übergebene HTML-Code an den HTML-Head angehängt.

2.1.18 addCSS($css)

Fügt dem HTML-Head ein `link` Tag hinzu der als `href` den übergebenen String hat.

2.1.19 addJS($js)

Fügt dem HTML-Head ein `script type='text/javascript'` Tag hinzu der als `src` den übergebenen String hat.

2.1.20 setFileDebugMode($mode)

Setzt man `$mode` auf `true` so wird jeder geladenen Datei mit HTML-Kommentaren der Dateiname vorne und hinten an den Inhalt angehängt.

2.1.21 setForceTplExtension($mode)

Setzt man `$mode` auf `true` so wird jede Datei, die nicht `.tpl` als Dateityp hat zurückgewiesen.

2.1.22 setJailToTemplatePath($mode)

Setzt man `$mode` auf `true` so wird jede Datei, die nicht im `TEMPLATE_PATH` gefunden wird zurückgewiesen.

2.1.23 forceMode($mode)

Setzt den Debug-level auf den übergebenen `$mode` und verhindert alle weiteren änderungen durch `setMode` oder {`LOGLEVEL=...`}. Gültig sind alle werte von `TEMode`, standardmäßig ist `TEMode :: error` gesetzt.

2.1.24 setMode($mode)

Setzt den Debug-level auf den übergebenen `$mode` nur dann wenn das nicht durch `forceMode` verhindert wurde. Dieser Wert kann auch durch `forceMode` und {`LOGLEVEL=...`} geändert werden. Gültig sind alle werte von `TEMode`, standardmäßig ist `TEMode :: error` gesetzt.

2.2 Plugin API

2.2.1 pushContext($templateString, array $context)

Damit wird ein neuer Context auf den Context-Stack geschoben und verarbeitet. `$templateString` ist der zu verarbeitende Template String, das Array `$context` anthält alle im Kontext verfügbaren Variablen.

2.2.2 escape($escaper, $value)

Mit dieser Methode wird die Escape methode mit dem in `$escaper` übergebenen Namen ausgeführt und das Ergebnis des escapens von `$value` zurückgegeben.

2.2.3 registerPlugin($plugin, $regexp, $callback)

Diese Methode registriert ein Plugin für die Verwendung in Templates.

`$name` der Name des Plugins - muss eindeutig sein, ansonsten wird das zuvor registrierte Plugin überschrieben. Namen mit `TE_` am Anfang sind reserviert!

`$regexp` der Reguläre Ausdruck mit dem nach Vorkommen für dieses Plugin gesucht wird.

`$callback` die Funktion die mit Treffern für das Plugin aufgerufen wird - bei jedem Aufruf wird nur ein Treffer verarbeitet.

Das Callback Interface:
Die Callback Funktion erhält zwei Parameter:

`$context` ein Array in dem alle Werte des Aktuellen Kontext entahlten sind. Wird der angeforderte Wert im aktuellen Context nicht gefunden, dann kann `lookupVar` verwendet werden um im Context-Stack nach Unten zu suchen.

`$match` der Parameter der von `preg_replace_callback` an die Interne Callback Funktion übergeben wurde, sie enthält alle Matches des Regulären Ausdrucks - so wie er registriert wurde.

Der return-Wert der Funktion muss entweder ein String oder ein Boolean(`false`) sein. Mit `false` lehnt das Plugin den Treffer ab und er wird unverändert belassen, ansonsten wird der Treffer mit dem zurückgegebenen String ersetzt.

2.2.4 unregisterPlugin($plugin)

Hebt die Registrierung eines Plugins auf - es kann anschließend nicht mehr verwendet werden.

2.2.5 lookupVar($name, &$value)

`lookupVar` findet den Wert der in `$name` übergebenen Variable und gibt `true` zurück wenn der Wert auch gefunden wurde, ansonsten wird `false` zurüchgegeben und der Wert von `$value` bleibt unverändert. Diese Methode sollte nur dann aufgerufen werden wenn die Variable im Kontext der dem Plugin Callback übergeben wurde nicht enthalten ist.

2.2.6 getFile($name, &$content)

`getFile` lädt eine Datei und gibt bei Erfolg (return `true`) in ihrem 2. Parameter den Inhalt zurück. Diese Funktion beachtet dabei `$force_tpl_extension`, `$jail_to_template_path` und `$debug_files`.

2.3 Escape Method API

2.3.1 registerEscapeMethod($method, $callback)

Mit dieser Methode wird die Escape Methode mit dem Namen `$method` für die Verwendung registriert.

Das Callback Interface:
Die Callback Funktion erhält 2 Parameter:

`$value` der Rohwert der für die zu escapende Variable gesetzt wurde - was das ist hängt außschließlich davon ab was für die Variable gesetzt wurde - beispielsweise ein String, ein Boolean, ein Objekt oder ein Array,...

`$config` wurde für die Escape Methode eine Konfiguration(beispielsweise eine Zeitzohne, oder Sprache,...) mit `setEscapeMethodConfig` hinterlegt wird diese hier übergeben, andernfalls ist dieser Parameter `null`.

Der return Wert der Funktion sollte ein String sein. Der Wert wird dann zum auslöser der Escape methode gegeben.

2.3.2 unregisterEscapeMethod($method)

Löscht eine registrierte Escape Methode wieder.

2.3.3 setEscapeMethodConfig($method, $config)

Kann beliebige Daten als Konfiguration für eine Escape Methode speichern.

2.3.4 getEscapeMethodConfig($method

Liefert die aktuell gespeicherte Konfiguration für die Escape Methode.

2.4 Misc API

2.4.1 LogMsg($msg, $success = true, $mode = TEMode :: debug, $finished = true)

Mit der LogMsg Methode kann Debug Information aufgezeichnet werden, die Informationen werden nur aufgezeichnet, wenn der übergebene `$mode` größer oder gleich dem Eingestellten ist. Mit `$finished = false` wird die Nachricht erst mit dem nächsten Aufruf(`$finished = true`) an den Puffer angehängt, dieser kann auch ein neues Log-Level für die gesamte Nachricht angeben. `$success` gibt an ob der Vorgang erfolgreich war.

2.4.2 captureTime($milestone)

`captureTime` kann dazu verwendet werden Timing-Probleme beim Aufbau einer Seite auf die Schliche zu kommen. Der `$milestone` Parameter ist einfach ein Name für die Position im Ablauf des Seitenaufbaus, beispielsweise `startTE` für den Start der Template Verarbeitung oder `stopTE` für das Ende der Template Verarbeitung. `printTimingStatistics` wird automatisch bei scriptbeendung aufgerufen und gibt die aufgezeicheten Milestones aus und markiert damit das Ende der Skriptausführung (da `TEincluded` den Anfang markiert kann aus dem Time Offset von `printTimingStatistics` die Skriptlaufzeit abgelesen werden).

3 TemplateEngine Plugins

3.1 Skalar Ersetzung

Die Skalar Ersetzung ersetzt alle Vorkommen von {`VAR`} mit dem Wert der der TemplateEngine für `VAR` bekannt gemacht wurde(siehe `set`). Die Skalar Ersetzung verwendet auch Escape Methoden, so kann beispielsweise die Anzahl der Elemente eines Arrays names `ARMIES` mit {`ARMIES|LEN`} Ausgegeben werden. Vorraussetzung für die Verwendung von Escape Methoden ist das diese Registriert sind - gilt auch für `LEN`.

3.2 LOAD

Mit der `LOAD` Direktive wird an der Stelle des Vorkommens ein anderes Template geladen. Verwendung: {`LOAD=path/to/file.tpl`} der Pfad ist relativ zum **TemplatePath**.

3.3 LOAD_WITHID

Mit der `LOAD_WITHID` Direktive wird an der Stelle des Vorkommens ein anderes Template geladen. zusätzlich zur normalen `LOAD` wird innerhalb der geladenen Datei {`LOAD:ID`} durch den angegebenen Wert ersetzt. Verwendung: {`LOAD_WITHID=path/to/file.tpl;the-new-id`} der Pfad ist relativ zum **TemplatePath**.

3.4 LOGLEVEL

Setzt das Log-Level der TemplateEngine2 dynamisch auf den gegebenen Wert. Verwendung: {LOGLEVEL=ERROR}. Es sind folgende Werte erlaubt:

- DEBUG
- WARNING
- ERROR
- NONE

3.5 FOREACH

FOREACH wird dazu verwendet Arrays mit einfachem Aufbau darzustellen, gegeben sei ein Array mit dem Namen USERS mit folgendem Aufbau:
[[NAME=Mea, LEVEL=admin], [NAME=Schalk, LEVEL=admin], [NAME=cobexer, LEVEL=admin]]
bei der Ausführung wird für jedes Element im Array ein eingener Context erzeugt, das hat einerseits einen gewissen Geschwindigkeitsvorteil andererseits aber auch den Vorteil, daß alle Plugins in diesem Kontext ganz normal arbeiten können, weitere FOREACH oder IF oder jedes andere Plugin.
Zusaẗzlich ist im neuen Context eine variable namens {ODDROW} verfügbar die entweder odd oder leer ist je nachdem ob die aktuelle Zeile ungerade ist oder nicht.
Der Index der foreach Iteration wir mit dem speziellen Token {FOREACH:INDEX} bereitgestellt.
Verwendung:

```
<ul>
{FOREACH[USERS]=userlist.tpl}
</ul>
```

Inhalt von userlist.tpl:

```
<li>{NAME} ({LEVEL})</li>
```

Ergebnis:

```
<ul>
<li>Mea (admin)</li>
<li>Schalk (admin)</li>
<li>cobexer (admin)</li>
</ul>
```

3.6 SELECT

Die SELECT Direktive dient dazu HTML-Select Elemente zu generieren. Verwendung:

```
<select name="myselect">
{SELECT=MYOPTIONS}
</select>
```

Wobei MYOPTIONS ein Array mit NAME / VALUE Paaren ist:
[[NAME=Mea, VALUE=1], [NAME=Schalk, VALUE=2]]
Ergebnis:

```
<select name="myselect">
<option value="1">Mea</option>
<option value="2">Schalk</option>
</select>
```

3.7 IF

Die`IF` Direktive wird dazu verwendet eine Bedingung auszuwerten. Aufbau der `IF` Direktive:
{`IF(VAR_NAME[|ESCAPE_METHOD] operator wert)`}
ist die Bedingung wahr bleibt dieser Teil erhalten
{`IF:ELSE`}
ansonsten dieser
{`/IF`}
für den Operator kan einer der Folgenden verwendet werden:

- gt (>)
- lt (<)
- eq (==)
- gte (>=)
- lte (<=)
- ne (!=)

Der Wert `'null'` hat die besondere Bedeutung von `null` in PHP, er kann mit `ne` dazu verwendet werden zu prüfen ob eine Variable gesetzt ist.

4 TemplateEngine Escape Methods

Escape Methoden werde dazu verwendet Template Variablen zu verändern, beispielsweise um zu verhindern das User HTML- oder Template-Code einschleusen können.

4.1 LEN

Die Escape Methode `LEN` kann nur auf Strings und Arrays angewendet werden, das Ergebnis der Escape Operation ist entweder die Stringlänge oder die Anzahl der Elemente im Array.
Anwendungsbeispiel:
`Anzahl der Spieler:` {`USER_ARRAY|LEN`}

5 spezielle URL Parameter

force_debug setzt den TemplateEngine Mode auf Debug, es werden alle {`LOGLEVEL=...`} und `setMode` aufrufe ignoriert.

show_timing listet alle „Milestones“ die mit `captureTime` aufgezeichent wurden auf.

debug_files hängt allen eingebundenen Dateien einen HTML-Kommentar mit dem Dateinamen vorne und hinten an.

no_inline sucht nach allen inline `style` attributen und entfernt sie.

te_dump gibt nach dem verarbeiten der Seite alle Template-Variablen und deren Werte aus.

force_def_err_handler FIXME

force_def_exception_handler FIXME

6 GomBG spezifische Features

Für GomBG sind einige extra Plugins vorhanden:

6.1 TemplateEngine Plugins

6.1.1 LINKTO

Mit `LINKTO` werden HTML-Links zu verschiedenen 'Objekten' in GomBG erstellt.
{`LINKTO=WHAT;TYPE=TP;TEXT=TX;TITLE=TI`}
Wobei man `WHAT, TP, TX` und `TI` entsprechend ersetzen muss.

LINKTO= der einzusetzende Wert ist beispielsweise der Nick, Allianzname, Uid, Pos,...

TYPE= Der `TYPE` ist zur Zeit einer der folgenden 7 Werte:

PLAYER (Nick) Link zum Profil.

SENDMSG (Nick) Link zum Nachrichtenmodul mit dem Nick als Empfänger.

ALLI (Allianzname) Link zur Allianzseite.

MAP (Pos) Link zum Feld auf der Map ([\d{5}] im `TEXT` wird durch [XX / YY / Z] ersetzt).

ARMY (Army ID) Link zum bearbeiten einer Armee.

BUILDING (Building ID) Link zur Beschreibungsseite eines Gebäudes mit allen Stufen, Abhängigkeiten und so(gibt es noch nicht).

FAV (Nick) Link zum NAchrichtenmodul um einen User als Buddy zu speichern.

TEXT= Dieser Text wird dem User angezeigt(<a ...>TEXT)

TITLE= Das ist der Titel der in einem Tool-Tip angezeigt wird wenn man mit dem Cursor auf einen Link zeigt.

6.2 TemplateEngine Escape Methods

6.2.1 TIMESTAMP

Die `TIMESTAMP` escape Methode formatiert einen als Integer gesetzten Datumswert in dem default Format für User. Beispiel:
{`FINISHED_AT|TIMESTAMP`}

6.3 spezielle URL Parameter

show_queries zeigt eine Liste mit allen Datenbak Queries, mehrfach ausgeführten Queries und der Ausführungszeit, das Passwort ist am ende der `inc/DB.php` zu finden.