OLIV

Einleitung

Oliv ist ein CMS und zeichnet sich dadurch aus, das es auf die Nutzung von vielen Sprachvarianten ausgelegt ist. Basis aller Komponenten ist eine konsequente Verwendung von Sprachdateien, die für beliebig viele Sprachen angelegt werden können. Besonders die angewandte Struktur bei statischen Seitentexten bringt bei dieser Methode eine Vereinfachung von Übersetzungen, da nur Textteile, und nicht ganze Seiten für mehrere Sprachen angelegt werden müssen.

Ein intelligentes Management setzt vorhandene Textteile automatisch in der gewählten Sprache ein und nutzt für fehlende Übersetzungen die Standardsprache des Seitentextes. Neben der gewählten Standardsprache der gesamten Internet Präsentation kann auch jede einzelne Seite eine eigene Standardsprache besitzen. Im Editiermodus werden diese fehlenden Passagen farblich hinterlegt und damit die Arbeit erleichtert.

Das Basissystem verzichtet auf die Nutzung einer Datenbank, sondern benötigt für den Betrieb nur PHP ab Version 5. Jegliche Informationen, wie Konfiguration, Templates, Sprachdateien und Inhalt wird in XML Dateien abgelegt. Damit ist es jederzeit möglich die gesamte Seite mit einem einfachen Texteditor zu bearbeiten. Ein Backup des Root Verzeichnisses erfasst so auch sämtliche Inhalte.

Das System bietet über eine Schnittstelle für Module ein einfaches System der Erweiterung. Für diese Module werden Funktionen zur Verfügung gestellt, um die Forderung der Multilingualität auch hier umsetzen zu können.

Seitenaufbau

Die Seite wird durch drei Elemente definiert.

- Template
- CSS
- Content

Template

Die grundlegende, hierarchische Anordnung der Element zueinander wird im Seiten-Template festgelegt. Die Formarierung dieser Elemente kommt aus dem entsprechenden CSS File. Grafiken werden als Style-Attribut im jeweiligen Tag des Templates angegeben, um die Grafik-Klassen von OLIV nutzen zu können.

Als Tag des Bereiches sind gültige Html Tags, wie oder möglich. Der Bereichs-Id wird als

Id-Attribut übergeben. Es ist auch möglich Tabellen zu definieren. Mit den Variablen @row und @colom können während des Rendeings die jeweiligen Werte zugewiesen werden, z.B. für durchlaufende Ids.

Den Template Tags können Attribute übergeben werden:

- **style**: Style String, der diesem ID beim Rendering zugewiesen wird. Hier kann die Grafik Klasse zum Einfügen von Bildern genutzt werden.
- **link**: Macht den gesamten Bereich zu einem Hyperlink. Die Links werden über den Router definiert.
- **title**: Gibt einen Titel an, der beim Überfahren des Links im Browser angezeigt wird (vorausgesetzt der Browser unterstützt dies).
- **script** (nur in Modulen nutzbar): Name des Scriptes, das an dieser Stelle ausgeführt wird. Diese Funktion steht nur in Modulen zur Verfügung, wenn keine Content Datei verwendet wird.

Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<template name='templateName'>
  <div id='divName>
    <div id='subDivName' script='function'/>
  </div>
  <div id='linkedDivName' link='linkName' title='linkTitle'/>
</template>
```

Die Template Klasse kann auch für die Darstellung innerhalb eines Moduls genutzt werden. Dabei übernimmt die Teplate Klasse das Rendering und den Aufruf der nötigen Scripte, die für die Inhaltsdarstellung zuständig sind.

CSS

Content

Der Inhalt der so festgelegten Bereiche wird im Content angegeben. Dabei wird ein Modul mit dem Anzeigebereich verlinkt und die wichtigsten Parameter übergeben.

Der eigentliche Seiteninhalt wird im Content-Verzeichnis angegeben. In content.xml werden alle Seiten registriert.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<content name='index' land='de-DE'>
```

```
  <masterpage>masterPageName</masterpage>

  <div id='element_id' script='moduleName'>
    value
  </div>
</content>
```

Mit der Masterpage Angabe wird eine andere Content-Definition an dieser Stelle eingefügt. So lassen sich Seiten aus komplexen Kompinenteninhalten zusammenstellen. Masterpages können auch verschachtelt werden.

Value kann auch aus einem XML bestehen. Spezielle Tags rufen bestimmte Funkionen auf:

- **<text>** rendert den darin enthaltenen Text mit allen Html Tags. Dies ist die Hauptfunktion für statische Texte innerhalb des CMS. Für alle Textteile auf einer Seite gilt, wie im gesamten System, die Unterstützung von multilingalen Texten. Die Seite wird bei der Erzeugung einer bestimmten Grundsprache zugeordnet. Dies definiert nicht nur die Sprache des Ursprungsautors, es gibt auch die Strukturierung des Textes vor. Diese Struktur, das sind Anzahl und Abfolge von Überschriften, Absätzen, Bildern, etc., kann nur vom Ursprungsautor verändert werden. Die wird klar, wenn man sich die Art der Textverwaltung ansieht

Multilingualität

Textverwaltung

In der Content Definition sind zwischen den Format Tags nur Lang-Code-Ids eingefügt. Die eigentlichen Textstellen befinden sich in der Lang-Datei. Wenn ein Tag entfernt wird, dann werden alle Übersetzungen dieser Textstelle beeinflusst und nicht mehr angezeigt. Ein Messaging System ermöglicht eine Benachrichtigung der Autoren von Übersetzungen von den Änderungen. Der kann dann die Textpassagen anpassen und nicht mehr angezeigte Passagen löschen.

Dafür muss bei einer Übersetzung in eine neue Sprache nur der Text, und nicht das Grundgerüst angelegt werden. Bilder, Links etc. bleiben einmalig in der Content Datei definert.

Editor

Als Editor für Texte kommt der TnyMCE Javascript Editor zum Einsatz. Je nach Bereich wird er unterschiedlich genutzt.

Bei der Neuerstellung von Texten kommt die volle Funktionalität zur Formatierung zum Einsatz. Wird ein Text in der Usprungssprache editiert, ist es ebenso. Bei der Editierung einer anderen, als der Ursprungssprache können dagegen nur die Inhalte der einzelnen Tags editiert werden und nicht der gesamte Text. Die Ursprungssprache wird in der content-Datei mit dem default-lang='lang-code' Attribut des content-Tag definiert.

Bein Abspeichern wird der Text analysiert und der Inhalt der einzelnen Tags in der Sprachdatei abgelegt. Als Textcode zur Referenz wird das Format PAGENAME_TEXTxxx verwendet, wobei die Textstellen durchnummeriert werden.

Im Fall einer Fremdsprache entspricht dies der Aktualisierung der Inhalte, ohne den Textcode zu verändern. Wird der Ursprungstext verändert und ein Textteil eingefügt oder entfernt, hat dies auch Auswirkungen auf die Übersetzungen. Bei einem Löschvorgang wird der Autor darauf hingewiesen, dass es auch in den anderen Sprachen zu einer Löschung kommt. Über die Autoren ID wird der Übersetzer über die Veränderungen in Kenntnis gesetzt. Die endgültige Löschung erfolgt dann durch den Übersetzer, der Textteil ist aber nicht mehr sichtbar. Wird ein Teil eingefügt, so erscheint in den Übersetzungen dieser Teil, farbig hervorgehoben und mit einem Hinweis versehen, in der Ursprungssprache.

Multilinguale Grafik

Auch bei der Grafikklasse wird die Mehrsprachigkeit unterstützt. Das kann dort genutzt werden, wo Grrafiken mit eingebundenen Texten zum Einsatz kommen.

Bilder werden in den Images Verzeichnissen abgelegt. Innerhalb dieser können Sprach-Unterverzeichnisse angelegt werden, die dann die verschiedenen Sprachversionen beinhalten. Die Syntax ist so wie bei den Sprachdateien:

```
lang-code.graficFileName.extension
```

Normale, textfreie Grafiken werden direkt im Image Verzeichnis abgelegt.

Bildaufruf

Der Bildaufruf erfolgt mit dem Filenamen.extension. Es wird von der Image-Klasse zuerst eine Sprachversion der aktuellen Sprache, wenn diese Datei nicht existiert die Default-Sprachversion gesucht. Erst danach wird die reine Grafik im Images Verzeichnis genutzt.

Bei der Grafiksuche wird zuerst das Systemverzeichnis unter /Images durchsucht und danach die Image Verzeichnisse aller registrierten Module.

URL-Verarbeitung

Der Router teilt sich in zwei Bereiche: den Router und den Generator.

Router

Der Router übernimmt die Parameter aus der argv Systemvariable und stellt die Verbindung zur korrekten Seite her. Dabei greift der Router auf die Informationen in der content.xml zu. Er nutzt das Attribut name='pageName_ID' zur Übersetzung der Url. Die content.xml wird aktualisiert wenn die Url nicht rückübersetzt werden kann. In dem Fall werden alle Seitendefinitionen nach den jeweiligen

Spracheinträgen durchsucht.

content.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<content>
<!-- referenz by page id -->
  <id>
    <pageName_ID>
      <de-DE>germanName</de-DE>
      <en-GB>englishName</en-GB>
    </pageName_ID>
  </id>

<!-- referenz by page name -->
  <name>
    <germanName id='de-DE'>pageName_ID</germanName>
    <englishName id='en-GB'>pageName_ID</englishName>
  </name>
</content>
```

Damit ist es dem Router möglich die jeweiligen muktilingualen Seitentitel in der lesbaren Url richtig zuzuordnen. Durch die Rückübersetzung des Routers ist es unerheblich in welcher Sprache die Url aufgebaut und welche Sprache angezeigt werden soll.

Generator

Der Generator stellt das Werkzeug zur Verfügung, um die richtigen Aufrufparameter (lesbare Url) in der jeweils aktuellen Sprache zu erzeugen. Für den Seitennamen wird auch hier auf die content.xml zugegriffen.

Für die Benutzerfreundlichkeit werden die Urls mit Parametern in eine Verzeichnisstruktur gewandelt. Dies geschieht durch den Generator.

Es werden drei Parameter direkt unterstützt:

- **Sprache**: Das erste virtuelle Unterverzeichnis ist die Sprache. Es wird der Sprach-Landescode genutzt. Dieser Parameter wird immer übergeben.
- **Page**: Der zweite Parameter ist die aufgerufene Seite. Hier wir die multilinguale Übersetzung des Routers genutzt, um die virtuellen Verzeichnisse in der jeweiligen Sprache zu ermöglichen.
- **Value**: Der dritte Parameter enthällt alle anderen Werte, die vom jeweiligen Modul selbst interpretiert werden müssen.

Beispiel:

Aufruf der Startseite in Deutsch und Englisch

```
www.meineseite.de/de-DE/Startseite/
www.meineseite.de/en-GB/Home/
```

Aufruf einer Unterseite Anfahtsplan von Kontakt

```
www.meineseite.de/de-DE/Kontakt/Anfahrt/
```

Module

OLIVS kann in seiner Funktionalität einfach erweitert werden, indem weitere Module geladen werden. Die Module werden im Unterverzeichnis IG_MODULE_PATH abgelegt und müssen eine bestimmte Grundstruktur erfüllen. Der Verzeichnisname muss gleich dem Modulnamen gewählt werden.

Filestruktur

Im Unterverzeichnis des Moduls müssen zwei Dateien existieren

```
[modulname]
  [templateDirectory]
  [languageDirectory]
  define.xml
  modulScript.php
  [searchScript.php]
```

Die XML-Datei define.xml enthält die nötigen Definitionen für den Aufruf des Moduls. Die Datei muss folgende Angaben enthalten:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<module>
  <name>moduleName</name>
  <version>version</version>
  <author>Author_name</author>
  <copyright>Copyright</copyright>

  <script>
    <main>moduleScript.php</main>
    [<search>searchScript.php</search>]
    <template>templateDirectory/</template>
    <language>languageDirectory/</language>
    <default_language>defaultLanguageCode</default_language>
  </script>
</module>
```

Im Unterbereich <script> sind die nötigen Informationen zum Aufruf des Moduls abgelegt. Dabei werden folgende Informationen übergeben:

- **moduleScript.php** Name des Hauptscripts
- **templateDirectory/** Name des Unterverzeichnises für Templates
- **languageDirectory/** Name des Unterverzeichnises für Language-files
- **defaultLanguageCode** Language-code für die Defaultsprache des Moduls

(kann von der Seiten-Defaultsprache abweichen)

- **searchScript.php** Name des Suchscripts, wenn das Modul eine Suchfunktion anbietet.

Weitere Unterverzeichnisse und Scripte können bei bedarf angelegt werden.

Einbinden

Die Module werden während der Core-Initialisierung im System registriert. Im Preprozessor wird der Seiteninhalt analysiert und die nötigen Modul-Klassen geladen. Ein Modul kann auf einer Seite dadurch mehrfach genutzt werden (z.B. ein Menü-Modul).

Während des Rendering-Prozesses werden Instanzen der benötigten Klassen für jeden Anzeigebereich erstellt. Der Start des Moduls muss also im Constructor erfolgen. Als Parameter wird das Header-Object übergeben.

Header

Modul Templates

Templates sind das Mittel zur Seitengestaltung. Templates werden für den grundlegenden Seitenaufbau, aber auch für den Aufbau der Module genutzt. Dabei greifen alle Funktionen auf die gleichen Klassen zu. Durch die globale Definition des benutzten Templates kann das Seitenlayout einfach umgestellt werden. Die Syntax ist gleich den allgemeinen Seiten-Templates.

Bei der Seiteninitialisierung wird der Name der globalen Templates festgelegt und in IG_TEMPLATE allgemein zugänglich gemacht. Im Standardfall ist der Wert default. Das Unterverzeichnis default muss daher in allen Modulen die Templates nutzen, vorhanden sein, da es auch als Rückfallebene dient, sollte ein anderes Template in einem Modul nicht realisiert sein.

Verzeichnisstruktur

Modul-Templates müssen in einer bestimmten Seitenstruktur untergebracht sein.

```
[modules]
  [modulName]
    [template]
      [globalTemplateName]
        [images] {optional}
        default.xml
        localTemplateName.xml
        default.css
```

Das Grundverzeichnis für die Templates (hier template) wird im Modul-Script-Header abgelegt und in ist in der jeweiligen define.xml des Moduls definiert.

Unterhalb des Grundverzeichnises muss für jedes Template-Design ein dementsprechendes Verzeichnis mit dessen Name angelegt werden. Für die Grundeinstellung ist dies default.

Unterhalb sind die allgemeinen Template-Definitionen in default.xml abgelegt. Die allgemeinen CSS-Klassen finden sich in default.css.

Werden vom Template Bilder genutzt, so sind diese im images Verzeichnis abgelegt.

Sub-Templates

Innerhalb des templateName Verzeichnisses können neben dem Default-Template noch weitere Template-Varianten angelegt werden. Diese werden mit dem <template>templateName</template> Tag in der Contentdefinition angegeben.

```
<ig_header mod='header'>
  <template>localTemplateName</template>
</ig_header>
```

In diesem Fall wird für das Modul header statt default.xml das Sub-Template localTemplateName.xml genutzt. In jedem Fall aber im Haupt-Template-Verzeichnis globalTemplateName. Durch diese Technik ist es möglich, z.B. für unterschiedliche Seiten das Layout des Headers etwas anzupassen. Für jede globale Templateänderung werden dann die entsprechenden Sub-Templates aus dem globalen Template Verzeichnis genutzt.

Sollte beim Aufruf eines Subtemplates, oder auch eines speziellen globalen Templates, dieses nicht gefunden werden, so wird das Default-Template statt dessen genutzt.

Stylesheets

Neben der Designberschreibung durch das Template-XML kann für jedes Template oder Sub-Template auch eine eigene CSS-Klassenbeschreibung angegeben werden. Diese muss sich im gleichen Verzeichnis befinden und den gleichen Namen wie die entsprechende Template-Datei besitzten, lediglich mit der Endung .css.

Das Stylesheet wird beim Laden des Templates automatisch mit der Seite verlinkt.