Analyse der OntoWiki-Struktur

Inhaltsverzeichnis

1 Strukturierung des OntoWikis mittels Model-View-Controller

1.1 Allgemeines zu MVC

Mit Model-View-Controller (kurz: MVC) wird ein Design Pattern bezeichnet, welches zur Strukturierung von Software eingesetzt wird. Kern ist die Dreiteilung von Software in ein Datenmodell, in eine Präsentationseinheit, sowie eine Programmsteuerung. Dies ermöglicht eine höhere Flexibilität hinsichtlich Einsatz der Software und Wiederverwendbarkeit einzelner Programmkomponenten.
Im Detail betrachtet beinhaltet das Model alle relevanten Daten des Programms, die weitestgehend Plattform-unabhängig verwendet werden können, die aber ohne weitere Software-Bausteine für den Nutzer noch nicht handhabbar sind. Diese Verwendbarkeit wird durch View und Controller gegeben, welche durch die klare Abgrenzung zum Model für verschiedenste Plattformen konzipiert werden können und damit das Programm vielfältig einsetzbar machen können.

1.2 MVC im OntoWiki

Das OntoWiki basiert auf dem Entwurfsmuster MVC. Das Model stellen dabei die Ontologien dar. Diese Datenbasis ist der Kern des OntoWikis und muss mit Hilfe von einem Controllersystem verwaltet werden. Aus einer Controller-Basisklasse, welche selbst von Zend_Controller_Action erbt, leiten sich verschiedene Controller ab, die jeweils spezielle Aufgaben übernehmen (IndexController, DebugController, ModuleController, ApplicationController, ResourceController, ModelController; siehe Grafik!). Weiterhin gibt es ErrorController und ServiceController, die direkt von Zend erben.

- ApplicationController: beinhaltet grundlegende Funktionen des OntoWikis, wie Benutzeranlegen oder die ”About”-Seite
- DebugController: verwaltet verschiedene Caches
- ErrorController: legt Standardaktion fest, die im Fehlerfall durchgeführt werden soll
- IndexController: verantwortlich für verschiedene Meldungen, wie News
- ModelController: dient zum Verändern von Ontologien
- ResourceController: Verwaltet Ressourcen, z.B. Prüfen von URIs
- ServiceController: beinhaltet vielfältige Methoden und Hilfsfunktionen, wie beispielweise eine Suchfunktion
- ModuleController

Das View wird durch zahlreiche Templates im *.phtml-Format gebildet. Für jedes Fenster bzw. jedes Modul existiert eine solche Datei, sodass eine Veränderung des Layouts flexibel und unabhängig von der Funktionalität durchgeführt werden kann.

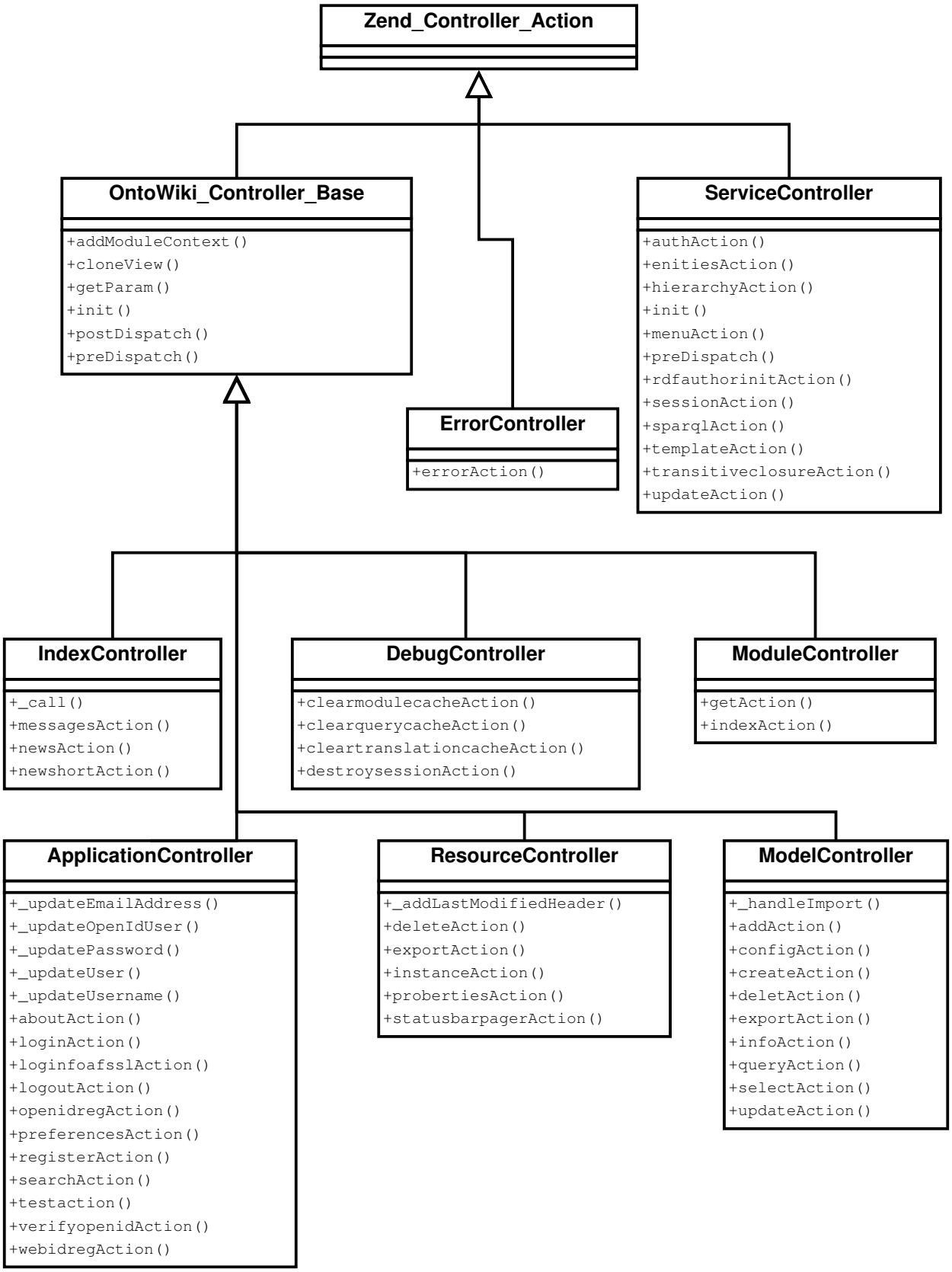


Abbildung: Die Controller des OntoWikis im Überblick

1.2.1 Das Zend Framework

Das Zend Framework ist ein mächtiges und unter Webentwicklern auch beliebtes Framework für Webauftritte. Es bietet eine generelle Grundstruktur nach dem Model-View-Controller Prinzip und stellt allerlei Funktionen und Interfaces zur Verfügung. Unter anderem werden wir zum Beispiel die Zend_Mail Klasse für unser Projekt nutzen.
Eine Übersicht der von Zend bereitgestellten Klassen befindet sich hier:
http://framework.zend.com/manual/en/reference.html
OntoWiki liegt eben dieses Framework zugrunde. Neben den Vererbungszweigen zwischen OntoWiki Controllern und Zend_Controller_Action erbt unter anderem auch OntoWiki_View von Zend_View, wodurch der Inhalt der letztendlichen Webseite für Nutzer mithilfe von Templates realisiert wird.

1.2.2 Das Erfurt API

Neben dem Zend Framework nutzt OntoWiki ein weiteres Interface: das Erfurt API. Dieses regelt die Grundfunktionalität für die Datenhaltung mittels RDF, stellt eine Zugriffskontrolle zur Verfügung und regelt das Benutzermanagement.
Die Zugriffskontrolle wird beispielsweise wie folgt geregelt:

```
<http://localhost/OntoWiki/Config/Test>
<http://ns.ontowiki.net/SysOnt/grantAccess>
<http://ns.ontowiki.net/SysOnt/registerNewUser>
```

Durch diese Befehle wird dem Nutzer ”Test” das Recht verliehen, einen neuen Nutzer zu registrieren.

Das Erfurt API unterstützt ebenfalls Caching (auch basierend auf Zend: Zend_Cache), wodurch es möglich ist, bereits geladene Elemente, Objekte, Rückgabewerte von Funktionen oder Sonstiges für zukünftigen Nutzen zu speichern, was sich positiv auf die Schnelligkeit der Seite auswirkt.

Die Versionsverwaltung der Ressourcen wird ebenfalls mittels der Erfurt API realisiert. Aktionen wie statement-added, statement-removed und statement-changed werden erfasst, um alle Änderungen an den Ressourcen zu verfolgen. Dies kann zusätzlich auf bestimmte Modele oder Nutzer eingeschränkt werden.

1.3 Implementierung eigener Sichten

Ein Großteil des dynamischen Inhalts des OntoWiki wird mit der JavaScript API realisiert, die wiederum jQuery nutzt. Das allgemeine Aussehen der Seiten wird über CSS beschrieben. Der Inhalt des OntoWiki wird durch Templates umgesetzt, die auf Zend_View basieren. So ein Template definiert ein bestimmtes Layout für die Seite, in das dann zum Beispiel das Hauptfenster und eine Navigation geladen werden können. Module - unter anderem auch eigene - werden dann explizit in das Hauptfenster eingefügt.

1.4 Packages

Als ”Package” bzw. ”Paket” bezeichnet man in der Softwareentwicklung die explizite Gruppierung von Klassen, Schnittstellen und Exceptions, die zusammen eine logische und funktionale Einheit bilden. Diese Gruppierung kann für php, und damit für das Onto-Wiki, entweder (bevorzugt) durch die Dokumentation (”@package”- bzw. ”@subpackage”-Tag am Anfang jeder Klasse, Klassen ohne Tag gehören zum ”default”-Package) oder auch (grob) durch eine passend gewählte Ordnerstruktur erfolgen. Einige Packages im OntoWiki sind:

- Application
- MVC (ApplicationController.php, DebugController.php, ErrorController.php, ...)
- Menu (Exception.php, Menu.php)
- Dispatcher (Dispatcher.php)
- Message (Message.php)
- Model (...)
- Module (...)
- Navigation (...)
- Pager (...)
- Plugin (...)
- Request (...)
- Toolbar (...)
- Url (...)
- Utils (...)
- View (...)
- Default (OntoWiki.php, Resource.php, ...)

Problematisch ist, dass relativ viele Klassen nicht mit einem @package-Tag versehen sind und somit standardmäßig in ”default” zusammengefasst werden.
Über die Ordnerstruktur ist OntoWiki zunächst in die Teile ”application” (die Kernfunktionalitäten), ”extensions” (Erweiterungen des Funktionsumfangs) und ”libraries” (Programmbibliotheken) aufgeteilt. Der application-Ordner verzweigt sich weiter in ”classes”, ”config”, ”controllers”, ”scripts”, ”tests” und ”views”, während ”libraries” sich in ”ARC2”, ”Erfurt”, ”RDFauthor” und ”Zend” aufspaltet. Im ”extensions”-Ordner befinden sich entsprechend die Unterordner zu den einzelnen Erweiterungen.

2 Plugin-Strukturierung am Beispiel des Bookmarklet-Plugins

Im folgenden wird der Aufbau einer Erweiterung anhand des Bookmarklet-Plugins erläutert. Die Bookmarklet-Erweiterung ermöglicht das komfortable Hinzufügen von Inhalten zu einer bestehenden Ontologie. Für den OntoWiki-User befindet sich auf der Darstellungsseite einer Ontologie-Instanz ein Link, der als Bookmark im Browser gespeichert werden kann. Klickt man beim Surfen auf dieses Bookmark, so wird eine Javascript-Funktion ausgeführt, die das Hinzufügen der aktuell offenen Website zur gewählten Ontologie zur Folge hat.
Da die Erweiterung mit Erläuterung und Bookmarklet-Link als Fenster im OntoWiki dargestellt wird, eignet sie sich hervorragend zur Analyse und der damit verbundenen Erkenntnisgewinnung für unsere Invitation-Erweiterung.

2.1 Möglichkeiten für Erweiterungen

Der Aufbau einer Erweiterung folgt in der OntoWiki festen Vorgaben. Zuerst gibt es verschiedene Arten von Erweiterungen: Komponenten, Plugins und Module.
Komponenten sind oftmals die Bestandteile, die im Hauptfenster des OntoWikis zu sehen sind. Man kann sie unter anderem mit der Navigation verknüpfen, sodass sie als Menüpunkt für den Benutzer erreichbar sind. Komponenten können allerdings auch im Hintergrund fungieren und dort als MVC-Controller arbeiten.
Die zweite Möglichkeit ist das Plugin: Plugins sind die flexibelsten Erweiterungen des OntoWikis. Sie verfügen in der Regel nicht über visuelle Elemente und können fast jede Art von Funktionalität übernehmen. Plugins werden meist durch Events getriggert.
Als weitere Variante kann ein Modul verwendet werden: diese sind entweder im linken Navigationsbereich als eigenständige Fenster zu finden oder innerhalb von Komponenten, wo sie Funktionalitäten ergänzen. Bei der Bookmarklet-Erweiterung handelt es sich also um ein Modul.

2.2 Dateistruktur

Im OntoWiki-Hauptverzeichnis gibt es einen Ordner ”extensions”, in dem alle Erweiterungen abgelegt werden. Dafür wird für jede Erweiterung ein Unterordner angelegt, der genau den Name der Erweiterung trägt (”extensions/bookmarklet”). In dem Verzeichnis der Erweiterung gibt es mehrere Dateien: Zum einen die ”default.ini”, welche Definitionen zur Erweiterung enthält. Die eigentlichen PHP-Dateien, die den Quellcode der Erweiterung beinhalten, müssen alle mit dem Namen der Erweiterung beginnen und mit dem Erweiterungstyp enden, in unserem Fall ”BookmarkletModule.php”. Weiterhin befindet sich im Verzeichnis die Datei "bookmarklet.phtml", welche die Basis für die graphische Ausgabe darstellt.

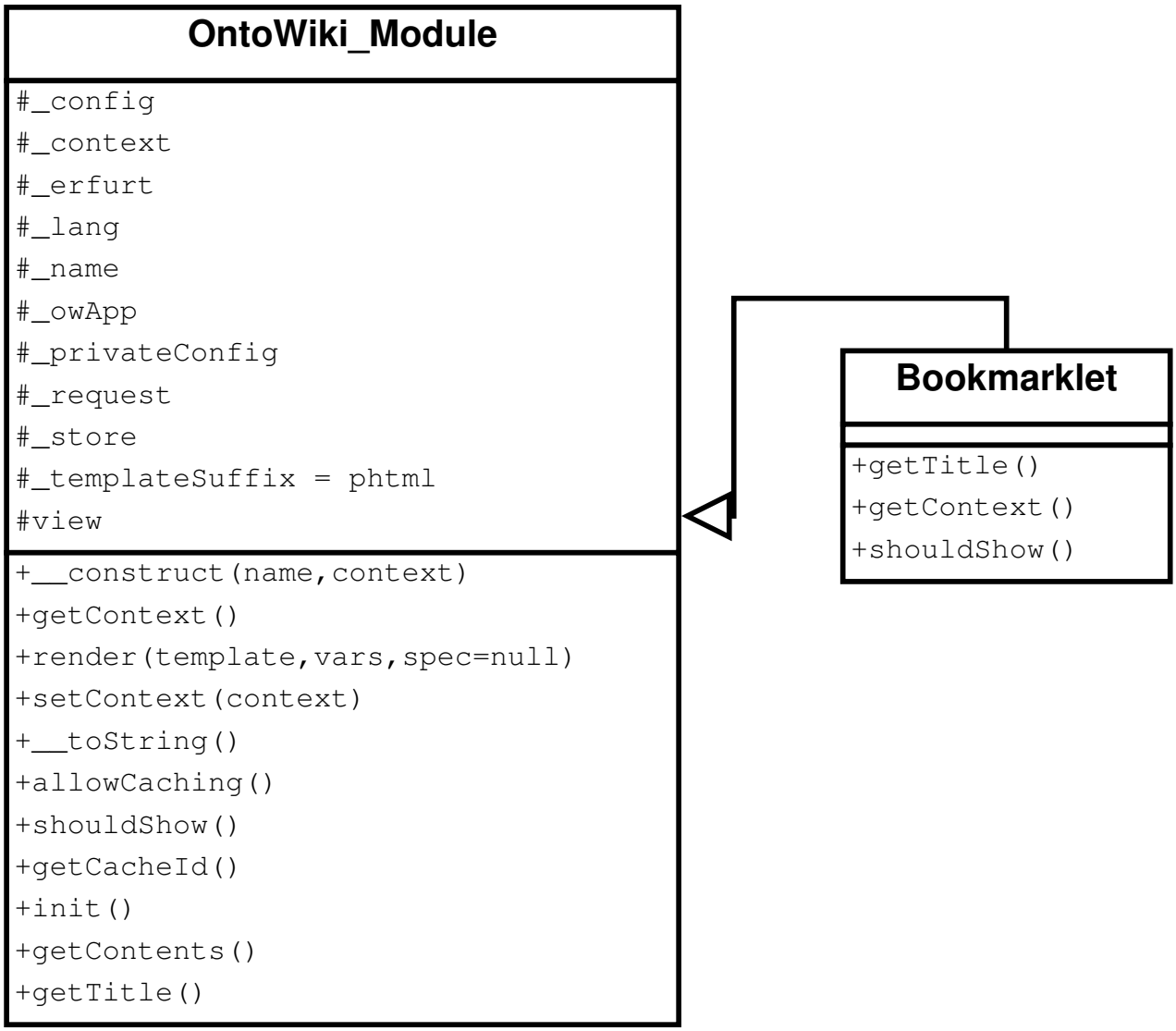
2.3 default.ini

In dieser Datei werden grundlegende Einstellungen des Plugins gespeichert. Dabei handelt es sich beispielsweise um den Aktivierungszustand, Name und Beschreibung der Erweiterung, sowie Informationen zum Autor. Im Falle des Bookmarklets erfährt man durch die default.ini also unter anderem, dass das Plugin (standardmäßig) nicht aktiv ist, dass der korrekte Titel ”RDFauthor Bookmarklet” ist und dass die Website des Autors unter ”http://aksw.org” zu finden ist.
Grundsätzlich kann der Aktivierungszustand auch in der Konfiguration des OntoWikis umgestellt

werden. Diese Einstellung ist der default.ini übergeordnet, d.h. gegebenfalls wird der Wert der default.ini ignoriert.

2.4 BookmarkletModule.php

Hierbei handelt es sich um den funktionalen Kern der Erweiterung, der automatisch durch den extension Manager der OntoWiki erfasst und eingebunden wird.
Nach den phpDocumentor-Einträgen, die Informationen zur Klasse und zum Autor beinhalten, folgt die eigentliche Klasse ”BookmarkletModule”, welche wie alle Module von ”OntoWiki_Module” erbt. Nun werden einige Methoden überschrieben, um sie dem gewünschten Verhalten der Erweiterung anzupassen.
Zuerst wird die Methode getTitle() überschrieben, sodass sie als Rückgabe den Titel der Erweiterung ("Bookmarklet") ausgibt.
Weiterhin wird die Methode getContents() überschrieben. Damit wird dem Modul der eigentliche Inhalt hinzugefügt. In der Bookmarklet-Erweiterung werden an dieser Stelle hauptsätzlich verschiedene Werte, wie Dateipfade von der aktuellen Ontologie (zu der gerade etwas hinzugefügt werden kann) oder Informationen zum Autor und zur OntoWiki-Instanz, an die Präsentations-Einheit (dem View des MVC) weitergegeben. Zudem wird der Frontcontroller festgelegt. Abschließened wird noch das Template, welches zur graphischen Ausgabe verwendet werden soll, zugewiesen (”render(’bookmarklet’)”; dies hat zur Folge, dass die bookmarklet.phtml genutzt wird), es wird der gerenderte Code zurückgegeben und damit wird letztendlich die Erweiterung auf der OntoWiki-GUI gezeichnet.
Die Methode shouldShow(), die als letztes überschrieben wird, bestimmt, ob die Erweiterung angezeigt werden soll oder nicht. In diesem Fall wird das Bookmarklet-Modul immer dann ausgegeben, wenn die Ontologie beschreibbar ist (ansonsten wäre das Modul auch zwecklos).



2.5 bookmarklet.phtml

Diese Datei wird durch den render()-Befehl der BookmarkletModule.php in die Erweiterung eingebunden und dient damit als Grundlage für die graphische Oberfläche. Inhaltlich bietet diese Datei entsprechend verschiedene Ein- und Ausgabe-Befehle für das Popup, die weitestgehend auf Javascript basieren. Beispielsweise werden dynamisch, je nach Bedarf, verschieden viele Eingabefelder erzeugt. Außerdem wird natürlich der Javascript-Link erstellt, der schließlich als Bookmarklet gespeichert dafür dient, Daten an die Ontologie zu überführen.