Entwurfsbeschreibung

Inhaltsverzeichnis:

1. Allgemeines

Das OntoWiki ist eine Webgestützte Software zum Erzeugen, Manipulieren und Durchsuchen von Ontologien bzw. den dahinter stehenden Daten. Im allgemeinen ist es ein browserbasierendes Werkzeug. Es ist an ein normales Wiki angelehnt und es sollte mit Hilfe des OntoWikis leichter sein Fehler zu beheben als mit einen normalen Wiki und schwerer selbst welche zu machen. Es ist somit ein generischer Browser für das semantic Web, welches auf der Grundlage von RDF arbeitet, aber auch eine Vielzahl von anderen Kommunikations- und Interaktionsmöglichkeiten unterstützt.
OntoWiki ist ein Ontologie-Editor zur Unterstützung agiler, verteilter Verarbeitung von Wissensbasen. Die Aufgabe eines OWL-Reasoners ist es, implizit und explizit in einer Ontologie vorhandenes Wissen zu ermitteln. Der Reasoning-Service ist somit einer der zentralen Komponenten eines Ontologie-Editors. Auch die Integration von Tools zur einfacheren Konstruktion von Ontologien mit Methoden des maschinellen Lernens ist eine sinnvolle Erweiterung und ein aktuell interessantes Forschungsgebiet. Eine Herausforderung besteht darin diese Komponenten in geeigneter Weise zu integrieren.

2. Produktübersicht

Zu implementieren ist ein View für das OntoWiki. Um die Daten in der Bier-Ontologie auszunutzen bzw. besser darzustellen wurden zwei Erweiterungen entwickelt.

2.1 Webseitenvorschau

Für einzelne Brauereien, die in der Eigenschaft WWW eine Internetseite angegeben haben, wird über den Reiter _Website_ die Vorschau der Webseite angezeigt.
Dazu wird aus den Eigenschaften der aktuell angezeigten Instanz die Eigenschaft WWW ausgewählt (über das Model). Ist diese nicht vorhanden, so erscheint der Hinweis _URL must be in property 'WWW'_. Andernfalls wird ein Iframe als zusätzlicher Seiteninhalt (Inhalt des _Views_) gerendert (d.h. angezeigt), der die unter der in WWW angegebenen URL erreichbare Internetseite anzeigt.
Dieses View greift dabei auf die allgemein von der View-Klasse abgeleiteten Eigenschaften und Methoden zu um als Reiter im OntoWiki zu erscheinen und um seinen Inhalt anzuzeigen.

2.2 Tabellengrafik (Alkoholgehalt)

Um Biersorten bezüglich ihres Alkoholgehaltes grafisch miteinander vergleichen zu können wird von diesem View eine Veranschaulichung erzeugt: Eine Tabelle mit Zellen, deren Länge sich zueinander verhalten wie jeweils das Verhältnis des Alkoholgehaltes der durch sie repräsentierten Biersorten.
Über das Model wird für alle Instanzen der Alkoholgehalt abgefragt. Für jeden Volumenprozentwert des Alkohols in der Biersorte wird dann eine Zelle deren Länge vom Alkoholgehalt abhängt als zu rendernd aufgenommen sowie der Name der Biersorte (als Link auf die Instanz) und der numerische Alkoholwert hinzugefügt.
Wiederum die Eigenschaften und Methoden des generischen Views nutzend wird dann

der zu rendernde Content dargestellt.

3. Grundsätzliche Struktur- und Entwurfsprinzipien des Gesamtsystems

3.1 Projekt OntoWiki

3.1.1 Visualisierung der Inhalte

Das OntoWiki soll sowohl Browser als auch Editor gleichzeitig sein. In Abbildung 1 erkennt man in der Mitte den eigentlichen Hauptinhalt. Hier das von uns implementiert View des Alkoholgehaltes. Weiterhin existieren eine linke und eine rechte Seitenleiste. In der linken Seitenleiste bietet sich dem Benutzer den Hauptinhalt auszuwählen. Hier Brauerei oder Bier. Es bietet sich ihm dort auch noch die Möglichkeit, sich in das Sytem einzuloggen oder die Sprache zu ändern. In der rechten Seitenleiste kann der Inhalt mittels einer Suche ausgewählt werden. Im Hauptbereich wird dann die Auswahl in Form einer Liste oder einer Grafik angezeigt.

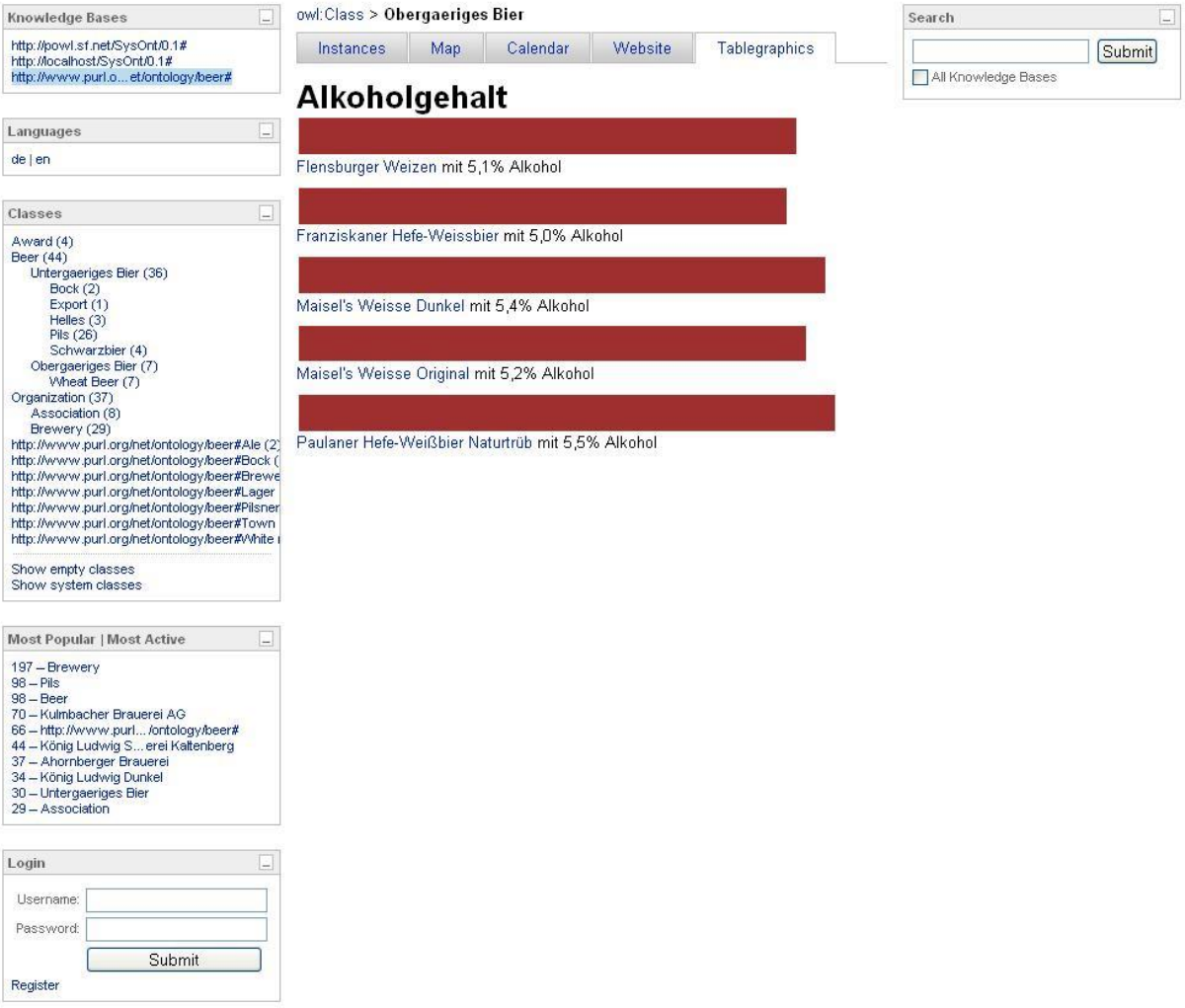


Abbildung 1.

3.1.2 Verschiedene Sichten

OntoWiki identifiziert die innerhalb der Instanzen verwendeten Relationen und Attribute und bietet sie für eine erweiterte Listenansicht (unter Show Properties) an.
OntoWiki Prototyp unterstützt unterschiedlichen Sichten auf Instanzen.

API ist eine Schnittstelle, die von einem Softwaresystem anderen Programmen zur Anbindung an das eigenem System zur Verfügung gestellt wird. (Englisch: application programming interface Deutsch: Schnittstelle zur Anwendungsprogrammierung).
Neben dem Zugriff auf Datenbanken oder die Hardware wie Festplatte oder Grafikkarte kann eine API auch das Erstellen von Komponenten der grafischen Benutzeroberfläche ermöglichen oder vereinfachen.

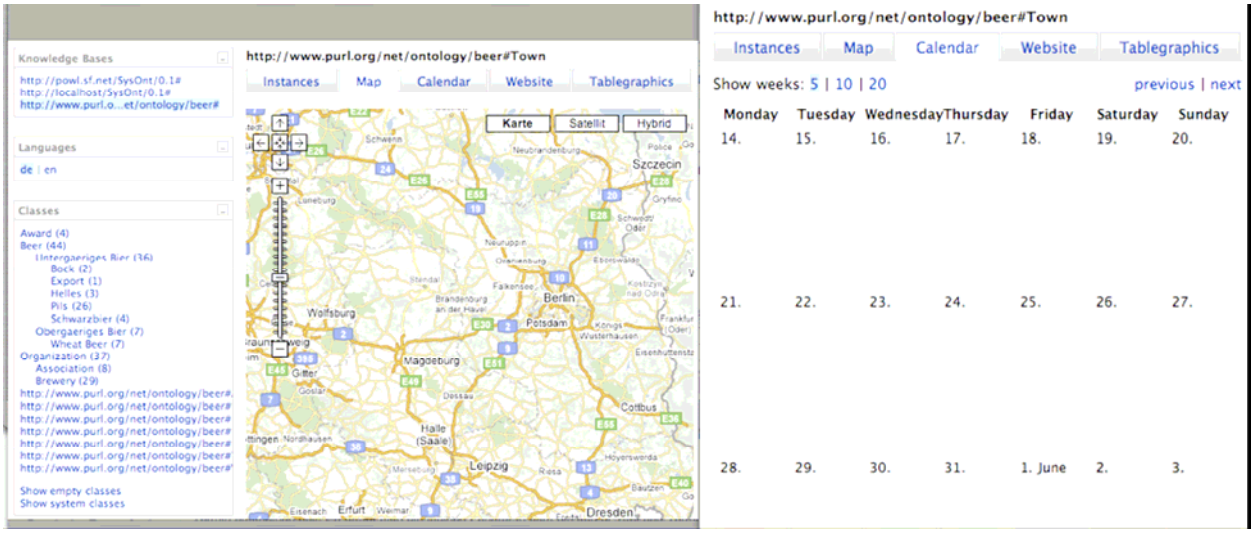
Google API-Key:
Google veröffentlichte im Frühling 2002 Google Web API, über die es registrierten Entwicklern möglich ist, eigene Anwendungen bzw. Schnittstellen zu schreiben, die den Datenbestand von Google abfragen. Die Abfragen sind pro registriertem Anwender auf 1000 pro Tag begrenzt. Es gibt inzwischen eine Vielzahl von Anwendungen, die auf dieser API aufbauen und von Anwendern per eigenem Developer Key freigeschaltet werden können. Die Dienstleistung befindet sich noch in der Betaphase.
Das API für Google Maps ermöglicht es mit JavaScript das Einbetten von Google Maps in eigenen Webseiten. Es können Overlays (darunter Markierungen und Mehrfachlinien) in die Karte eingefügt und abgeblendete "Infofenster" wie Google Maps angezeigt werden.

Beispiel: **Kartendarstellung** bietet Informationen über den Standort zu den ausgewählten Daten. Die Sicht wird durch die Integration der *Google Maps API* realisiert. Die Integration ist bidirektional, deshalb werden Instanzdaten zu auf der Karte dargestellten Objekten automatisch aus der Wissensbasis abgerufen und in der Kartendarstellung sichtbar gemacht.
Das API für Google Maps ist ein kostenloser Beta-Service, der auf jeder Webseite verwendet werden darf, auf die die Nutzer kostenlos zugreifen können.

Die Kartendarstellung ermöglicht es somit die Ausgewählten Daten auf einer Karte darzustellen, wenn sie Werte enthalten die geografischen Daten wie Längen- und Breitengrad entsprechen. Realisiert wurde dieses Sicht durch Integration der Google Api.

Kalendersicht: Jedes dargestellte Kalenderobjekt ist mit der Individualsicht der korrespondierenden Ressource verknüpft. Die Seitenleiste bietet eine Exportfunktion, mit der die Kalenderobjekte in das verbreitete iCal Format umgewandelt werden können, um sie in andere Kalenderanwendungen wie Sunbird oder Google Calendar zu importieren.
Calendar bietet viele verschiedene Ansichtsmöglichkeiten. Das Hauptkriterium zur Auswahl der richtigen Ansicht ist wohl die Anzahl der Termine und deren Dichte (Tages-Ansicht, Wochen-Ansicht, Monats-Ansicht, Next 4 Days, Agenda-Alle...)

Die Kalenderdarstellung ermöglicht es somit die ausgewählten Daten in einem Kalender darzustellen, wenn die Daten vom Datentyp xsd:date sind.



3.2 Erweiterbarkeit des Projektes (Beispiel View)

Unter Punkt 4. wird noch allgemein auf die MVC Architektur des Zend Frameworks mit seiner Benutzung im OntoWiki eingegangen, ich werde mich hier zunächst auf die eigenen Implementation von Views bechränken. Beginnen möchte ich hierbei mit dem Standard-Template View welches durch die Controller in apps/controllers und den Ordner template/default verwirklicht wurde.

3.2.1 Generelle Instanziierung der nötigen Zend-Objekte des OntoWiki

Nach der Instanziierung eines neuen Objekts der Zend-Klasse "Zend_Controller_Front" und unter Einbeziehung der aus /apps/config/settings.php wird mithilfe der Controller Methoden setRouter() und setControllerDirectory() der Router für die Zend-Standard Pfadstruktur nach dem Schema 'http://hostname.xy/controller/action' instanziiert und an den Zend_Controller_Front übergeben, sowie mittels der zweiten Methode das Verzeichnis in dem die jeweiligen Controller php-Dateien liegen zugewiesen. Weiterhin wird eine neue Instanz von Zend_View instanziiert, die im Weiteren Verwendung findet. Hierbei fällt auf, das kein eigentliches Model verwendet wird, die Geschäftslogik des OntoWiki wird in den entsprechen Controller Dateien erledigt, beziehungsweise in den Klassen der Quelldateien von lib/OntoWiki. Da aber bereits ein leerer Ordner app/model im aktuellen SVN-Checkout vorhanden ist, liegt es mir nahe zu vermuten, das darin eine neue zukünftige Trennung von Controller/Model vorgesehen. ist. Daher ist momentan eher von einer View/Controller-Architektur zu sprechen denn von einem vollständig

modularen MVC-Aufbau.

3.2.2 Aufbau des Standard-Controllers

Nachdem der Zend_Controller_Front über alle nötigen Informationen der Pfadstruktur verfügt wird er mit dispatch() gestartet. Von nun an wird beispielsweise bei einem Aufruf von 'http://hostname.xy/test/abc' die Methode abcAction() im Controller TestController.php im vorhergehend zugewiesenen Controller-Pfad aufgerufen. Diese Funktionalität bietet der definierte Router, wobei auch Methoden zur Defintion beliebiger Action-Maps, dass heißt Schemen wie Request-Pfade in Controller Methodenaufrufe übersetzt werden, angeboten werden. Alle diese Voreinstellungen werden in der Settings.php erledigt. In den Controllern des Standard-Views werden Aufrufe zu den Seiten aus /template/gewähltestemplate (bei uns noch default) umgeformt wie im Router definiert.

3.2.3 Aufbau des Standard-Views

Das Standard-View wird ebenfalls wie ja bereits erwähnt in der Settings.php instanziiert. In den Controller-Action-Aufrufen wird dann deren Referenz benutzt, die zuvor in der Zend-Registry gespeichert wurden. Der Aufbau des View sowie das Design sind auch im Template durch Cascading Style Sheets (kurz CSS) definiert. Aus diesen Stylesheets und den entsprechenden Daten werden von dem Objekt der View-Klasse dann die HTMLDarstellung generiert.

3.2.4 Modifikation des Standard-Views und Controllers für Implementation eingener View

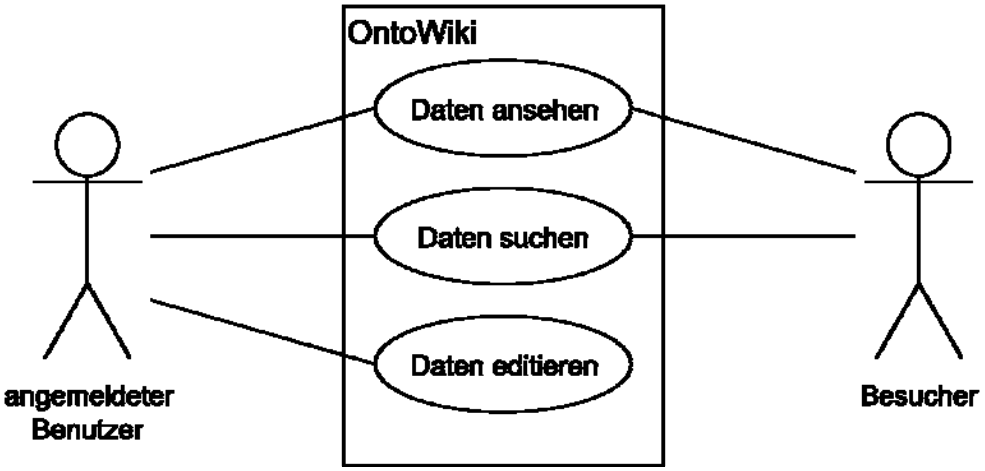
Nun werde ich kurz unsere Modifikation des Views darstellen um neue Datenansichten beziehungsweise Hervorhebung andere Aspekte zu ermöglichen. In unserem konkreten Fall haben wir uns nicht nur auf das View beschränkt, sondern auch Modifikationen des Controllers vorgenommen, um weitere Funktionalität zu erhalten. Modifiziert wurde der ResourceController.php, also bei dem vorhanden Mapping der OntoWiki alle Actions unter http://hostname.xy/resource. Konkret wurden die Methoden für die Actions 'tablegfx' zur Darstellung des Alkoholgehaltes von Bier (leider nur als Tabellen, da PHP auf dem zur Verfügung stehenden Server ohne GD-Unterstüzung läuft), sowie die Action 'statistics' die eine Webseite anzeigt, die mit der aktuellen Resourceninstanz verknüpft ist. Weiterhin wurden für diese beiden Actions noch Vorlagen in den template/default-Ordner eingefügt.
Die eigentliche View bleibt unverändert, alle neuen Aktionen arbeiten nur im 'content'-Bereich des Webseitenlayouts, wie in dem oben erwähnten CSS festgelegt. Außerdem wurden die "Tab-Generierung" für die Navigation in dem ResourceController.php angepasst, damit diese Tableiste stets konsistent und funktionsfähig ist. Eine Anpassung im Bezug auf die Kartenansicht wurde vorgenommen, indem die einzuzeichnenden Properties auf 'lat' und 'long' geändert wurden. Ausserdem ein Google API-Key für den pcai042 eingetragen. Das sind alle Modifikationen die wir in unserem View implentiert haben, allgemein ist eine Erweiterung des Projekts wie im Folgenden erklärt zu implementieren.

3.2.5 Erweiterbarkeit des Projektes im Allgemeinen

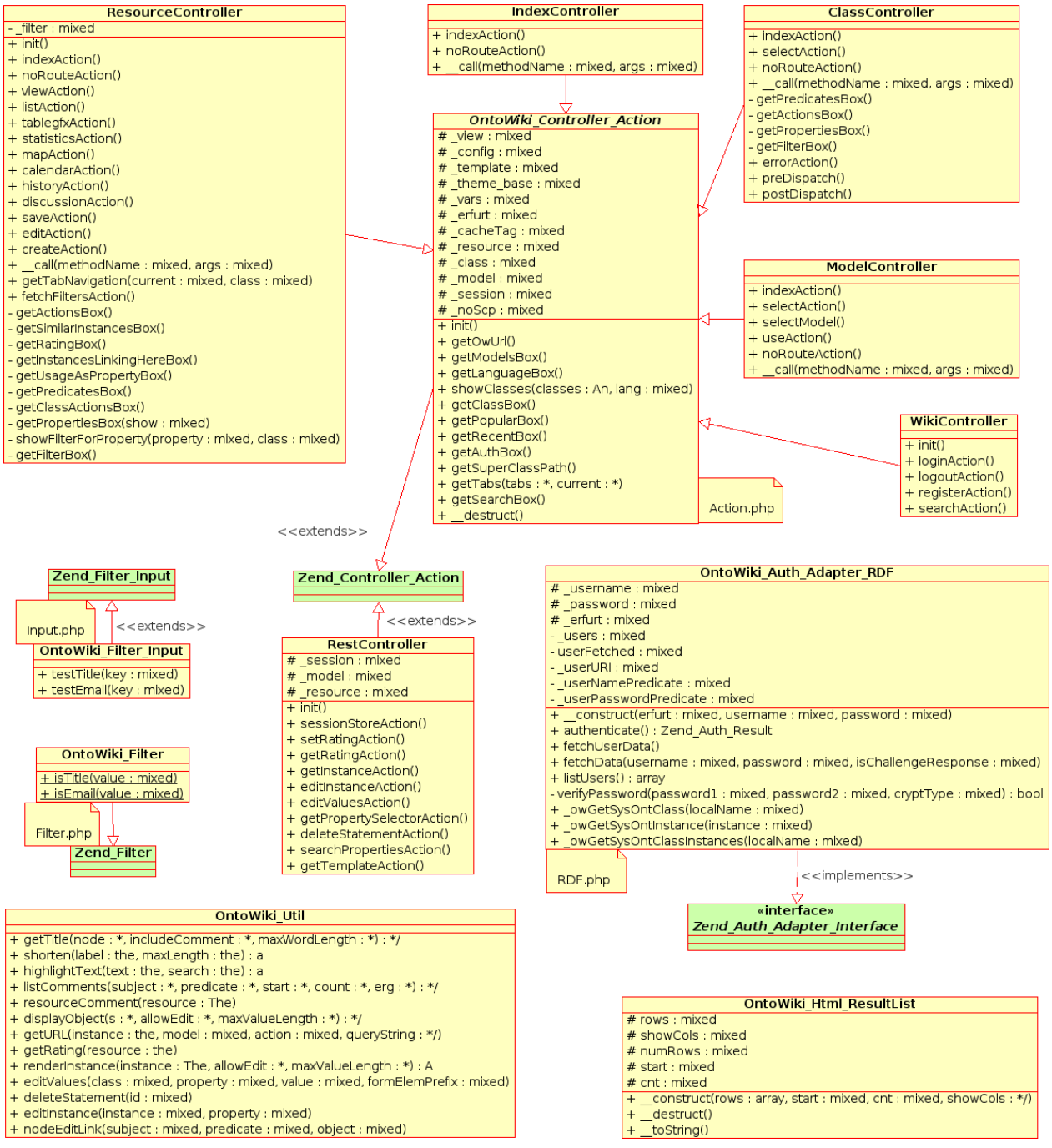
Das Projekt ist im Allgemeinen leicht erweiterbar, wenn auch die Modularisierung bei der aktuellen Implementation noch verbesserungswürdig erscheint, Suchwort Modell / Controller Disjunktheit. Eine Änderung der Produktfunktionen wird durch die Modifikation des Controllers erreicht. Dazu muss die View in Form der Templates entweder modifiziert werden beziehungsweise ein komplettes Rewrite eines neuen Templates erfolgen. Die in lib/OntoWiki angebotenen Funktionen zur Datenaufbereitung sind die einzigen Interoperablen Methoden, im Gegensatz zu den Controllern, die ihre Funktionen jeweils nur im zugehörigen View anbieten. (vgl. Einheit View/Controller 3.2.1).

4. Grundsätzliche Struktur- und Entwurfsprinzipien für einzelne Pakete

4.1 Use-Case Diagramm:



4.2 Klassendiagramm:



4.3 Zend Framework:

Das Zend Framework ist ein qualitativ hochwertiges Open Source Fremework für die Entwicklung von Webanwendungen und Web Services mit PHP. Das Zend Framework bietet einfach anzuwendende und leistungsfähige Funktionalität. Es stellt Lösung bereit, um moderne, widerstandsfähige und sichere Webseite zu erstellen.

Pakete/Klassen:

Der Zend_Controller:

stellt das Fundament für den Aufbau der Webseite auf Basis des Model-View-Controller (MVC) Entwurfsmusters bereit.

Zend_Controller_Router:

wird verwendet, um Router (siehe 3.2.1) zu definieren.

Zend_Controller_Front:

verarbeitet alle Anfragen, die der Server erhält, und ist letztendlich dafür verantwortlich, die Anfragen an die ActionController (Zend_Controller_Action) zu deligieren.

Zend_Controller_Action:

ist die elementare Controller Komponente. Jeder Controller ist eine einzelne Klasse, welche die Zend_Controller_Action Klasse erweitert und Methoden für die Aktionen enthält.

Zend_Filter:

Der Zend_Filter Bestandteil liefert ein Set geläufiger erforderlicher Datenfilter. Es liefert auch einen einfachen Filter, der Einheiten verkettet. Durch mehrfache Filter können die einzelnen Daten in einer benutzerbestimmten Ordnung angewendet werden.
Methode:

```
<?php
require_once 'Zend/Filter/HtmlEntities.php';
$htmlEntities = new Zend_Filter_HtmlEntities();
echo $htmlEntities->filter('&'); // &amp;
echo $htmlEntities->filter(""); // &quot;
?>
```

Zend_Auth:

Zend_Auth stellt eine API für Authentisierungen zur Verfügung und enthält konkrete Authentisierungs Adapter für geläufige Gebrauchsfalldrehbücher. Ein Zend_Auth Adapter wird benutzt, um gegen einen bestimmten Typen, Authentisierung Service, wie zum Beispiel LDAP, RDBMS oder Datei-gegründete Speicherung, zu beglaubigen.

Inferface:

Zend_Auth_Adapter_Interface wirde von OntoWiki benutzt. Jede Zend_Auth Adapterkategorie führt Zend_Auth_Adapter_Interface ein. Diese Schnittstelle definiert eine Methode authenticate(), das eine Adapterkategorie für das Durchführen einer Authentisierungsabfrage einführen muss. Jede Adapterkategorie muss vor dem initialisieren von authenticate() vorbereitet werden.
Zend_Auth wird nur mit Authentisierung und nicht mit Ermächtigung in Verbindung gebracht. Authentisierungen werden lose definiert, wie zum Beispiel das Feststellen, ob eine Instanz wirklich das ist was sie behauptet zu sein. Ermächtigung ist der Prozess des Endscheidens, ob man auf eine Instanz zuzugreifen berechtigt ist, oder ob man Operationen auf ihr durchzuführen darf.

Zend_Log:

Zend_Log ist ein Bestandteil für das universelle Protokollieren. Es unterstützt mehrfaches Protokoll backend, formatiert die Meldungen, die zum Protokoll geschickt werden, und filtert Meldungen die protokolliert werden sollen.

Zend_Search:

Zend_Search_Lucene ist eine komplett in PHP 5 geschriebene Textsuchmaschine für viele Zwecke. Da es seinen Index im Dateisystem ablegt und keinen Datenbankserver erfordert, kann es eine Suchfunktion für nahezu jede auf PHP basierende Website bereitstellen.

Zend_View:

Zend_View ist eine Klasse für die Verarbeitung des "View" Teils des Model-View-Controller Entwurfsmusters. Er existiert, um das View Skript von den Model und Controller Skripten zu trennen. Es stellt ein System an Helfern, Ausgabefiltern und Variablenmaskierung bereit.
Zend_View ist unabhängig von einem Template System. Man kann in einen View Skript PHP als Template Sprache verwenden oder Instanzen anderer Template Systeme erstellen und diese verarbeiten.