Allgemeines

Folgende Punkte lassen sich zusammerfassend nennen bei der Analyse der Ontowiki Quellen:

- Ontowiki Pakete unvollständig definiert in den Quellen

Einzig die Paket Definition `html` wurde bei der Analyse gefunden, die Einteilung der Pakete folgt somit einer idealisierten Form, die sich aus einer Quellcodeanalyse ergibt. Die Arbeit mit PHP2XMI gestaltete sich aufgrund des frühen Stadiums der Software und der unvollständigen Dokumentationen des Ontowiki Quellcodes als schwierig.

- Ontowiki basiert auf Zend Framework Komponenten
- Ontowiki nutzt ähnlich aufgebaute Paketstruktur wie Zend Framework

Ontowiki nutzt Komponenten des Zend Frameworks in der Applikation, die Strukturierung der Pakete wurde somit an der Zend Struktur angelehnt. Das Paket `Ontowiki_Html` stellt aber bspw. eine Ontowiki eigene Erweiterung dieser Struktur dar.

- Ontowiki als Powl Frontend

Ontowiki greift bei der Modellierung auf das Powl Framework zu. Die mittels Powl importierten oder erstellten Modelle bilden die Basis der Applikation.

Pakete

Die Ontowiki Applikation nutzt Komponenten aus den globalen Paketen `Ontowiki, Powl` und `Zend`. Das Paket `Ontowiki` stellt die Ontowiki eigenen Klassen in Subpaketen zur Verfügung, `Powl` beinhaltet die komplette Applikation auf deren Klassen Ontowiki zugreift. Das Zendframework wird über das Paket `Zend` eingebunden. Die einzelnen Subpakete des Frameworks liegen dabei im „Zend_“ Namensraum. Die Ontowiki Applikation referenziert dabei im Wesentlichen Komponenten der Pakete `Zend_Auth, Zend_Controller, Zend_Filter`.

Paketübersicht.

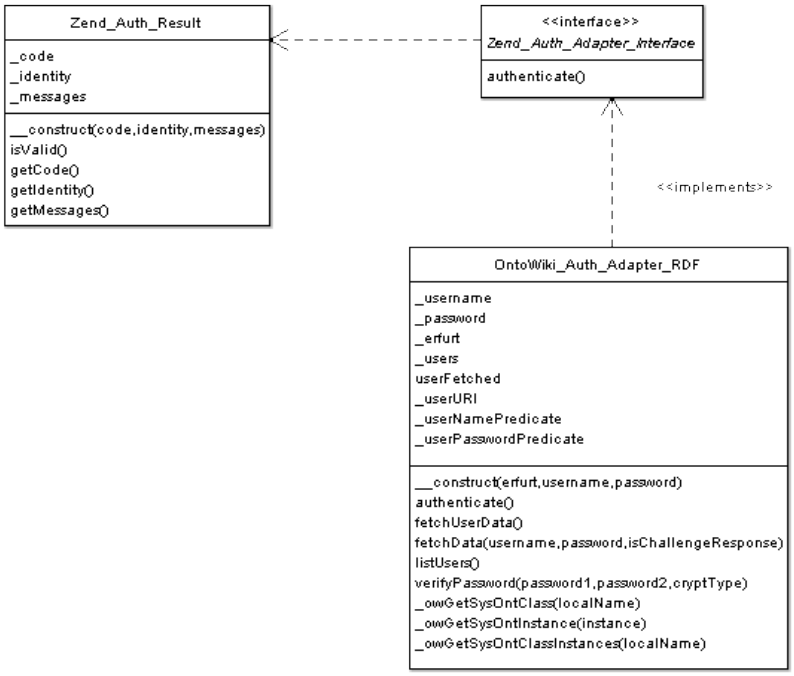
Ontowiki_Auth
Ontowiki_Controller
Ontowiki_Filter
Ontowiki_Html
Ontowiki_Template

POWL

Zend_Acl
Zend_Auth
Zend_Cache
Zend_Config
Zend_Console
Zend_Controller
Zend_Date
Zend_Db
Zend_Feed
Zend_Filter
Zend_Gdata
Zend_Http
Zend_Json
Zend_Locale
Zend_Log
Zend_Mail
Zend_Measure
Zend_Mime
Zend_Pdf
Zend_Request
Zend_Rest
Zend_Search
Zend_Server
Zend_Service
Zend_Session
Zend_Translate
Zend_Uri
Zend_Validate
Zend_View
Zend_XmlRpc

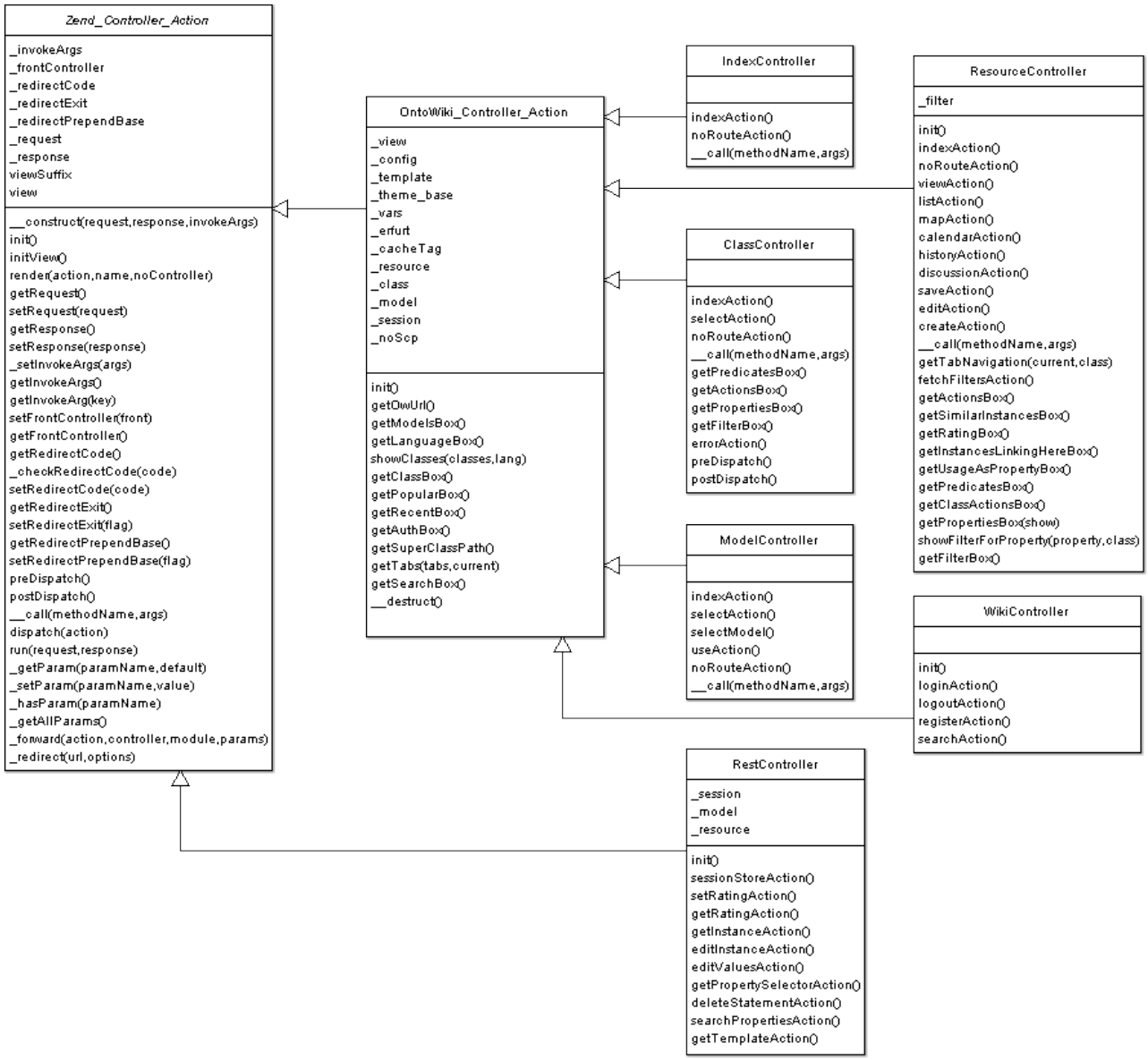
Ontowiki_Auth

Das Ontowiki_Auth Paket enthält die Klasse Ontowiki_Auth_Adapter_RDF, die das Zend Interface Zend_Auth_Adapter_Interface implementiert. Das Interface und dessen assoziierte Klasse Zend_Auth_Result gehören zum Paket Zend_Auth. Die Ontowiki Klasse stellt Funktionen für die Nutzeranmeldung und Bereitstellung der Nutzersession zur Verfügung.


Zend_Auth_Result
_code
_identity
_messages
__construct(code,identity,messages)
isValid()
getCode()
getIdentity()
getMessages()
<<interface>>
Zend_Auth_Adapter_Interface
authenticate()
<<implements>>
OntoWiki_Auth_Adapter_RDF
_username
_password
_erfurt
_users
userFetched
_userURI
_userNamePredicate
_userPasswordPredicate
__construct(erfurt,username,password)
authenticate()
fetchUserData()
fetchData(username,password,isChallengeResponse)
listUsers()
verifyPassword(password1,password2,cryptType)
_owGetSysOntClass(localName)
_owGetSysOntInstance(instance)
_owGetSysOntClassInstances(localName)

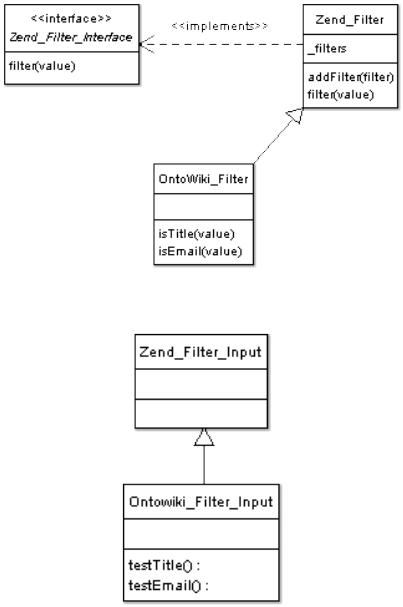
Ontowiki_Controller

Das Paket Ontowiki_Controller setzt die Controllerkomponente des MVC Architekturmusters um. Dabei erben die Klassen ClassController, IndexController, ModelController, ResourceController, WikiController von der abstrakten Klasse Ontowiki_Controller_Action. Diese wiederum ist eine Erweiterung der abstrakten Klasse Zend_Controller_Action aus dem Paket Zend_Controller. Einzig die Klasse RestController erweitert direkt Zend_Controller_Action. Die Controller Klassen steuern den Aufruf der Viewer Instanzen für jede Nutzer Session.


Zend_Controller_Action
_invokeArgs
_frontController
_redirectCode
_redirectExit
_redirectPrependBase
_request
_response
viewSuffix
view
__construct(request,response,invokeArgs)
init()
initView()
render(action,name,noController)
getRequest()
setRequest(request)
getResponse()
setResponse(response)
_setInvokeArgs(args)
getInvokeArgs()
getInvokeArg(key)
setFrontController(front)
getFrontController()
getRedirectCode()
_checkRedirectCode(code)
setRedirectCode(code)
getRedirectExit()
setRedirectExit(flag)
getRedirectPrependBase()
setRedirectPrependBase(flag)
preDispatch()
postDispatch()
__call(methodName,args)
dispatch(action)
run(request,response)
_getParam(paramName,default)
_setParam(paramName,value)
_hasParam(paramName)
_getAllParams()
_forward(action,controller,module,params)
_redirect(url,options)
OntoWiki_Controller_Action
_view
_config
_template
_theme_base
_vars
_erfurt
_cacheTag
_resource
_class
_model
_session
_noScp
init()
getOwUrl()
getModelsBox()
getLanguageBox()
showClasses(classes,lang)
getClassBox()
getPopularBox()
getRecentBox()
getAuthBox()
getSuperClassPath()
getTabs(tabs,current)
getSearchBox()
__destruct()
IndexController
indexAction()
noRouteAction()
__call(methodName,args)
ClassController
indexAction()
selectAction()
noRouteAction()
__call(methodName,args)
getPredicatesBox()
getActionsBox()
getPropertiesBox()
getFilterBox()
errorAction()
preDispatch()
postDispatch()
ModelController
indexAction()
selectAction()
selectModel()
useAction()
noRouteAction()
__call(methodName,args)
RestController
_session
_model
_resource
init()
sessionStoreAction()
setRatingAction()
getRatingAction()
getInstanceAction()
editInstanceAction()
editValuesAction()
getPropertySelectorAction()
deleteStatementAction()
searchPropertiesAction()
getTemplateAction()
ResourceController
_filter
init()
indexAction()
noRouteAction()
viewAction()
listAction()
mapAction()
calendarAction()
historyAction()
discussionAction()
saveAction()
editAction()
createAction()
__call(methodName,args)
getTabNavigation(current,class)
fetchFiltersAction()
getActionsBox()
getSimilarInstancesBox()
getRatingBox()
getInstancesLinkingHereBox()
getUsageAsPropertyBox()
getPredicatesBox()
getClassActionsBox()
getPropertiesBox(show)
showFilterForProperty(property,class)
getFilterBox()
WikiController
init()
loginAction()
logoutAction()
registerAction()
searchAction()

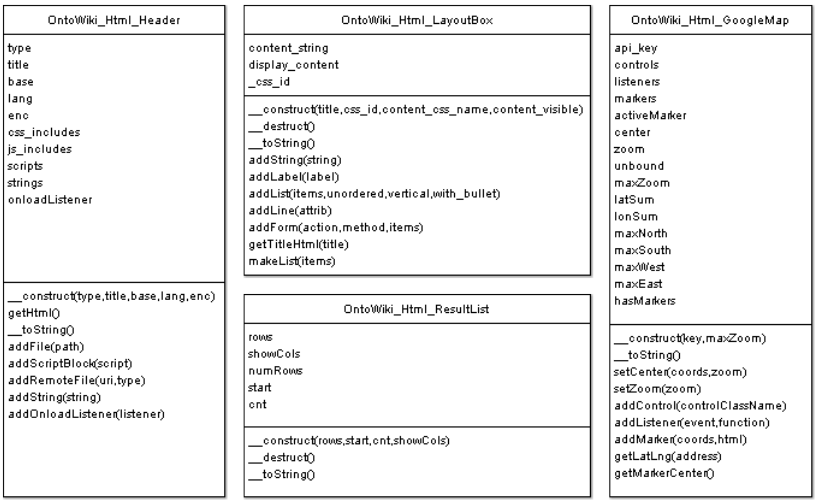
Ontowiki_Filter

Die Klassen des Pakets Filter beinhalten Filtererweiterungen für Ontowiki.



Ontowiki_Html

Das Html Paket stellt Klassen zur Verfügung, die zur View Komponente des MVC Musters gezählt werden können. Die Klasse `OntoWiki_Html_Header` dient als Vorlage für HTML Header, die für den generierten Output verwendet werden. `OntoWiki_Html_LayoutBox` ist eine Basisklasse für die Generierung der Ontowiki Boxen. `OntoWiki_Html_GoogleMap` ist eine Wrapper Klasse für die GoogleMaps API. `OntoWiki_Html_ResultList` generiert Tabellen.

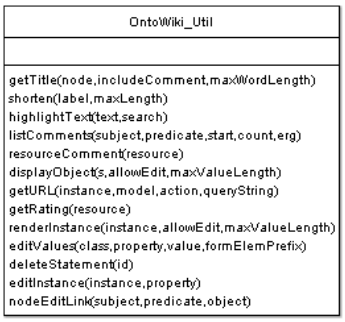


Ontowiki_themes

Das Paket stellt keine Klassen zur Verfügung, sondern beinhaltet als Ganzes Komponenten zur Darstellung der Webapplikation im Browser. Das Subpaket stellt aber die Möglichkeit zur Erweiterung des applikationseigenen Looks dar, weitere Themen können hier definiert werden. Templates können als Erweiterungen in der Darstellung, sogenannte Skins aufgefasst werden. Beim momentanen Entwicklungsstand existiert nur ein `default` Template als Vorlage. Das `themes` Paket gehört unserer Ansicht nach zur Viewer Komponente des MVC Musters.

Weiteres

Die Klasse Ontowiki_Util gehört keinem Subpaket aus dem Namensraum „Ontwiki_“. Sie stellt Hilfefunktionen der Webapplikation zur Verfügung. Sollten weitere Klassen diese Funktionen erweitern ist die Einführung eines Pakets „Ontowiki_Util“ in Betracht zu ziehen.


OntoWiki_Util
getTitle(node,includeComment,maxWordLength)
shorten(label,maxLength)
highlightText(text,search)
listComments(subject,predicate,start,count,erg)
resourceComment(resource)
displayObject(s,allowEdit,maxValueLength)
getURL(instance,model,action,queryString)
getRating(resource)
renderInstance(instance,allowEdit,maxValueLength)
editValues(class,property,value,formElemPrefix)
deleteStatement(id)
editInstance(instance,property)
nodeEditLink(subject,predicate,object)

Strukturierung der APIs

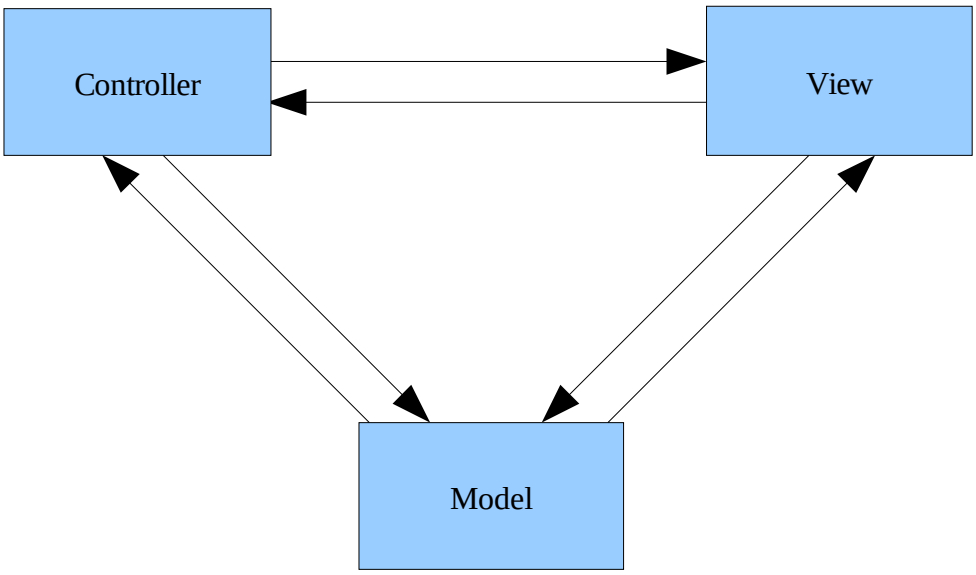
- RDF API for PHP (RAP)
- Google Maps API
- Powl Framework
- Zend Framework

Ontowiki bindet in seinen Komponenten die o.g. APIs ein. Zend stellt z.B. Controllerfunktionen des MVC Musters zur Verfügung oder aber Authentifizierungs- und Session-Management-Funktionen. Das Powl Framework dient der Generierung und Berabeiten von Ontologien und deren Einbindung in Ontowiki als Modelle. Die Modelle folgen dabei der RDF Spezifikation, Ontowiki bedient sich RAP um auf deren Daten zugreifen zu können. Die Google Maps API wird bspw. durch die Klasse `Ontowiki_Html_GoogleMap` im Paket `Ontowiki_Html` referenziert.

Umsetzung der Model-View-Controller-Architektur

Das Model-View-Controller-Konzept, kurz MVC-Konzept, wurde zunächst für Benutzeroberflächen in Smalltalk durch Trygve Reenskaug beschrieben (Seeheim-Modell), gilt mittlerweile aber als de-facto Standard für den Grobentwurf aller komplexen Softwaresysteme.

Das OntoWiki verwendet das Zend-Framework, das bereits alle Voraussetzungen, um ein Webprojekt nach dem Model-View-Controller-Konzept aufbauen zu können, enthält. Bei einer MVC-Architektur wird die Anwendung in die drei wesentlichen Teilkomponenten Model (Modell), View (Präsentation) und Controller (Steuerung) aufgeteilt. Diese drei Komponenten können je nach nach Realisierung unterschiedlich stark voneinander abhängen.



Model

Das Model, zu deutsch Modell, enthält die darzustellenden Daten. Woher die Daten kommen und wie diese zusammenhängen, spielt keine Rolle. Das Modell kennt weder die Präsentation (View) noch die Steuerung (Controller). Es weiß also nicht wie, ob und wie oft es dargestellt und verändert wird.

Im OntoWiki ist die Model Komponente für die Interaktion mit den Daten zuständig. Die Daten stammen hierbei aus einer MySQL-Datenbank, die als Datenquelle dient. Das Model liest die Daten aus der Datenbank und gibt die gelesenen Daten an den Controller zurück. Das Model enthält aber auch Daten vom Controller und speichert sie in der MySQL-Datenbank ab.

View

Das View, zu deutsch Präsentation, ist für die Darstellung der relevanten Daten aus dem Modell zuständig. Es ist nicht für die Interaktion mit dem Benutzer verantwortlich (dies übernimmt der Controller), sondern lediglich für die Beschaffung der Daten aus dem Modell, deren Darstellung und, bei Änderungen im Modell, die Darstellungen passend zu aktualisieren.

Die View Komponente ist im OntoWiki für die Präsentation der Daten zuständig. In diesem Fall geschieht dies in Form von HTML Templates. Die View Komponente erhält die zu präsentierenden Daten vom Controller und interagiert niemals direkt mit dem Model. Lediglich Präsentationslogik, keine Geschäftslogik, ist im View enthalten. Die Verarbeitung der

Geschäftslogik bleibt ausschließlich der Controller Komponente vorenthalten.

Controller

Der Controller, zu deutsch Steuerung, verwaltet die Sichten, nimmt von ihnen Benutzeraktionen entgegen, wertet diese aus und agiert dementsprechend. Er enthält die Intelligenz und steuert den Ablauf der Präsentation.

Im OntoWiki ist die Controller Komponente dafür zuständig, zwischen den anderen beiden Komponenten zu vermitteln. Bei einer Anfrage an den Webserver wird ermittelt, welcher Controller für die Verarbeitung der Anfrage zuständig ist. Wenn nötig, ruft der Controller bei der zuständigen Model Komponente die erforderlichen Daten ab, bereitet sie auf und übergibt sie an die zuständige View Komponente, welche die Daten darstellt. Der Controller kann bei einer Anfrage aber auch nur mit der View Komponente interagieren, wenn keine Daten gelesen oder geschrieben werden müssen. Genauso gut kann der Controller bei einer Anfrage aber auch nur mit der entsprechenden Model Komponente interagieren, wenn nach der Verarbeitung der Daten keine Ausgabe erforderlich ist.

Üblicherweise spielen ein View- und ein Control-Modul zusammen, welche sich gegenseitig kennen und sich auf ein Modell beziehen. Im OntoWiki kann ein Modell aber auch von mehreren View-Control-Paaren gesteuert werden. Datenmaterial aus dem Modell kann so auf verschiedenen Views dargestellt werden. Dies geschieht zum Beispiel in Form von Tabellen und Diagrammen. Verändert einer der dazugehörigen Control-Module den Wert, so sendet das Modell einen Event, der dazu führt, dass sich beide grafischen Controls parallel aktualisieren.

Das OntoWiki verwendet mehrere Controller-Klassen, die öffentliche Methoden, so genannte Aktionen, enthalten. Typische Aktionen sind, hier nur verallgemeinert, anlegen, ändern, löschen, anzeigen, einloggen und ausloggen. Jede Aktionsmethode verarbeitet nur eine Aktion und vermeidet so komplizierte switch- oder else-Anweisungen. Dadurch wird eine unnötige Komplexität vermieden.

Erweiterung des OntoWiki-Projekts

Derzeit befindet sich das OntoWiki-Projekt in einem frühen Stadium und ist mit der Version 0.6 auf dem Weg zum Beta-Release. In der Regel werden Entwicklungen durch Bedürfnisse der Benutzer oder Notwendigkeiten in der Arbeit mit einem System geprägt. Eine frühzeitige Implementierung spezieller Funktionalitäten kann im schlimmsten Fall dazu führen, dass OntoWiki sich dem Einsatz in dem ein oder anderen Gebiet frühzeitig entzieht.
Sinnvolle Erweiterungen sind wohl gerade die, die die Arbeit mit dem OntoWiki flexibler und vielseitiger gestalten und es dadurch einem breiten Spektrum von Einsatzgebieten zugänglich machen.

Einbettung von Feeds

Es ist denkbar die jeweiligen Themen mit Feeds anzureichern. So könnten zu jedem Zeitpunkt ohne Änderungen der Daten innerhalb des OntoWiki aktuelle Informationen bereitgestellt werden, sofern das Feed Bestandteil eines externen Systems ist.
OntoWiki ist bereits in der Lage das RDF-Format (RSS 1.0) zu verarbeiten. Das beste Ergebnis bei geringstem Aufwand wird sicherlich durch die Verwendung bereist verfügbarer Komponenten erzielt. Die Implementierung dieser Funktion erfordert demnach eine Erweiterung der View-Komponente.

Import/ Export von Datensätzen

Die Migration von Datensätzen ist ein wesentlicher Bestandteil des OntoWiki, sobald mehrere verschiedenen Wissensbereichen aufeinander treffen und kombiniert werden müssen.
Eine etwaige Implementierung dieser Funktion arbeitet primär auf der Model-Komponente und kann weitestgehend entkoppelt von der View-Komponente entwickelt werden.

FOAF

Personendaten können in OntoWiki als FOAF-kompatible RDF-Daten abgespeichert werden.
Dadurch kann eine Syndikation von Personendaten zwischen verschiedenen Webseiten realisiert werden und der Informationsaustausch über Personen und deren Aktivitäten wird auf eine relativ einfache Art ermöglicht.