Anwendungen von Semantic MediaWiki - Forschungsaufgabenmanagement

Seminararbeit

von

David Kleinmann

Matrikelnummer: 1604408

An der Fakultät für Wirtschaftswissenschaften

Institut für Angewandte Informatik und

Formale Beschreibungsverfahren

Forschungsgruppe Wissensmanagement

Prof. Dr. Rudi Studer

Betreuender Mitarbeiter: Basil Ell

Eidesstattliche Erklärung

Ich versichere hiermit wahrheitsgemäß, die Arbeit und alle Teile daraus selbständig oder in Zusammenarbeit mit meinem Team angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

Karlsruhe, den 31. März 2014 *David Kleinmann*

Inhaltsverzeichnis

1 Motivation - SMW CorA **1**

2 Semantic MediaWiki **1**

3 Bereits vorhandene Erweiterungen **2**

4 Eine Aufgabe - Das Konzept **2**

5 Task Management - Die Erweiterung **3**

6 Momentaner Stand und Ausblick **10**

7 Fazit **11**

A Anhang **12**

A.1 Templates 12

A.2 Das Formular 16

1 Motivation - SMW CorA

Semantic MediaWiki for Collaborative Corpora Analysis, kurz SMW CorA, ist eine virtuelle Forschungsumgebung und verkörpert in diesem Seminar die Motivation für das Thema *Forschungsaufgabenmanagement*. Es ist, wie der Name bereits sagt, ein Semantic MediaWiki, welches für "die kollaborative Analyse von umfangreichen digitalisierten Textkorpora und deren exemplarische nachhaltige Einbettung in die Fachcommunity der Historischen Bildungsforschung"[1] genutzt wird. Die Forscher importieren digitale Versionen von historischen Bildungswerken, analysieren und annotieren sie. Dabei nutzen Sie die von MediaWiki und Seminatic MediaWiki zur Verfügung gestellten Möglichkeiten.

Mit der Zeit ist die Sammlung an Werken sehr stark angestiegen und zur Koordination ist eine Aufgabenverwaltung notwendig geworden. Mit ihrer Hilfe soll es möglich sein, die einzelnen Aufgabenschritte einfach zu administrieren, übersichtlich aufzulisten und diese Funktionen auch direkt in der Forschungsumgebung nutzen zu können. Da bereits Semantic MediaWiki, kurz SMW, als Basis der virtuellen Forschungsumgebung vorhanden war, entstand der Gedanke einer Aufgabenverwaltung, die eine Erweiterung des bereits vorhanden Systems ist und dabei ebenfalls die immensen Möglichkeiten der semantischen Webtechnologie nutzt.

In dieser Seminararbeit werde ich am Beispiel von SMW CorA zeigen, dass eine Erweiterung für Semantic MediaWiki ein geeignetes Werkzeug ist, um das Problem einer die Anforderungen erfüllenden Aufgabenverwaltung zu lösen. Durch SMW ist es sowohl sehr einfach die Daten zu strukturieren und zu verwalten als auch dem Endnutzer eine sehr benutzerfreundliche Oberfläche zu bieten. Das Konzept einer Aufgabe wurde nach den Vorstellungen der Forscher von SMW CorA erstellt und ihre Wünsche wurden ebenfalls bei der Implementierung berücksichtigt.

2 Semantic MediaWiki

Semantic MediaWiki verbindet eine etablierte Technologie mit einer aufstrebenden Technologie. MediaWiki ist eine freie Software, die sich innerhalb weniger Jahre zu einem der größten und wichtigsten Wissensmanagement-Tools entwickelt hat.

Sie ermöglicht es, dass (viele) Menschen die Inhalte kooperativ und kollaborativ benutzen und bearbeiten können. Durch die Integration der semantischen Webtechnologien in MediaWiki wird es möglich, dass auch ein Computer diese Inhalte analysieren und verknüpfen kann.

[1] http://www.aifb.kit.edu/web/SMW-CorA

Genau diese beiden Aspekte kommen bei diesem Seminar zu tragen.

3 Bereits vorhandene Erweiterungen

Im Bereich der Aufgabenverwaltung existieren bereits diverse Erweiterungen für die Software MediaWiki. Diese sind unterschiedlicher Art. Grob unterscheiden kann man in Erweiterungen, die externe Datenbanken mit dem MediaWiki verknüpfen, und Erweiterungen, die die Aufgaben in der Datenbank des MediaWikis speichern. Die externen Aufgabenverwaltungen sind von vornherein ausgeschieden, da die Aufgaben direkt in der MediaWiki Datenbank gespeichert werden sollten sowie auch nur so die Nutzung der semantischen Möglichkeiten realisierbar ist. Die restlichen Erweiterungen lieferten einige hilfreiche Ansätze, allerdings erfüllt leider keine von ihnen die Anforderungen, die von den Forschern von SMW CorA gestellt wurden. Insbesondere ist nur eine der Erweiterungen, Semantic Tasks[2] von Steren Giannini und Ryan Lane, für Semantic MediaWiki entwickelt; diese ist allerdings mittlerweile veraltet und wird nicht mehr gepflegt.

4 Eine Aufgabe - Das Konzept

Eine Aufgabe ist ein Objekt, das verschiedene Eigenschaften besitzt. Diese Definition wurde von mir nach den Wünschen der Forscher von SMW CorA aufgestellt.

Jede Aufgabe wird identifiziert durch einen Titel. Weitere Eigenschaften sind zur Informationsspeicherung vorgesehen, sie werden vom Benutzer festgelegt:

- Ersteller
- Bearbeiter
- Erstellungsdatum
- Erinnerungsdatum
- Fristsetzung
- Priorität

Zu beachten ist, dass eine Aufgabe mehrere Bearbeiter haben kann.
Ebenso verfügt jede Aufgabe über eine Beschreibung und Notizen.
Eine weitere wichtige Eigenschaft ist die “Entität“. Entitäten sind für die Aufgabe relevante Objekte beziehungsweise Seiten im MediaWiki. Eine Aufgabe kann beliebig viele

[2] https://www.mediawiki.org/wiki/Extension:Semantic_Tasks

Entitäten haben sowie eine Entität mit beliebig vielen Aufgaben verknüpft sein kann.

Des Weiteren ist jeder Aufgabe ein Objekt zugeordnet, welches Informationen über die Bearbeitung speichert. Es werden Wertepaare aus Benutzername und Entitätsname gespeichert. So ist es möglich nachzuvollziehen, welcher Bearbeiter für welche Entität die Aufgabe bereits erledigt hat.

Aus der Anzahl der Entitäten, der Anzahl der Bearbeiter und den Informationen über die bereits bearbeiteten Entitäten durch die jeweiligen Bearbeiter ist es möglich einen Prozentsatz für den Bearbeitungsfortschritt zu errechnen. Somit ist jede Aufgabe noch mit einem Prozentsatz und einem daraus abgeleiteten Status verknüpft. Der Status kann einen von drei Werten annehmen: *nicht begonnen*, *in Bearbeitung*, *fertiggestellt*.

5 Task Management - Die Erweiterung

Die Implementierung der von mir vorgelegten Erweiterung ist ein Zusammenspiel verschiedener Komponenten. Ein auf der Erweiterung *Semantic Forms*[3] von Yaron Koren und Stephan Gambke basierendes Formular sowie zwei Templates bilden den Kern der Erweiterung. Mittels diesen ist es möglich Aufgaben anzulegen, zu bearbeiten und auf den Seiten der Entitäten als erledigt zu markieren.

Diese Funktionalität wird erweitert durch mehrere PHP-Dateien. Sie erzeugen eine Spezialseite, die als Übersichtsseite dient. Ebenso basiert auf ihnen die Funktionalität für Erinnerungsmails und das automatische Einbinden beziehungsweise Entfernen des TaskBox-Templates, einer kleinen Box auf jeder Entitätsseite, die es ermöglicht, Aufgaben als erledigt zu markieren.

In den folgenden Abschnitten werde ich die einzelnen Bestandteile genauer erklären.

Eine Aufgabe im SMW

Die Daten jeder Aufgabe sind auf einer eigenen Seite gespeichert, identifiziert durch den Titel.
Ein Template übernimmt die Darstellung und führt kleinere Berechnungen zum Arbeitsstand der Aufgabe durch.
Abbildung 1 zeigt eine typische Aufgabenseite.

[3] http://www.mediawiki.org/wiki/Extension:Semantic_Forms

Abbildung 1: Eine Aufgabe

Das Template, von mir als Task-Template bezeichnet, ist im Anhang zu finden.

Attribute und weitere Daten

In Abschnitt 2 habe ich bereits das grundlegende Konzept einer Aufgabe vorgestellt. Für jede der dort vorgestellten Eigenschaften existiert ein semantisches Attribut im Semantic MediaWiki. Damit wird eine Aufgabe repräsentiert durch eine Seite im Wiki, die mit einer Reihe von semantischen Attributen verknüpft ist. In der folgenden Tabelle sind diese mit ihren jeweiligen Datentypen aufgelistet.

Attribut	**Datentyp**
Title	Text
Creator	Page
Assignee	Page
Created	Date
Reminder	Date
Deadline	Date
Priority	Number
Percent	Number
Status	Text
Entity	Page

Tabelle 1: Attribute mit jeweiligen Datentypen

Neben diesen Attributen existieren noch zwei weitere *versteckte* (Benutzer können sie nicht direkt bearbeiten) Attribute, die zum reibungslosen Funktionieren der Erweiterung beitragen.

Noassignee

Dieses Attribut ist auf den Wert "True" gesetzt, falls die Aufgabe von einem beliebigen Benutzer bearbeitet werden kann. In alle anderen Fällen wird es nicht gesetzt. Es dient der Erleichterung von Abfragen durch die TaskBox.

Subobject#did

In diesem Subobject wird gespeichert, welcher Benutzer für welche Entität die Aufgabe bereits erledigt hat. Man kann sich dies als einfache Tabelle vorstellen. Für jeden Bearbeiter existiert eine Zeile; in dieser Zeile stehen die bereits bearbeiteten Entitäten. Es wird durch das Task-Template angelegt und durch den Button in der TaskBox verändert.

Darstellung

Die Darstellung ist sehr übersichtlich gehalten und ist vollständig in Wiki-Syntax realisiert Die wichtigsten Informationen werden dem Benutzer in einer Tabelle präsentiert. Unterhalb der Tabelle befinden sich die Aufgabenbeschreibung sowie die Notizen, die Benutzer über das Formular eingeben können. In diesen beiden Texten kann normaler Wiki-Syntax verwendet werden.

Auf der rechten Seite findet man Informationen über den Bearbeitungsstatus der Aufgabe. Neben einer Angabe über den exakten Prozentsatz gibt es ein Ampelsymbol als visuelle Repräsentation. Sollte die Aufgabe komplett bearbeitet sein, ist eine grüne Ampel zu sehen. Falls die Bearbeitung noch nicht begonnen, wurde ist die Ampel rot. Für den Fall, dass die Aufgabe angefangen, aber noch nicht abgeschlossen wurde, ist eine gelbe Ampel zu sehen wie in Abbildung 1.

Anlegen neuer Aufgaben über ein Formular

Das Anlegen von Aufgaben wird über ein Formular ermöglicht. Dieses Formular basiert auf der Erweiterung *Semantic Forms*. Beim Speichern erzeugt es eine Seite, die alle relevanten Informationen einer Aufgabe enthält. Ebenso dient es dazu Aufgaben zu editieren. Dazu ist auf jeder Aufgabenseite ein Link vorhanden, mit dem das Formular mit den Daten der Aufgaben geladen wird.

Um eine neue Aufgabe zu definieren, muss zuerst eine Name vergeben werden. Da dieser

Name sowohl die Aufgabe über das Attribut “Title“ identifiziert als auch als Seitenname fungiert, muss dieser Name eindeutig sein und darf nicht mehrmals vergeben werden.
In der folgenden Tabelle sind alle Angaben aufgelistet, die beim Erstellen einer Aufgabe gemacht werden müssen beziehungsweise können:

Feld	**Kommentar**
Description	“Description“ ist ein reines Textfeld und wird nicht auf ein semantisches Attribut übertragen. Es dient lediglich zur Speicherung der Aufgabenbeschreibung.
Creator	Im Feld “Creator“ wird der Benutzername gespeichert, der die Aufgabe anlegt. Es wird von der Erweiterung mit dem Namen des eingeloggten Benutzer vorausgefüllt und wird auf das gleichnamige semantische Attribut übertragen.
Assignees	In dieses Feld werden die Bearbeiter der Aufgabe als durch Kommata getrennte Liste eingetragen. Jeder Bearbeiter wird über das Attribut “Assignee“ mit der Aufgabe verknüpft. Sollte es egal sein, wer die Aufgabe bearbeitet, muss in das Feld “ALL“ eingetragen werden.
Created	Dieses Feld enthält das Datum der Erstellung. Es wird automatisch von der Erweiterung ausgefüllt. Das Datum wird auf das gleichnamige semantische Attribut übertragen.
Reminder	Sollte es gewünscht sein, dass die Bearbeiter an einem bestimmten Tag an die Aufgabe erinnert werden, kann hier das entsprechende Datum eingetragen werden. Das Datum wird auf das gleichnamige semantische Attribut übertragen.
Deadline	In dieses Feld kann ein Datum für eine Frist eingetragen werden. An diesem Tag werden ebenfalls alle Bearbeiter per Email benachrichtigt. Das Datum wird auf das gleichnamige semantische Attribut übertragen.
Notes	“Notes“ ist ein reines Textfeld und wird nicht auf ein semantisches Attribut übertragen. Es dient lediglich zur Speicherung beliebiger Notizen, zum Beispiel zur Protokollierung von Zwischenständen.
Priority	Hier kann über ein Dropdown-Menü eine Priorität für die Aufgabe ausgewählt werden. Der Wert dient lediglich zur Sortierung von Aufgaben und wird auf das gleichnamige semantische Attribut übertragen.
Entities	Dieses Feld erwartet ein ask-Query der folgenden Form: {{#ask: [[Category:TESTCATEGORY]] [[Property::VALUE]] [[Property2::VALUE2]] |format=list |link=none |headers=hide |sep=, }} Über Kategorien und semantische Attribute kann die Menge der Entitäten spezifiziert werden. Die Resultate des Queries werden über das Attribut “Entity“ mit der Aufgabe verknüpft. Die Verknüpfung hat den Vorteil, dass die Liste der Entitäten dynamisch vom MediaWiki bestimmt wird und sich auch ändert, falls nach Anlegen der Aufgabe eine neue relevante Seite angelegt wird.

Tabelle 2: Die Felder des Formulars

Neben diesen Feldern enthält das Formular noch zwei versteckte Felder:

Title
: Dieses Feld dient dazu, den Aufgabennamen auf das gleichnamige semantische Attribut zu übertragen. Es wird von der Erweiterung automatisch ausgefüllt.

Dids
: Dieses Feld ermöglicht die Bearbeitung des Arbeitsstandes durch die #autoedit-Parserfunktion. Der Wert wird über das Task-Template auf das bereits erwähnte Subobject did übertragen.

Die TaskBox

Die TaskBox ist eine kleine Box und wird auf jeder Seite, die einer Aufgabe über das Attribut “Entity“ zugeordnet ist, angezeigt. Sie listet jede verknüpfte Aufgabe auf. Dabei wird der Titel angezeigt, welcher ebenso als Link zu der Aufgabe dient. Des Weiteren wird angezeigt, ob die Aufgabe für die Entität bereits bearbeitet wurde.

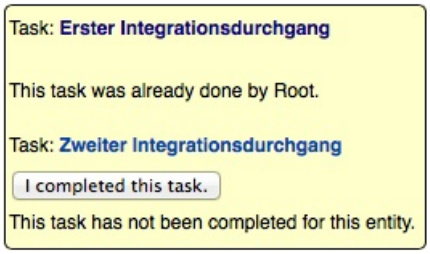

Abbildung 2: Die TaskBox

Falls der eingeloggte Benutzer als Bearbeiter der Aufgabe gelistet oder die Bearbeitung der Aufgabe frei für alle ist, wird ebenso ein Button angezeigt. Dieser Button dient dazu, die Aufgabe für diese Entität für diesen Benutzer als erledigt zu markieren.

Die Darstellung der TaskBox basiert auf einfachem HTML. Es wird lediglich ein div-Container benutzt, um die Elemente zu gruppieren und anzuordnen. Dadurch ist es sehr einfach, die Darstellung und die Positionierung direkt im Template den eigenen Wünschen anzupassen.

Für die Funktionalität werden verschiedene Parserfunktionen benutzt.

#arraymap
: Diese Funktion erlaubt es, einen Wert an einem Trennzeichen zu trennen, zum Beispiel eine per Kommata getrennte Liste. Jeder dieser Einzelwerte kann separat behandelt werden. In Kombination mit einem ask-Query fungiert die Parserfunktion als Schleife mit einer Variablen.

#ifeq

Die ifeq-Parserfunktion ermöglicht das Vergleichen zweier Strings miteinander. Falls beide Strings numerische Werte sind, werden die Zahlenwerte miteinander verglichen. Sie wird benutzt, um die verschiedenen möglichen Situation abzudecken. Zum Beispiel wird überprüft, ob der eingeloggte Benutzer als Bearbeiter für eine Aufgabe gelistet ist. Zum Beispiel muss die Anzahl der Ergebnisse eines dementsprechenden ask-Queries 1 sein.

#autoedit

Durch diese Parserfunktion wird das Editieren eines Formulars per Klick auf einen Button möglich. Beim Klicken des Buttons wird die Kombination aus Benutzer und Entität über auf der Aufgabenseite gespeichert.

Die Kombination der Parserfunktionen und ihre Verschachtelung innerhalb des Templates ermöglichen es, sämtliche Situationen abzudecken, ohne die Möglichkeiten von Wiki-Syntax zu verlassen.

Das Template für die TaskBox ist im Anhang zu finden.

Eine Spezialseite zur Übersicht

Zur Übersicht wurde eine Spezialseite implementiert.

Als Standardansicht werden sämtliche gespeicherten Aufgaben gelistet.

Über den GET-Parameter *user* lässt sich die Liste personalisieren. Falls der Parameter gesetzt ist, werden nur noch Aufgaben angezeigt, die mit dem spezifizierten Benutzer über das Attribut “Assignee“ verknüpft sind. Des Weiteren wird nun nicht mehr nur eine Tabelle angezeigt, sondern zwei Tabellen. Die erste Tabelle enthält alle nicht fertig gestellten Aufgaben und ermöglicht das schnelle Auffinden von momentan relevanten Aufgaben. Die zweite Tabelle hingegen listet alle Aufgaben für den Benutzer auf.

Natürlich ist es möglich eigene Versionen von Listen anzulegen, da über die Kategorie “Task(s)“ sämtliche Aufgaben erfasst und über die diversen semantischen Attribute gefiltert werden können. Hilfreich dabei ist die Erweiterung GetUserName[4] von Ejcaputo, die eine Parserfunktion hinzufügt und es so ermöglicht den Namen des eingeloggten Benutzers zu ermitteln.

[4]http://www.mediawiki.org/wiki/Extension:GetUserName

Zusätzliche Funktionalität

Um die Benutzerfreundlichkeit der Erweiterung zu erhöhen wurden die folgenden Features implementiert.

Erinnerungsmails

Die Erweiterung verschickt Erinnerungsmails an die Bearbeiter einer Aufgabe in den folgenden Fällen:

- Eine neue Aufgabe wurde angelegt
- Das Datum des Attributes “Reminder“ ist auf den heutigen Tag gesetzt
- Das Datum des Attributes “Deadline“ ist auf den heutigen Tag gesetzt

Der erste Punkt ist über einen Hook implementiert. Hooks erlauben es Entwicklern benutzerdefinierten Code an Stellen im eigentlichen MediaWiki-Code auszuführen. Eine Funktion, die einem Hook zugewiesen ist, wird ausgeführt, als würde sie direkt an der Programmstelle, an der der Hook definiert ist, stehen.

Wenn über das Formular der Erweiterung Semantic Forms eine neue Aufgabe angelegt wird, wird der Hook *sfWritePageData* aktiviert. Dieser ist mit der Funktion *notifyAssignees* der Klasse *TaskManagementUtils* verknüpft. Diese Funktion verschickt an jeden über das Attribut “Assignee“ mit der Aufgabe verknüpften Benutzer eine Email mit dem Titel und einem Link. Da zu dem Zeitpunkt des Funktionsaufrufes der Datensatz der Aufgabe noch nicht gespeichert ist, wird dies über die Zerlegung eines Strings, welcher den gesamten Seiteninhalt beinhaltet, realisiert.

Die anderen beiden Punkten sind über einen CronJob implementiert.
Täglich um 12 Uhr wird die Funktion *remindAssignees* der Klasse *TaskManagementUtils* aufgerufen. Mit Hilfe semantischer Abfragen an das MediaWiki werden die Aufgaben erfasst, die entweder das “Reminder“-Datum oder das “Deadline“-Datum auf den heutigen Tag gesetzt haben. Für jede dieser Aufgaben werden über weitere semantische Abfragen die Bearbeiter erfasst und dann jeweils mit einer Email benachrichtigt. Auch diese Emails enthalten nur einen kurzen Text mit dem Titel der Aufgabe und einem Link.

Sämtliche Emailtexte sind in der Sprachdatei der Erweiterung definiert und daher auch sehr einfach anpassbar.

Automatisches Einbinden/Entfernen des TaskBox-Templates

Auf jeder Entitätenseite ist die bereits beschriebene TaskBox zu finden. Die Anzeige der TaskBox ist über ein Template geregelt. Das Einbinden dieses Templates manuell durch den Benutzer wäre mit einem sehr hohen Aufwand verbunden. Ebenso das Entfernen des Templates würde das manuelle Aufrufen und Editieren jeder Seite bedeuten. Aus diesem Grund wurde eine automatische Lösung implementiert. Beim Anlegen beziehungsweise Editieren einer Aufgabe wird über den Hook *sfWritePageData* die Funktion *insertTaskBoxTemplate* der Klasse *TaskManagementUtils* aufgerufen. Dieser Funktion wird der Inhalt der Aufgabenseite übergeben, aus diesem wird das ask-Query gefiltert. Mit einer semantischen Abfrage bestimmt die Funktion die zu der Aufgabe gehörenden Entitäten und verändert jeweils den Seiteninhalt. Dabei wird am Anfang des Inhaltes der String "{{TaskBox}}" hinzugefügt, somit wird das Template auf jeder Entitätenseite automatisch eingebunden. Das automatisierte Entfernen geschieht wie die Erinnerungsmails per CronJob. Dazu wird die Funktion *deleteUnneededTaskBoxTemplates* der Klasse *TaskManagementUtils* aufgerufen. Über eine semantische Abfrage werden alle Seiten gefiltert, die der Kategorie "Page with Task" angehören. Diese Kategorie wird über die TaskBox gesteuert. Nachfolgend wird für jede dieser Seiten eine semantische Abfrage durchgeführt, die nach Aufgaben sucht, welche über das Attribut "Entity" mit der Seite verknüpft sind. Falls die Anzahl für diese Abfrage 0 ist, wird das Template aus dem Inhalt der Seite entfernt.

6 Momentaner Stand und Ausblick

Die von mir entwickelte Erweiterung habe ich auf mediawiki.org[5] veröffentlicht. Sie ist unter der GPLv2-Lizenz verfügbar und kann damit frei genutzt werden.

Ich plane die Erweiterung weiter zu entwickeln und kontinuierlich zu verbessern.
Als wichtigste Verbesserung erscheint mir die Implementierung verschiedener Aufgabentypen. Bis jetzt werden lediglich einfache Aufgaben unterstützt und sämtliche Besonderheiten können lediglich in der Aufgabenbeschreibung festgehalten werden. Ein Beispiel für komplexere Typen wäre Aufgaben mit mehreren Schritten, die nacheinander bearbeitet werden müssten, oder Aufgaben, die erfordern, dass der Bearbeiter einen Rückgabewert einträgt.
Den verschiedenen Möglichkeiten sind hier sicherlich keine Grenzen gesetzt.
Des Weiteren arbeite ich bereits an Filter- und Sortierungsoptionen für die Übersichtsseite.

[5] http://www.mediawiki.org/wiki/Extension:TaskManagement

7 Fazit

In der Arbeit konnte gezeigt werden, dass eine Erweiterung für Semantic MediaWiki ein geeignetes Werkzeug ist, um eine Aufgabenverwaltung zu implementieren.

Der Modellierung von Aufgaben sind dank der offenen Struktur von Semantic MediaWiki keine Grenzen gesetzt.
Eine weitere Stärke zeigt sich eindeutig in der Erweiterbarkeit von MediaWikis. Den Möglichkeiten hierbei sind ebenfalls kaum Grenzen gesetzt. Die vorhandene Dokumentation ist sehr umfangreich und detailliert und man findet sehr schnell Hilfe.

Probleme bereit haben mir vor allem die Parserfunktionen. Die sind zum Teil sehr spezifisch; zum Beispiel funktionieren manche nur mit bestimmten Datentypen. Genau an diesem Punkt versagt leider auch die Dokumentation. Beide Einzelbestandteile, sprich Parserfunktionen des MediaWikis und Datentypen des Semantic MediaWikis sind für sich selber gut dokumentiert, aber für die Verknüpfung von Bestandteilen der beiden Projekte ist kaum etwas Hilfreiches zu finden.

Insgesamt ist es eine Erweiterung ein einfaches und leicht benutzbares Instrument für eine Aufgabenverwaltung.

A Anhang

A.1 Templates

Das Task-Template

Das Task-Template wird für die Darstellung jeder Aufgabe auf ihrer Seite benutzt. Neben der reinen Darstellung führt es kleinere Berechnungen für den Bearbeitungsstand der Aufgabe durch.

```
<noinclude>
This is the "Task" template.
</noinclude>
<includeonly>
<!-- create subobject for tracking status -->
{{#arraymap:{{{dids|}}}|;|x| {{#subobject:did |x }} |}}
<!-- if task is open for all set specific property -->
{{#ifeq:{{{assignees|}}}|ALL| {{#set: noassignee=true}} |  }}
<!-- calculate how much of the task is done -->
{{#set: percent=
{{#expr: ({{#arraymap:{{{assignees|}}}|,|x|
{{#ask: [[-x::{{FULLPAGENAME}}#did]] | mainlabel=- | headers=hide |
format=count}} |+}}) / ({{#ask:[[-assignee::{{PAGENAME}}]]|
format=count}}*{{#ask:[[-entity::{{PAGENAME}}]]|format=count}}) }} }}
<!-- show percentage of completion verbally and visually -->
<div style="float: right; clear: both; margin: 1em; padding: 2px;
width: auto; text-align: left; font-size: 90%; line-height: 1.4em;">
This task is done to {{#expr: {{#ask:[[{{PAGENAME}}]]| ?percent |
headers=hide |mainlabel=-}} *100}}%.
{{#ifeq:
{{#expr: {{#ask:[[{{PAGENAME}}]]| ?percent | headers=hide |mainlabel=-}}
}}| 1 |[[File:Ampel_gruen.png]]{{#set: status=Completed}} |
{{#ifeq:
{{#expr: {{#ask:[[{{PAGENAME}}]]| ?percent | headers=hide |mainlabel=-}}
}}| 0 | [[File:Ampel_rot.png]]{{#set: status=Not started}} |
[[File:Ampel_gelb.png]]{{#set: status=In progress}} }} }}
</div>
{| class="wikitable"
! Title
| [[Title::{{{title|}}}]]
|-
```

```
! Creator
| [[Creator::{{{creator|}}}]]
|-
! Assignees
| {{#arraymap:{{{assignees|}}}|,|x|[[assignee::x]]|,&#32;}}
|-
! Status
| {{#ask:[[{{PAGENAME}}]]|?status|headers=hide |mainlabel=-}}
|-
! Created
| [[Created::{{{created|}}}]]
|-
! Reminder
| [[Reminder::{{{reminder|}}}]]
|-
! Deadline
| [[Deadline::{{{deadline|}}}]]
|-
! Priority
| [[Priority::{{{priority|}}}]]
|-
! Entities
|  {{#arraymap: {{{entities|}}}|,|x|[[entity::x]]|,&#32;}}
|}
{{#formlink:form=Task|target={{FULLPAGENAME}}|link text=Edit this task|link

==Description==
{{{description|}}}

==Notes==
{{{notes|}}}

<br >
A list of all tasks: [[Special:Tasks]]
[[Category:Task(s)]]
</includeonly>
```

Das TaskBox-Template

Das TaskBox-Template erzeugt eine kleine Box, die es ermöglicht Aufgaben als erledigt zu markieren. Ebenso zeigt es Informationen über die Aufgabe an.

```
<noinclude>
This is the "TaskBox" template.
</noinclude>
<includeonly>
<div style="float: right; clear: both; margin: 1em; padding: 2px;
width: auto; text-align: left; font-size: 90%; line-height: 1.4em;
border:1px solid; border-radius:5px; -moz-border-radius:5px;
background-color: #FFFFCC;">
{{#arraymap:
{{#ask: [[Category:Task(s)]] [[entity::~*{{FULLPAGENAME}}*]]| ?title |
format=list | mainlabel=- | headers=hide | link=none | sep=,}}|,|@@@|
Task: '''{{#ask: [[title::@@@]]}}'''<br>
<!-- check if task is assigned to user -->
{{#ifeq: {{#ask: [[title::@@@]] [[assignee::{{#USERNAME:}}]] |
format=count}}|1|
<!-- check if user has already finished the task -->
{{#ifeq: {{#ask: [[-Has subobject::@@@]]
[[{{#USERNAME:}}::~*{{FULLPAGENAME}}*]] |format=count}}|0|
<!-- create autoedit button -->
{{#autoedit:form=Task|target={{#ask: [[title::@@@]] | link=none}}|
link type=button|link text=I completed this task.|query string=
Task[dids]={{#USERNAME:}}={{FULLPAGENAME}};
{{#arraymap:{{#ask: [[title::@@@]]| ?assignee | format=list |
mainlabel=- | headers=hide | link=none | sep=,}}
|,|x| {{#arraymap:{{#ask: [[-Has subobject::@@@]] | ?x |
format=list | mainlabel=- | headers=hide | link=none | sep=,}}|,|y|x=y;|}}
|}}}} <!-- /autoedit -->| }}
<!-- /ifeq -->| }}
<!-- /ifeq -->
<!-- check if task can be done by anyone -->
{{#ifeq: {{#ask: [[title::@@@]] [[noassignee::true]]| format=count}}|1|
<!-- check if task is already done -->
{{#ifeq: {{#ask: [[-Has subobject::@@@]] [[ALL::~*{{FULLPAGENAME}}*]] |
format=count}}|0|
<!-- create autoedit button -->
{{#autoedit:form=Task|target={{#ask: [[title::@@@]] | link=none}}|
```

```
link type=button|link text=I completed this task.|
query string=Task[dids]=ALL={{FULLPAGENAME}};
{{#arraymap:{{#ask: [[-Has subobject::@@@]] | ?ALL | format=list |
mainlabel=- | headers=hide | link=none | sep=,}}|,|y|ALL=y;|}} }}
<!-- /autoedit -->| }}
<!-- /ifeq -->| }}
<!-- /ifeq -->
<!-- check if task is open for all or user specific -->
{{#ifeq:{{#ask: [[title::@@@]] |?Noassignee |mainlabel=- | headers=hide |
link=none }}|True|
<!-- open for all, display if done or not -->
{{#ifeq: {{#ask: [[-Has subobject::@@@]] [[ALL::~*{{FULLPAGENAME}}*]] |
format=count}} | 1 |
This task has been completed for this entity.|
This task has not been completed for this entity. }} |
<!-- user specific, display who has done the task -->
{{#ifeq:
<!-- check which assignee has done this task and sum the count -->
{{#expr: {{#arraymap: {{#ask: [[title::@@@]] |?assignee| format=list |
mainlabel=- | headers=hide | link=none | sep=, | default=}}|,|x|
{{#ifeq:
{{#ask: [[-Has subobject::@@@]] [[x::~*{{FULLPAGENAME}}*]] |format=count}}
|0|0|1 }} |+}}
<!-- /expr -->
}}| 0 | This task has not been completed for this entity.|
This task was already done by
{{#arraymap:{{#ask: [[title::@@@]] |?assignee|format=list|
mainlabel=- | headers=hide | link=none | sep=, | default=}}|,|x|
{{#ifeq: {{#ask: [[-Has subobject::@@@]] [[x::~*{{FULLPAGENAME}}*]] |
format=count}}|0| |x }} |,&#32;}}.}}
<!-- /ifeq --> <br> }}
<!-- /ifeq --> |<br>}}
</div>
[[Category:Page with Task]]
</includeonly>
```

A.2 Das Formular

Der nachfolgende Code erzeugt das von mir für die Erweiterung verwendete Formular.

```
<noinclude>
This is the "Task" form.
To create a page with this form, enter the page name below;
if a page with that name already exists, you will be sent to
a form to edit that page.


{{#forminput:form=Task}}

</noinclude><includeonly>
<div id="wikiPreview" style="display: none; padding-bottom: 25px;
margin-bottom: 25px; border-bottom: 1px solid #AAAAAA;"></div>
{{{for template|Task}}}
{| class="formtable"
! Description:
| {{{field|description|input type=textarea|rows=8|cols=50|
property=description}}}
|-
! Creator:
| {{{field|creator|mandatory|property=creator|default=current user}}}
|-
! Assignees:
| {{{field|assignees|property=assignees}}}
|-
! Created:
| {{{field|created|mandatory|property=created|input type=date|
default=now}}}
|-
! Reminder:
| {{{field|reminder|property=reminder|input type=date}}}
|-
! Deadline:
| {{{field|deadline|property=deadline|input type=date}}}
|-
! Notes:
| {{{field|notes|property=notes|input type=textarea|rows=8|cols=50}}}
|-
```

```
! Priority:
| {{{field|priority|mandatory|property=priority}}}
|-
! Entities:
| {{{field|entities|mandatory|property=entities}}}
{{{field|dids|hidden}}}
{{{field|title|hidden|property=title|default={{FULLPAGENAME}} }}}
|}
{{{end template}}}

{{{standard input|save}}} {{{standard input|preview}}}
{{{standard input|cancel}}}
</includeonly>
```