Inhaltsverzeichnis

Einrichten eines SemanticWiki mit 4Store-DB

Neuen User anlegen

```
adduser semantic
```

Apache Location in das Home- Dir des semantic user legen

/etc/apache2/conf.d/mediawiki.conf

```
Alias /smw /home/semantic/mediawiki

<Directory /home/semantic/mediawiki/>
        Options +FollowSymLinks
        AllowOverride All
        order allow,deny
        allow from all
</Directory>
```

User semantic in MySql anlegen mit gleichnamiger DB und allen Rechten auf dieser DB (z.B. mit phpMyAdmin)

MediaWiki installieren

```
cd /home/semantic/
wget http://download.wikimedia.org/mediawiki/1.22/mediawiki-1.22.2.tar.gz
tar -xvzf mediawiki-1.22.2.tar.gz
```

Symbolischen Link auf das ausgepackte Verzeichnis

```
ln -s mediawiki-1.22.2 mediawiki
```

Rechte auf den www- user ändern

```
chown -R www-data:www-data mediawiki-1.22.2
```

apache neu starten

```
service apache2 restart
```

Im Browser die MediaWiki- Konfiguration starten: http://129.103.38.60/smw/ und durcharbeiten

Das war MediaWiki.

Nun kommt Semantic Media Wiki:

Installieren von php curl

```
apt-get install php5-curl
service apache2 restart
```

Installieren des Composers (den Proxy- Parameter braucht man nur hinter einer Firewall)

```
curl --proxy xx.yy.zz.85:8080 -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

Laufenlassen des Composers im Mediawiki- Basisdirectory

```
composer require mediawiki/semantic-media-wiki "~1.9"
php maintenance/update.php
```

Den Aufruf von `enableSemantics()` ans Ende der `LocalSettings.php` Datei hängen. `enableSemantics()` nimmt als Parameter den Domain Namen des Wiki; ein Wiki auf „example.org“, hätte zum Beispiel den folgenden Aufruf:

```
enableSemantics( 'example.org' );
```

Testen der SMW Umgebung wie auf http://semantic-mediawiki.org/wiki/Help:Testing beschrieben

Installieren von 4Store

```
apt-get install 4store
```

DB- Basisverzeichnisse anlegen

```
mkdir /var/lib/4store /var/log/4store
```

`LocalSettings.php` am Ende ergänzen um:

```
$smwgDefaultStore = 'SMWSparqlStore';
$smwgSparqlDatabase = 'SMWSparqlDatabase4Store';                    # using
4Store as connector
$smwgSparqlQueryEndpoint = 'http://localhost:8000/sparql/';         #
location of query service
$smwgSparqlUpdateEndpoint = 'http://localhost:8000/update/';        #
location of update service
$smwgSparqlDataEndpoint = '';                                       #
location of SPARQL over HTTP service
                                                                     #
optional value; leave as is in case of problems
$smwgSparqlDefaultGraph = 'http://example.org/mydefaultgraphname';  #
optional name of default graph
```

Sourcecode patchen:

In `extensions/SemanticMediaWiki/includes/sparql/SMW_SparqlDatabase.php` in line

457 und 493, ändere

```
curl_setopt( $this->m_curlhandle, CURLOPT_HTTPHEADER, array('Content-Type: application/x-www-form-urlencoded;charset=UTF-8'));
```

auf

```
curl_setopt( $this->m_curlhandle, CURLOPT_HTTPHEADER, array('Content-Type: application/x-www-form-urlencoded'));
```

Leere Datenbank erzeugen

```
4s-backend-setup oobd
```

DB- Process starten

```
4s-backend oobd
```

http-Server starten

```
4s-httpd -p 8000 oobd
```

Die alten (System-) Datenbankeinträge in die neue RDF Datenbank transferieren

```
cd  extensions/SemanticMediaWiki/maintenance/
php SMW_refreshData.php
```

jetzt sollte es gehen ...

und wenn man die DB Prozesse wieder stoppen will

```
pkill -f '^4s-httpd -p 8000 oobd$'
pkill -f '^4s-backend-setup oobd$'
```

Finetuning

Hier nun die vermutet notwendigen Modifikationen, hauptsächlich abgeguckt aus http://www.amazon.com/Working-MediaWiki-Yaron-Koren/dp/0615720307 (vom Betreiber der Referata- SMW- Website)

Noch bestehende Unklarheiten

Wie genau legt man die noch fehlende Gruppe der Guardians an und verteilt die richtigen Rechte an Guardians und normale User?

Hintergrund: Per Default kennt Mediawiki nur drei Usergruppen: sysadmins, bureaucrats und user. sysadmins dürfen alles, bureaucrats dürfen User und Seiten verwalten und user dürfen alle Seiten

editieren. Und da liegt der Hund begraben: Wir müssen die Rechte der User darauf beschränken, nur die Seiten zu verändern, die die eigentlichen Daten beinhalten (das soll dann ein eigener Namespace OOBD werden). Die logische Beschreibung der dahinterliegenden Syntax (und die Hauptseite usw.) muss aber uns überlassen bleiben, sonst erfindet jeder User seine eigene Syntax und wir haben ein völlig inkompatibles Kuddelmuddel.

Funktioniert InviteSignup auch bei OpenID?

Sonstiges

Schickes Logo pinseln und mit Schriftzug versehen

Wie macht man am besten Updates?

- Mit git aktualisieren
- Update.php im Verzeichnis /maintenance aufrufen

Erzeugen des OOBD namespace

Aus https://www.mediawiki.org/Wiki/Extension_namespace_registration zwei noch freie aufeinander folgende Nummern auswählen (Im Beispiel unten wurde dann 500 verwendet), gerade Nummer für Namespace, darauffolgende ungerade Nummer für namespace:Talk

Dabei vorher im Buch auf Pos 973 den Hinweis beachten, wenn man Seiten schon mit Namespace benannt hat, ohne das der Namespace schon existierte und die heutigen paar Seiten entprechend wegräumen

Änderungen in der LocalSettings.php

Pfad für Logo setzen

```
$wgLogo="Pfad oder URL";
```

Erlaubt böse Revisionen einer Seite komplett aus dem System zu löschen (normale Löschoperation macht Seiten nur unsichtbar)

```
$wgGroupPermissions['sysop']['deleterevision'] = true;
```

Erzeugen einer Gruppe, die alle Seiten bearbeiten kann, während normale User nur noch OOBD Seiten bearbeiten können sollen. Das alleinige Vergeben eines Rechtes an eine bislang nicht existierende Gruppe reicht schon, um diese anzulegen.

```
$wgGroupPermissions['guardians']['read'] = true;
```

Gibt der Gruppe Seitenlöschrechte

```
$wgGroupPermissions['guardians']['delete'] = true;
$wgGroupPermissions['guardians']['undelete'] = true;
```

Erzeugen des OOBD namespace:

```
if (!defined($wgExtraNamespaces)){
    $wgExtraNamespaces=array();
}
#3000+ is, accourding to MediaWiki, actual reserved for site custom
namespaces, so we choose
define("NS_OOBD"      , 3000 + 0x00BD     );
define("NS_OOBD_TALK" , 3000 + 0x00BD + 1  );
$wgExtraNamespaces[NS_OOBD] = "OOBD";
$wgExtraNamespaces[NS_OOBD_TALK] = "OOBD_Discussion";
```

Wenn wir InvideSignup einsetzen und dies zusammen mit openID funktioniert, verhindern wir hiermit, daß ein User sich ohne Anmeldung einen Account basteln kann

```
# Prevent new user registrations except by sysops
$wgGroupPermissions['*']['createaccount'] = false;
```

Notwendige Extensions installieren

1. Extension LiquidThreads für Seitendiskussionen
2. Lockdown oder SemanticAcl für Write Access Control (Lockdown scheint mir geeigneter, weil wir ja nur nach Namespace kontrollieren wollen, da wäre ACL vielleicht etwas zuviel des Guten)
3. google_custom_search_engine für interne Google- Suche (hat den erwünschten Seiteneffekt,

   das Google die Seiten dann auch gleich mit im Fokus hat )
4. Data_Transfer für XML Import

   ```
   $wgGroupPermissions['guardians']['datatransferimport'] = true;
   ```

5. irgendwas Passendes für Google Adsense, denn irgendwie muß der Server ja auch bezahlt werden..
6. InviteSignup, um Neuanmeldungen nur über Einladung laufen zu lassen.

   ```
   $wgGroupPermissions['user']['invitesignup'] = true;
   ```