Semantic MediaWiki quick reference (1 of 3)

Covers MediaWiki 1.19 and MediaWiki extensions ParserFunctions 1.4, Semantic MediaWiki 1.8, Semantic Maps 2.0, Semantic Internal Objects 0.6, Semantic Compound Queries 0.3 and Semantic Forms 2.5

MediaWiki syntax

"italics" **"bold"** **"bold and italics"**
==standard header==
[[Internal link]]
[[Internal link|alternate text]]
[http://example.com text of external link]
[[Category:Example]] (category tag)

--- (horizontal line)

!= (greater), !~ (like), != (not), !~
:: (equals), < (less)

Comparators:
-> (greater), <= (less than), => (greater), == (equal), != (not equal)

Special pages: Types, CreateProperty, Properties, UnusedProperties, Browse, SearchByProperty

Defining a triple:

[[property-name::value]]

( silent definition)

Defining a property:

[Has type::type-name]

( [Allows value::enum-value] for enums)

Property types: Page (default), String, Text, Code, Number, Date, Boolean, URL, Email, Telephone number, Temperature; from Semantic Maps: Geographic coordinate

Defining an internal object:

{{#set_internal:object-to-page-property|main-prop=|prop1=val1|...|prop2=list=val2[...]]}}

Example (on a page called "United States"):

{{#set_internal:Is president of|Has name=James Madison|Has start year=1801|Has end year=1809|Has vice president#list=George Clinton, Elbridge Gerry}}

Notable settings for LocalSettings.php

$smwgShowFactbox - set to false to hide SMW fact box

$smwgShowInlinelinks - set to true to hide wiki-links within property values

$smwgShowFactbox - set to false to hide SMW fact box

$smwgShowInlinelinks - set to true to hide wiki-links within property values

Semantic Compound Queries

Running a compound query:

{{compound_query|query1|query2|...|joint parameters}}

Each query’s parameters should be separated by semicolons, instead of pipes.

Sample compound query: to show a map with one icon for clothing stores and another for fast food restaurants:

{{compound_query|[Category:Clothing stores]|Has coordinates;icon=Shirt.png|[Category:Restaurants]|Has coordinates;icon=Hamburger.png|format=googlemaps|height=400|width=600}}

Notable settings for LocalSettings.php

$smwgQMaxInlineLimit - maximum results displayed on a page; default is 50

$smwgQMaxInlineLimit - maximum results displayed on a page; default is 50

$smwgQMaxLimit - maximum results retrieved for a query; default is 10000

<table>
<thead>
<tr>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining a concept: In the 'Concept' namespace:</td>
</tr>
<tr>
<td>{{#concept:query-conditions</td>
</tr>
</tbody>
</table>
Semantic Drilldown

Special pages: CreateFilter, Filters, BrowseData

Defining a filter:

- [[Covers property::property-name]]
- [[Has value::value]]
- [[Use time period::[Year][Month]]]
- [[Gets values from category::category-name]]
- [[Has input type::[combo box][date range]]]
- [[Requires filter::Filter.filter-name]]
- [[Has label::label]]

In a category page:

- [[Has filter::Filter.filter-name]]
- [[Has display parameters::parameters]]
- "parameters" should have the same structure as #ask query parameters, but separated by semicolons

--- HIDEFROMDRILLDOWN ---
--- SHOWFROMDRILLDOWN ---

Notable settings for LocalSettings.php

$sdgFiltersSmallestFontSize, $sdgFiltersLargestFontSize - font sizes for "tag cloud" display of filter values

Semantic Maps

Standard formats:

- googlemaps - maps, using Google Maps service
- openlayers - maps, using OpenLayers service
- map - maps, using the default service specified

Notable params: center, geoservice, width, height, zoom, icon, lines, polygon

Export formats:

- kml - KML (Key Markup Language)

Maps

Displaying a map:

```
{{#display_map:points=|center=|service=|geoservice=|width=|height=|zoom=|icon=|lines=|polygons=}}
```

Format for "points" parameter:

```
point 1~title 1~caption 1~marker 1;point 2~title 2~caption 2~marker 2
```

Geocoding (displays coordinates):

```
{{#geocode:location=string or coordinates|service=|format=|directional=}}
```

Semantic Result Formats

Date formats:

- timeline - scrolling JavaScript-based timeline
- eventline - like timeline, but allows multiple points per event
- calendar - monthly calendar
- Use #calendarstartdate and #calendarenddate to limit results to current month
- eventcalendar - like calendar, but uses FullCalendar JS library
- timeseries - aggregates pages by date, using the float JS library

Graphing formats:

- graph - graph of pages as nodes, using the VizGraph application
- process - workflow graphs, using VizGraph

Charting formats:

- googlebar - bar chart, using the Google API
- googlepie - pie chart, using the Google API
- jqplotbar - bar chart, using the jqPlot JavaScript library
- jqplotpie - pie chart, using jqPlot
- jqplotchart - various chart types, using jqPlot
- jqplotseries - like jqplotchart, for multiple sets of data

Runtime param: charttype (can be bar, line, donut, bubble or scatter)

- d3chart - various chart types, using the D3 JS library
- Relevant param: charttype (can be bubble or treemap)

- sparkline - small, inline charts, using the jquery.sparkline JS library
- Relevant param: charttype (can be bar, line, pie or discrete)

- dygraphs - charts of non-semantic raw data, with semantic annotations
- Relevant param: datasource
- jqPlot, D3 and dygraphs formats allow the "height", "width", "charttitle" and "charttext" params, among others.
- jqPlot, D3 and sparklines allow "distribution".

Page display formats:

- gallery - display of images in a gallery or slideshow
- listwidget - results separated into pages based on alphabetical and other sorting
- Relevant params: widget (can be alphabet, menu or pagination), pageitems
- pagewidget - scrolling through result pages

Math formats:

- sum, average, min, max, median, product - operations on numbers
- For all these formats, "limit" parameter must be greater than number of queried pages, for accurate results.

Other display formats:

- outline - hierarchical display of properties
- Relevant param: outlineproperties
- tree - shows pages connected via a property
- otree - like tree, but a bulleted list
- tagcloud - tag cloud display, based on frequency of values
- valuerrank - like tagcloud, but shows # of instances, without changing font size
- filtered - JavaScript-based property drilldown
- array - like list, but allows more delimiters
- hash - like array, but meant to be used by HashTables extension
- incoming - shows all incoming properties to each page

Export formats:

- bibtext - BibTeX bibliographic format
- icalendar - iCalendar event information format
- vcard - vCard personal information format
- feed - both the RSS and Atom syndication types
Semantic MediaWiki quick reference (3 of 3)

Covers MediaWiki extensions Semantic Forms 2.5 and External Data 1.3

**Semantic Forms**

Special pages: CreateForm, Forms, CreateClass, RunQuery

Simple form definition:

```markdown
{{for template|template-name}}
|field-label: {{(field|field-name)}}|
|end template}}
```

Free text: `{{{standard input|free text}}}`

Form tags: info field for template standard input end template

'info' parameters:
- create title=
- edit title=
- query title=
- field name=
- field type=
- form name=
- form type=
- form name=
- template name=
- query string=

'for template' parameters:
- label=
- multiple
- add button text=
- embed in field=
- allow modify=
- allow edit=
- allow save=
- show on form=
- show on page=
- show on list=
- show on category=
- show on concept=
- show on namespace=

'field' parameters:
- input type=
- hidden= class=
- mandatory
- list= Restricted
- delimiter= Default
- property= Preload
- holds template
- number of columns=
- size= Uploadable
- image preview
- preloads= Remote
- default values= Remote
- allow overwrite=
- allow insert=
- allow edit=
- allow remove=
- allow save=

'for enums and autocomplete':
- values= Values from property category|concept|namespace
- from= Source
- to= Destination
- default= Default value

'for enews and autocomplete':
- values= Values from enews category|concept|namespace
- from= Source
- to= Destination
- default= Default value

'standard input' parameters:
- label=
- class=

Input types

<table>
<thead>
<tr>
<th>Field type</th>
<th>Allowed input types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>text with autocomplete, combobox, text</td>
</tr>
<tr>
<td>String, URL, Number, etc.</td>
<td>text, text with autocomplete, combobox</td>
</tr>
<tr>
<td>Text, Code</td>
<td>textarea, text</td>
</tr>
<tr>
<td>Date</td>
<td>date, datetime, year</td>
</tr>
<tr>
<td>Enumeration</td>
<td>dropdown, radiobutton</td>
</tr>
<tr>
<td>Boolean</td>
<td>checkbox</td>
</tr>
<tr>
<td>List of Page</td>
<td>text with autocomplete, text, textarea with autocomplete</td>
</tr>
<tr>
<td>List of String, etc.</td>
<td>text, text with autocomplete, textarea with autocomplete</td>
</tr>
<tr>
<td>List of enums</td>
<td>checkboxes, listbox</td>
</tr>
</tbody>
</table>

More input types:
- from Semantic Maps: map, googlemaps, openlayers;
- from Semantic Forms Inputs: datepicker, timepicker, datetimepicker, menuselect, regexpat, from Semantic Image Input: instantaneous;
- using CategoryTree:

Forms that set an automatic page name

In form definition:

```markdown
|{{for template|template-name}}|field-name> ...
|<template-name|
```

Linking to the form:

```markdown
|{{for linkform|form=|link text=|link type=|query string=|popup|autocomplete on|
```

Linking to Special:RunQuery:

```markdown
|{{for queryform|form=|link text=|link type=|query string=|popup|autocomplete on|
```

Pointing red links to forms:
- (in a property or namespace page)
- (in a property page)
- (in a namespace page)

Making red links create pages automatically:
- (in a property page)
- (in a namespace page)

A link to edit a page automatically:
- (in a property page)
- (in a namespace page)

Notable settings for LocalSettings.php

- $wgRenameEditTabs - set true to change 'edit with form' to 'edit', 'edit' to 'edit source' $wgRenameMainEditTab - set to true to change 'edit to 'edit source' $wgAmericanDates - set to true to show month first in date input $wg24HourTime - set to true to display time input in 24-hour notation

**External Data**

**Getting data from a URL:**

```markdown
|{{get_by_url|format=|data=mappings|filters=|where=|query=|popup|autocomplete on|
```

Allowed formats: CSV, JSON, XML, GFF

**Getting data from a database:**

```markdown
|{{get_db_data|server=|from=|where=|query=|popup|autocomplete on|
```

**Getting data from an LDAP server:**

```markdown
|{{get_ldap_data|domain=|filter=|data=mappings|format=|from=|where=|query=|popup|autocomplete on|
```

"data", "filters" values should be comma-separated.

**Displaying external data:**

```markdown
|{{external_value|variable_name=|expression=|popup|autocomplete on|
```

The expression value in #for_external_table should contain variables of the form `{{variable_name}}`.

**Storing external data - examples:**

Storing an external value with SWM:
- The capital of {{PAGENAME}} is {{Has capital::|external_value::capital|}}

Storing a table of external values (requires Semantic Internal Objects extension):
- (in a namespace page)
- (in a property page)

Notable settings for LocalSettings.php

- $sgdYggStringReplacements - array for hiding API keys
- $sgdAllowExternalDataFrom - array for an API "whitelist" - necessary if API keys are being hidden
- $sgdCacheTable - database table for caching data
- $sgdDB and LDAP access have other required settings.

Last updated January 22, 2013. Created by Yaron Koren; inspired by the "cheat sheets" at addedbytes.com.

Distributed under the Creative Commons by-3.0 license.