Semantisches Wissensmanagement im Unternehmen: Konzepte, Technologien, Anwendungen

Prof. Dr. Stefan Linus Zander

Kapitel 4: Fortschrittliche Ontologiemodellierung mittels Semantic MediaWiki

Inhalte

1. Page Forms
2. Query Forms
3. Fortgeschrittene Tipps
    i. common.css
    ii. ...

Page Forms

Ausgangsfrage

Wie kann man FachanwenderInnen darin unterstützen, semantisch-konforme und als Wissensgraph verarbeitbare Inhalte zu erzeugen ?

Page Forms

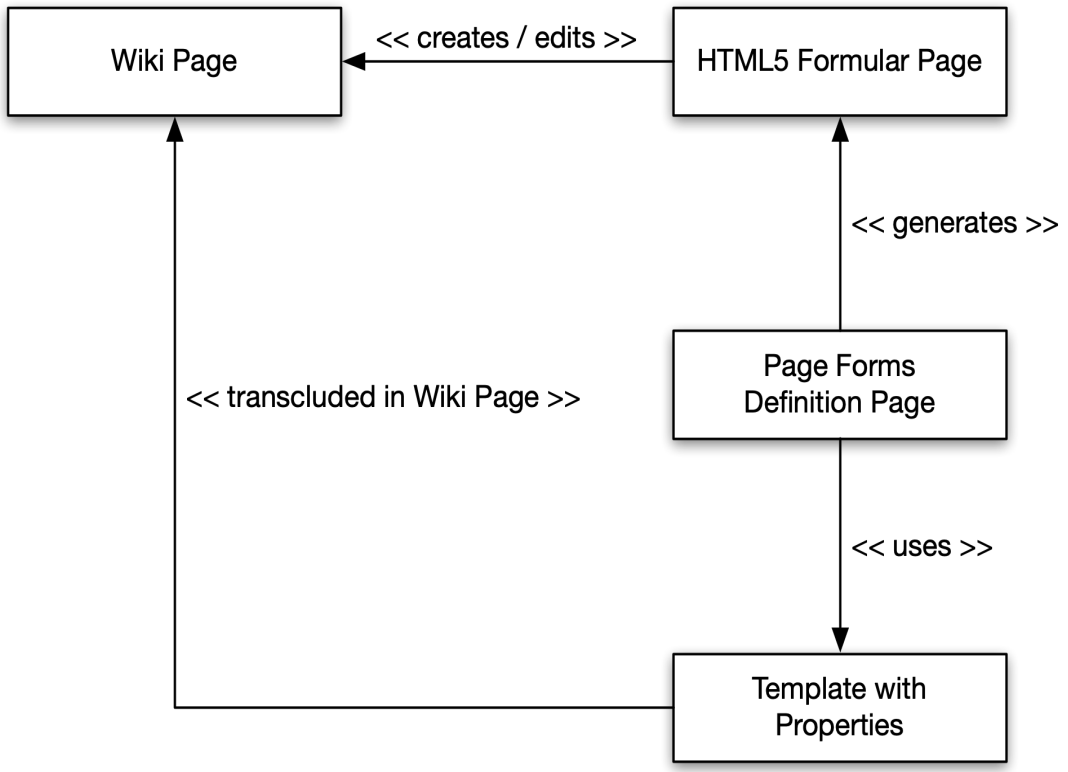
Page Forms (aka Semantic Forms bis 2016) ist eine Erweiterung für MediaWiki-Systeme.

Sie ermöglicht NutzerInnen durch den Einsatz von **HTML5-Formularelementen**...

- neue Inhalte zu erstellen
- bestehende Inhalte zu modifizieren
- Abfragen dynamisch zu erzeugen und auszuführen (--> Query Forms)

Page Forms setzen auf semantische Templates und erlauben **Parameterwerte** dynamisch und nutzerInnenfreundlich festzulegen.

- Statt MediaWiki-Syntax füllen NutzerInnen Formularfelder aus, deren Werte in einen Wissensgraphen transcludiert werden.



Umfangreiche Dokumentation zu Page Forms: https://www.mediawiki.org/wiki/Extension:Page_Forms

Hauptbestandteile

- Hauptbestandteil sind sog. Form Definition Pages im `Form:` Namesraum.
- Bei Aufruf der Form wird der Markup-Code geparst und eine HTML5-konforme Formularelement-basierte Seite generiert.
- Auf dieser generierten Seite können anschließend die Parameterwerte der zugrunde liegenden Semantic Templates gesetzt werden.
- Form Definitions können sich auf ganze Seiten oder einzelne Seitenbereiche (engl. Sections) beziehen, abhängig vom zugrunde liegenden Template

here comes a picture

How Page Forms Work

Page Forms use semantic templates in order to create new or edit existing wiki pages. Template parameters are set with form values when the template is embedded into a wiki page.

Usually, for each parameter in a template, Page Forms define a separate **input element declaration**, the value of which will then be assigned to the template parameter when the template call is embedded in the wiki page.

Syntax:

```
{{{field|title|mandatory|property=title|size=120|placeholder=The project's full title}}}
```

The second parameter (`|title`) in the `{{{field...}}}` tag refers to the named parameter (`{{{title|}}}`) in the semantic template a from is defined for, e.g., `[[title::{{{title|}}}]]`.

https://www.mediawiki.org/wiki/Extension:Page_Forms/Defining_forms

```
<noinclude> <!-- instructions -->
  {{#forminput:form=Project Factsheet v2}}
</noinclude><includeonly>
{{{info|add title=Add a new Project|edit title=Edit an existing project}}}

{{{for template|Project Factsheet v2}}}
{| class="formtable"
! Title:
| {{{field|title|mandatory|property=title|size=120
    |placeholder=please enter the full title of the project}}}
|-
! Acronym:
| {{{field|acronym|mandatory|property=acronym|size=20
    |placeholder=Please enter the project's short name (acronym)}}}
|-
! Budget:
| {{{field|budget|input type=text|property=has_budget|size=20
    |placeholder=Please enter the project's budget}}} EUR
|-
! Projektstart:
| {{{field|start_date|input type=datepicker|highlight days of week=1,2,3,4,5
    |property=has_start_date|date format=dd.mm.yyyy}}}
|-
! Beschreibung:
| {{{field|description|input type=textarea|property=has_description|cols=80
    |rows=20|placeholder=Please enter the project's description|autogrow}}}
|-
! Themenfelder:
| {{{field|topics|input type=tokens|property=has_topic|list|delimiter=;}}}
|}
{{{end template}}}
</includeonly>
```

Example:
Page Forms for Project Data

Form Definition Page

```
<noinclude>
  {{#forminput:form=Project Factsheet v2}}
</noinclude><includeonly>
{{{info|add title=Add a new Project|edit title=Edit an existing project}}}

{{{for template|Project Factsheet v2}}}
{| class="formtable"
! Title: | {{{field|title|mandatory|property=title|size=120|placeholder=please enter the full title of the project}}}
|-
! Acronym: | {{{field|acronym|mandatory|property=acronym|size=20|placeholder=Please enter the project's short name (acronym)}}}
|-
! Budget: | {{{field|budget|input type=text|property=has_budget|size=20|placeholder=Please enter the project's budget}}} EUR
|-
! Projektstart: | {{{field|start_date|input type=datepicker|highlight days of week=1,2,3,4,5|property=has_start_date|date format=dd.mm.yyyy}}}
|-
! Projektende: | {{{field|end_date|input type=datepicker|disable days of week=0,6|property=has_end_date}}}
|-
! Beschreibung: | {{{field|description|input type=textarea|property=has_description|cols=80|rows=20|placeholder=Please enter the project's description|autogrow}}}
|-
! Themenfelder: | {{{field|topics|input type=tokens|property=has_topic|list|delimiter=;}}}
|}
{{{end template}}}

{{{for template|Project member with role|label=ProjektmitarbeiterIn hinzufügen|multiple}}}
{| class="formtable"
! Role: | {{{field|role|input type=tokens|mandatory|property=has_role|placeholder=please add or select the employee's role}}}
|-
! Projekteinstieg: | {{{field|start_date|input type=datepicker|property=has_start_date}}}
|-
! Projektende: | {{{field|end_date|input type=datepicker|property=has_end_date}}}
|-
! ProjektmitarbeiterIn: | {{{field|member|input type=combobox|property=has_member}}}
|}
{{{end template}}}
</includeonly>
```

Semantic Template (1/2)

```
<!-- File: Template:Project Factsheet v2 -->

<noinclude>  <!-- instructions -->  </noinclude>
<includeonly>
{{#formlink:form=Project Factsheet v2
  |link text=Projektdaten bearbeiten
  |link type=button|target={{PAGENAME}} }}

{| class="wikitable"
! Projektname | [[title::{{{title|}}}]]
|-
! Akronym | [[acronym::{{{acronym|}}}]]
|-
! Budget | [[has_budget::{{{budget|}}}]]
|-
! Projektstart | [[has_start_date::{{{start_date|}}}]]
|-
! Projektende | [[has_end_date::{{{end_date|}}}]]
|-
! Beschreibung | [[has_description::{{{description|}}}]]
|-
! Forschungsthemen
{{#set:
 has_topic={{{topics|}}}
 |+sep=;
}}
| {{#show: {{PAGENAME}} |?has_topic }}
|-
|}
```

```
<!-- Fortsetzung -->

Die folgenden Mitarbeiter arbeiten auf dem Projekt:

{{#ask: [[refers_to_project::{{PAGENAME}}]] [[type::subobject]]
|?has_member=MitarbeiterIn
|?has_role=Rolle im Projekt
|?has_start_date=Eintrittsdatum
|?has_end_date=Austrittsdatum
|mainlabel=-
}}

[[Category:Research Project]]
</includeonly>
```

Semantic Template (2/2)

```
<!-- File: Project member with role -->

<noinclude>
This is the "Project member with role" subobject template.
It should be called in the following format:
<pre>
    {{Project member with role
     |project= #filled with {{PAGENAME}}
     |role=
     |start_date=
     |end_date=
     |member=
    }}
</pre>
Edit the page to see the template text.
</noinclude><includeonly>
{{#subobject:
|Refers to project={{PAGENAME}} <!-- TODO: Add if -->
|Has role={{{role|}}}
|Has start date={{{start_date|}}}
|Has end date={{{end_date|}}}
|Has member={{{member|}}}
|type=project_membership
}}
</includeonly>
```

Generated Wiki Page

```
{{Project Factsheet v2
|title=Hochpräzise 3-Dimensionale Digitalisierung von Kulturgütern
|acronym=CultLab3D
|budget=1,800,000
|start_date=02.03.2015
|end_date=2018/11/30
|description=Mit CultLab3D sollten Kulturgüter dreidimensional und in sehr hoher Qualität erfasst werden. Die Qualität der Daten sollte geeignet sein, um
auch wissenschaftlichen Ansprüchen zu genügen, die bislang Originalvorlagen erfordern. Das angestrebte System sollte insbesondere hinsichtlich des Aufwands
(u.a. Scan-Geschwindigkeit), der erzielbaren Qualität und der Kosten den Markt revolutionieren.

Der Lösungsansatz basierte auf drei Säulen:
* Entwicklung einer neuartigen Scan-Technologie in Form des mobilen Digitalisierungslabors (CultLab3D), das aus flexibel einsetzbaren Modulen für die Erfassung von
 3D-Geometrien und Materialeigenschaften bestehen sollte.
* Einsatz von semantischen Technologien: Es sollte eine Ontologie zur Erfassung und Klassifizierung von Kulturobjekten erstellt werden. Außerdem wurden die digitalen
3D Kulturobjekte über ihre Metadaten und semantischen Beschreibungen an externe Informationsquellen mit kulturellem Hintergrundwissen,
wie z.B. digitale Bibliotheken oder Informationen im Internet, angebunden.
* Konzeption und Evaluierung von neuen Geschäftsmodellen zum Nachweis der wirtschaftlichen Tragfähigkeit der neuen Technologie.
|topics=Semantic Technologies; Real-time Big Data
}}

{{Project member with role
|role=Research Associate
|start_date=2016/01/01
|end_date=2017/07/31
|member=Matthias Frank
}}
{{Project member with role
|role=Project Leader
|start_date=2015/05/04
|end_date=2018/07/31
|member=Stefan Zander
}}
{{Project member with role
|role=Research Associate
|start_date=2020/05/13
|end_date=2020/05/27
|member=Nicole Merkle
}}
```

Generated Forms Page (1/2)

Edit an existing project: CultLab3D

Title: Hochpräzise 3-Dimensionale Digitalisierung von Kulturgütern

Acronym: CultLab3D

Budget: 1,800,000 EUR

Projektstart: 02.03.2015

Projektende: 30.11.2018

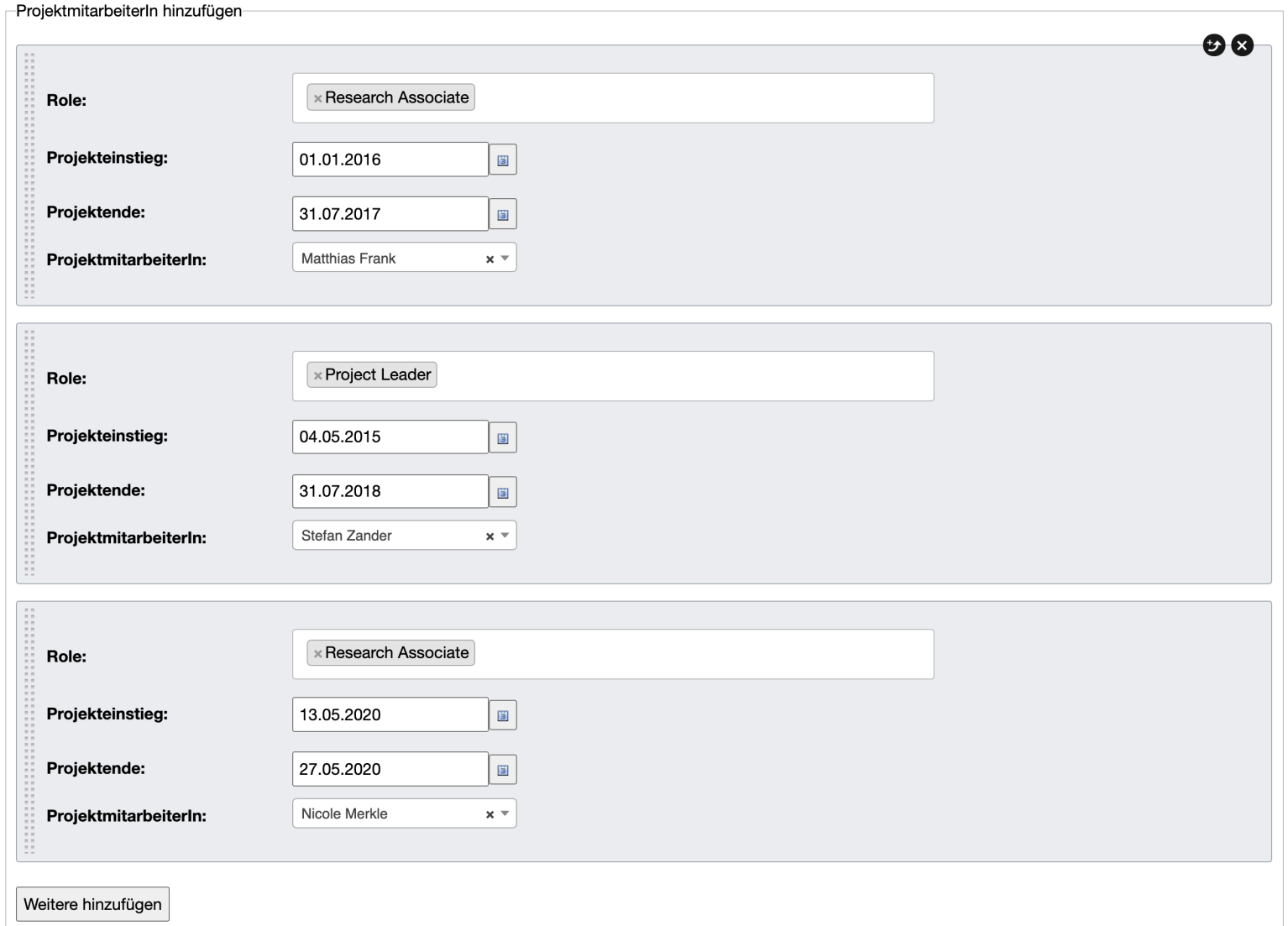
Beschreibung:

Mit CultLab3D sollten Kulturgüter dreidimensional und in sehr hoher Qualität erfasst werden. Die Qualität der Daten sollte geeignet sein, um auch wissenschaftlichen Ansprüchen zu genügen, die bislang Originalvorlagen erfordern. Das angestrebte System sollte insbesondere hinsichtlich des Aufwands (u.a. Scan-Geschwindigkeit), der erzielbaren Qualität und der Kosten den Markt revolutionieren.

Der Lösungsansatz basierte auf drei Säulen:
* Entwicklung einer neuartigen Scan-Technologie in Form des mobilen Digitalisierungslabors (CultLab3D), das aus flexibel einsetzbaren Modulen für die Erfassung von 3D-Geometrien und Materialeigenschaften bestehen sollte.
* Einsatz von semantischen Technologien: Es sollte eine Ontologie zur Erfassung und Klassifizierung von Kulturobjekten erstellt werden. Außerdem wurden die digitalen 3D Kulturobjekte über ihre Metadaten und semantischen Beschreibungen an externe Informationsquellen mit kulturellem Hintergrundwissen, wie z.B. digitale Bibliotheken oder Informationen im Internet, angebunden.
* Konzeption und Evaluierung von neuen Geschäftsmodellen zum Nachweis der wirtschaftlichen Tragfähigkeit der neuen Technologie.

Themenfelder: × Semantic Technologies × Real-time Big Data

Generated Forms Page (2/2)



Getting Started

In order to use **Page Forms** in a (Semantic) MediaWiki system, the following steps need to be taken:

1. **Create Properties**
   Properties are the basic building blocks and data connectors in a SMW system. Every property should be defined.
2. **Create Templates**
   Templates hold semantic markup (properties, categories and subobjects). While every page type is usually represented by a single template, pages could also transclude content from many templates.
3. **Create Forms**
   With Properties and Templates in place, the form definition pages can be created using manual markup code or the built-in creator tool (`Special:CreateForm`). Usually, there should by one form per page type.
4. **Create or Modify Categories**
   One template in each form should tag each article that the form creates as being part of a certain category. Categories can have a default form annotation so that every article that belongs to this category is editable by a form.
5. **Enable Links to Forms**
   Besides defining default forms for categories, it is possible to create links that directly links to forms using the `#formlink` and `#forminput` Parser functions. See [1] for more information on form linking.

Consult the man pages for detailed instructions

Source: Instructions taken and revised from https://www.mediawiki.org/wiki/Extension:Page_Forms/Quick_start_guide

[1] https://www.mediawiki.org/wiki/Extension:Page_Forms/Linking_to_forms

Seitenerzeugung mittels Formularen

Es gibt 2 Wege neue Seiten mittels Formularen zu erstellen oder zu editieren:

Einstufiger Prozess (vgl. one-step process)

1. Direkter Aufruf des Formulars mittels Link oder Button

```
{{#formlink:form=Project Factsheet v2
 |link text=Projektdaten bearbeiten
 |link type=button
 |target={{PAGENAME}}
}}
```

Die `#formlink` Parserfunktion kann auf bel. Seiten verwendet werden, bspw. als Bestandteil des in die Zielseite transkludierten Templates (siehe Screenshot auf Folgeseite).

Beispiel:

Was kann ich hier tun ? (Rolle: Besteller) Bearbeiten

- Eine neue Buchbestellung aufgeben

Zweistufiger Prozess (vgl. two-step process)

1. Separate Seite mit Eingabefeld für Seitennamen
2. Aufruf des Page Forms im 2. Schritt

```
{{#forminput:form=Project Factsheet v2}}
```

Eingebettet am Beginn der Form Definition Page

Quelle: https://www.mediawiki.org/wiki/Extension:Page_Forms/Linking_to_forms

Beispiel des Einstufigen Prozesses

CultLab3D

Seite Diskussion ★ Bearbeiten Versionsgeschichte Neu laden

Projektdaten bearbeiten

Projektname	Hochpräzise 3-Dimensionale Digitalisierung von Kulturgütern
Akronym	**CultLab3D**
Budget	1,800,000
Projektstart	02.03.2015
Projektende	2018/11/30
Beschreibung	Mit CultLab3D sollten Kulturgüter dreidimensional und in sehr hoher Qualität erfasst werden. Die Qualität der Daten sollte geeignet sein, um auch wissenschaftlichen Ansprüchen zu genügen, die bislang Originalvorlagen erfordern. Das angestrebte System sollte insbesondere hinsichtlich des Aufwands (u.a. Scan-Geschwindigkeit), der erzielbaren Qualität und der Kosten den Markt revolutionieren. Der Lösungsansatz basierte auf drei Säulen: • Entwicklung einer neuartigen Scan-Technologie in Form des mobilen Digitalisierungslabors (CultLab3D), das aus flexibel einsetzbaren Modulen für die Erfassung von 3D-Geometrien und Materialeigenschaften bestehen sollte. • Einsatz von semantischen Technologien: Es sollte eine Ontologie zur Erfassung und Klassifizierung von Kulturobjekten erstellt werden. Außerdem wurden die digitalen 3D Kulturobjekte über ihre Metadaten und semantischen Beschreibungen an externe Informationsquellen mit kulturellem Hintergrundwissen, wie z.B. digitale Bibliotheken oder Informationen im Internet, angebunden. • Konzeption und Evaluierung von neuen Geschäftsmodellen zum Nachweis der wirtschaftlichen Tragfähigkeit der neuen Technologie.
Forschungsthemen	Real-time Big Data, Semantic Technologies

Enabling Editing of Page Content via Forms

To get the "edit with form" tab to appear on the page, you must use the parser function `{{#default_form:form-name}}` where `form-name` is the name of the form (without the namespace prefix `Form:`).

a) Based on Category

- Requires that page belongs to the category
- Main template that defines a page type must contain a `[[Category:...]]` tag
- `{{#default_form:form-name}}` need to be placed on the category defintion page
- Every page that uses the template automatically belongs to the category and can call the form
- Recommended approach

b) Based on Namespace

- Every page in a namespace is editable via the specified form
- `{{#default_form:form-name}}` needs to be placed on the page defining the namespace, e.g. `Project:User` [2] for the 'User' namespace
- Categories take precedence over namespaces

c) Within the Page

- Directly associate a form with a page
- Usefull when pages belong to multiple categories with different default forms
- Place the default form parser function `{{#default_form:form-name}}` directly on the page
- Could be combined with templates

Source: https://www.mediawiki.org/wiki/Extension:Page_Forms/The_"edit_with_form"_tab

[2] Create the page if it does not exist. The page for the main-namespace is `Project:Main`.

Query Forms

Query Forms

Definition Query forms are a specific type of page forms and allow to pass individually entered input form data as parameters to query templates. A query form requires a query template, ie., a template with an inline query plus a special page that displays both form data and transcluded template on one wiki page.

Quelle: Individual Definition

1) A Query Form

- Defined as a regular page form
- Displays a "Run Query" button
- Tag `{{{standard input|run query|label=...}}}` allows to customize the "Run Query" button

2) A Query Template

- Defines an inline query together with a set of parameters that serve a query conditions
- Parameter values are passed to the query
- Displays the executed query as result

3) The `Special:RunQuery` Page

- Displays query form and results
- Passes the query form input data to the template and displays the transcluded content on top or below the form

Link to the query form via `[[Special:RunQuery/query form name]]` or `{{#queryformlink:form=query form name}}`.
The parser function is the preferred method as it provides to greatest flexibility.

Source: https://www.mediawiki.org/wiki/Extension:Page_Forms/Creating_query_forms

Query Forms: Implementation

Query Form

```
<includeonly>
{{{info|query form at top}}}
{{{for template|Query_Buchbestellungen}}}
{|
! Besteller*in:
| {{{field|Besteller|input type=tokens|property=Besteller
    |max values=1|values from category=Professor}}}
! Jahr:
|{{{field|Datum|input type=combobox|property=Bestelldatum
    |default={{#time:Y}}
    |values=2019,2020,2021,2022,2023,2024,2025,2026,2027}}}
|}
{{{end template}}}
</includeonly>
```

Query Template

```
{{#ask:
 [[Category:Buchbestellung]]
 [[Besteller::{{{Besteller|+}}}]]
 [[Bestelldatum::>1.1.{{{Datum|1970}}}]]
 [[Bestelldatum::<31.12.{{{Datum|2100}}}]]
 |?Besteller
 |?Kostenstelle |+align=center
 |?Bestellstatus=Status |+align=center
 |?Bestelldatum=Datum |+align=right
 |mainlabel=-
 |format=broadtable
 |class=smwtable-clean sortable
 |headers=plain
}}</includeonly>
```

"Special:RunQuery"-Link

```
[[Special:RunQuery/Query_Buchbestellungen|Meine bisherigen Buchbestellungen aufrufen]]
```

TODO: Add Screenshot

Query Forms – Teil 2

Query Forms kombinieren Formulare mit Templates, in denen eine `#ask`-Query enthalten ist.
Die Template Parameter werden mit den Formular-Werten belegt und über eine Spezialseite transkludiert.

Mittels Query Forms lassen sich die Template-Parameter über Formulareingabeelemente individuell befüllen.

Eine Spezialseite xxx bindet das Query-Form ein, überigbt deren Werte an das Template und stellt das transkludierte Template im Anschluss auf der Seite dar.

```
{{#ask:
 [[Category:BA_Abschlussmodul]]
 [[BA_Betreuer::{{{Betreuer|+}}}]]
 [[BA_Co-Betreuer::{{{Co-Betreuer|+}}}]]
 [[Status_Verfahren::{{{Status|+}}}]]
 [[Abschluss_Semester::{{{Semester|+}}}]]
 ...
```

Wichtig:
Damit das Query-Template bzw. die Query richtig funktioniert, muss bei den Parametern ein `+` als Alternative für einen leeren Wert gesetzt werden!

Other Tipps

- CSC Styles for MediaWiki

Defining individual CSS Styles using MediaWiki:Common.css

SMW allows to change the rendering of its elements using individual CSS style rules.

1. Call the `MediaWiki:Common.css` page (e.g. `https://smw-demo.fbi.h-da.de/index.php/MediaWiki:Common.css`)
2. Get the CSS selector for the elements in question using the style tab in the browser inspector
3. Add the corresponding style rules to the `MediaWiki:Common.css`

Subobjects: Querying using #ask

Using `#subobject` does not print out anything on the screen. To show the subobject data directly on the page where they are defined use an ask query `{{#ask: [[-Has subobject::{{FULLPAGENAME}}]] }}` with inverse property and add it after the definition of the subobjects.

Example

1. Define two subobjects with identifiers "first" and "second":

```
{{#subobject:first
|property1=value1
|property2=value2
}}

{{#subobject:second
|property1=value3
|property2=value4
}}
```

2. Use the `#ask`-query to print out the subobjects' data:

```
{{#ask:
 [[-Has subobject::{{FULLPAGENAME}}]]
 |?property1
 |?property2
}}
```

Please note that the `Has subobject` property need to be specified as inverse property (note the '`-`' in front of the property) in order to make the query work.

Source: https://www.semantic-mediawiki.org/wiki/Help:Subobjects_and_queries

Subobjects: Show Properties of the Subobject's Parent Page

In some cases it is necessary to query for properties that are defined on the subobject's parent page.

a) Using Property Chains

Example

```
{{#ask:
 [[-Has subobject::{{FULLPAGENAME}}]]
 |?-has subobject.YourParentProperty1
 |?-has subobject.YourParentProperty2
 |?YourSubobjectProperty1
 |?YourSubobjectProperty2
}}
```

- Chain members must be of type page
- Last member can be of any type
- Chaining depth is not limited by default

b) Using Templates

1. Query for a property of a subobject.
2. Use `format=template`.
3. Add another query in this template where you ask for `[[Has subobject::{{{1}}}]]`. Has subobject returns the parent page of a subobject. The subobject is queried in the first query and passed on to the template as `{{{1}}}`.

Example

```
{{#ask:
 [[YourSubobjectProperty::Foo]]
 |?YourSubobjectProperty1
 |?YourSubobjectProperty2
 |format=template
 |template=YourTemplate
}}
```

In `Template:YourTemplate` add the following:

```
{{#ask:
 [[Has subobject::{{{1}}}]]
 |?YourParentProperty1
 |?YourParentProperty2
}}
```

See https://www.semantic-mediawiki.org/wiki/Help:Property_chains_and_paths

Searching for Pages with certain Subobject Properties

In some business cases, it is necessary to search for pages with certain subobject properties and display their properties.
The solution is to use subqueries to query for certain subobject properties (e.g. a type property) – so called subproperties.

Example

```
{{#ask:
 [[Has subobject::<q>[[YourSubobjectProperty::Foo]]</q>]]
 |?YourParentProperty1
 |?YourParentProperty2
}}
```

- The query `[[Has subobject::]]` (without the "`-`") queries for the (parent) pages that have certain **subproperties**.
- With the subquery, you can then select certain subobject properties.
- As the query asks for parent pages, you can select properties of the parent pages as printouts.

Source: https://www.semantic-mediawiki.org/wiki/Help:Subobjects_and_queries

QueryForms: Define Yearly Time Frames

If you want to use for specific (e.g. yearly) time frames using **QueryForms**, you have to define 2 things:

1. Define a field tag with `|input type=year` (or use a combo-box with pre-defined values) for the years

```
'''Jahr:'''
{{{field|Datum|input type=combobox|property=Bestelldatum|default={{#time:Y}}
  |values=2019,2020,2021,2022,2023,2024,2025,2026,2027,2028,2029,2030 }}}
```

2. Pass the field parameter value (ie. `Datum`) to a template and use the following query conditions to set a time frame of, e.g., 1 year

```
{{#ask:
 [[Category:Buchbestellung]]
 [[Besteller::{{{Besteller|+}}}]]
 [[Bestelldatum::>1.1.{{{Datum|1970}}}]]
 [[Bestelldatum::<31.12.{{{Datum|2100}}}]]
...
```

cf. https://www.semantic-mediawiki.org/wiki/Help:Date_parsing

Query for Subobject Data on specific Parent Pages

Motivation:

Display the items (represented as subobjects) of all book orders issued in 2020.

Assumption:

Items are represented as subobjects embedded in order pages.

Solution:

1. Draw a data graph of the involved entities
2. Formulate the query conditions for the parent pages
3. Insert the parent page query conditions into a subquery
4. The subquery becomes the value of a query condition using the inverse `Has subobject` property

Example:

Build the subquery (i.e. the query conditions for parent pages)

```
[[Category:Buchbestellung]]
[[Bestelldatum::>1.1.2020]] [[Bestelldatum::<31.12.2020]]
```

Build the full query and embed the subquery

```
{{#ask:
 [[-Has subobject::<q>[[Category:Buchbestellung]]
 [[Bestelldatum::>1.1.2020]] [[Bestelldatum::<31.12.2020]]</q>]]
 |?Verfasser
 |?Buchtitel
 ...
}}
```

Combining Parent Page and Subobject Data in Queries

This can be achieved with subqueries and inverse property chaining in the properties' selection part of inline queries.

Example

```
{{#ask:
 [[-Has subobject::<q>[[Category:Buchbestellung]]
 [[Bestelldatum::>1.1.2020]] [[Bestelldatum::<31.12.2020]]</q>]]
 |?-Has subobject.Besteller         <!-- selecting parent page property via inverse property chaining -->
 |?-Has subobject.Kostenstelle      <!-- selecting parent page property via inverse property chaining -->
 |?Verfasser
 |?Buchtitel=Titel
 |?Erscheinungsjahr=Jahr |+align=center
 |mainlabel=-
 |format=broadtable
 |class=smwtable-clean sortable
 |headers=plain
}}
```

Automatically Setting Page Names in Page Forms

The name of the page created by a form can be set automatically by adding a `page name` parameter within the form definition's `info` tag[1].

Example[2]:

```
{{{info|page name=<Author[First name]> <Author[Last name]>}}}

{{{for template|Author}}}
'''First name:''' {{{field|first name}}}

'''Last name:''' {{{field|last name}}}

'''Political affiliation:''' {{{field|political affiliation|input type=combobox}}}
{{{end template}}}
```

[1] https://www.mediawiki.org/wiki/Extension:Page_Forms/Linking_to_forms

[2] Example is taken from https://discoursedb.org/w/index.php?title=Form:Author&action=edit

Benennung von Properties in Templates

Annahme:
Sie wollen Abschlussarbeiten (BAs/MAs) mittels SMW verwalten. Für BAs & MAs erstellen Sie jeweils separate Templates.

Im Verlauf der Arbeit stellen sich hierbei folgende Fragen:

- Benutzen Sie identische Properties für/in beiden Templates ?
- Wie benennen Sie die Properties ?
- Wie gelingt Ihnen die Unterscheidung der konkreten Instanzen (SWM Seiten) ?

Multiple Values for the same Subobject Property

a) Add Multiple Lines

```
{{#subobject:mysubobject
 |Has property 1=value1
 |Has property 1=value2
 |Has property 3=value1
 ...
}}
```

TODO: Check if it works

b) Separate Values using a Pipe

```
{{#subobject:mysubobject
 |Has property=value1|value2
 ...
}}
```

- Separate values using `|` as delimiter

The possibility of using pipes `|` for setting multiple values was deprecated starting with Semantic MediaWiki 3.0.0 and will be removed in a later version. It is strongly recommended to migrate to using the `|+sep` parameter.

c) Separate Values using Named Separator

```
{{#subobject:mysubobject
 |Has property 1=Value 1;Value 2;Value 3|+sep=;
 |Has property 2=12+22+3+4+5+6+7+8+9+10|+sep=+
 |Has property 3=123,1234,12345,|+sep=,
 |Has property 4=One,or,two,more,values|+sep
 ...
}}
```

- Semantic MediaWiki 1.9.0 introduces `|+sep=...` to identify the separator
- Provides greates flexibility and can be combined with PageForms
- Preferred method in most cases

Alternatively, the `{{#arraymap:...` parser function can be used to separate multiple values for one subobject property.

Source: https://www.semantic-mediawiki.org/wiki/Help:Adding_subobjects#Specifying_multiple_values_for_the_same_property

How to Query for Subobject Data in Pages

Tipps for Working with Forms

Remove 'Keine Angabe' in selection fields (radiobuttons or checkboxes)

- In order to remove the `Keine Angabe` option from the set of selection options in radiobuttons or checkboxes, it is necessary to add the `|mandatory`-flag to the input declaration.
- Optinally, set the `|default=...` parameter to a specific value to have it pre-selected.

Abfragen von Seiten auf denen ein Property NICHT gesetzt ist

Die `#ask` Abfragealgebra von SMW erlaubt es nicht, Seiten abzufragen, auf denen ein Property nicht gesetzt ist, d.h., keinen Wert hat[1].

Hierzu muss man sich eines Tricks mittels `#if` und `#set` behelfen[2]:

```
{{#if: {{{Competitor}}} | | {{#set:Competitor provided=No}} }}
```

Erklärung:
`#if` evaluiert, ob ein übergebener String leer ist oder nicht. Im obigen Beispiel wird der Parameter nicht übergeben, er ist also leer da ihm kein Wert zugeordnet ist, und so wird der `else`-Teil der `#if` Parser function ausgeführt und ein Property mit dem angegebenen Wert auf der Zielseite gesetzt.

Das Property `Competitor provided` muss vom Datentyp `Boolean` sein. Nicht notwendigerweise

Jetzt kann mittels einer Query nach Seiten mit dem gesetzten Wert des Properties (im o.g. Beispiel `Competitor provided` und `No`) gesucht werden.

```
{{#if: test string | value if test string is not empty | value if test string is empty (or only white space) }}
```

[1] Quelle: https://www.semantic-mediawiki.org/wiki/Thread:Semantic-mediawiki.org:Community_portal/Querying_for_non-empty_values
[2] Dieser Trick funktioniert nur in Zusammenhang mit Templates und Parametern.
[3] https://www.mediawiki.org/wiki/Help:Extension:ParserFunctions##if

Formatting of Boolean Values in Queries

By default, **Boolean values** will display as `true` and `false` in queries. This can be changed in Semantic MediaWiki >= 2.4.0 by specifying a format string to control what is displayed for the true and false values of a property.

The following formatting options are supported

- `#LOCL` formatter (#LOCL to localize the output)
- `#num` formatter (to produce 1 or 0)
- `#tick` formatter (to display ✓ or ✗)
- `#x` formatter (#x to output X that resembles a cross for true)
- Custom label formatter (Using custom labels to define a boolean output and assign a true, false value display)

Note
Not all result formats may provide support for these formatters.

Syntax

```
{{#ask:
 [[Category:Project]]
 |?Has boolean#<FORMATTER>
}}
```

Examples

```
|?Has boolean#LOCL

|?Praxisphase_absolviert#tick=Fertig
```

Source: https://www.semantic-mediawiki.org/wiki/Help:Type_Boolean

Multiple Values for the same Field

Use `\s` as new delimiter to insert a space between the single values.

Source: https://www.mediawiki.org/wiki/Extension:Page_Forms/Page_Forms_and_templates#Multiple_values_for_the_same_field

Tipps for Working with Dates and Page Forms

Using the current date in an input element[1]

Using input type `datepicker`, it is possible to set the current date as default value in the input element using the `#time` parser function[2] of the `Extension:ParserFunctions`[3]

Example:

```
! Datum:
| {{{field|Datum|input type=datepicker|property=Bestelldatum|date format=dd.mm.yy|default={{#time:d.m.Y}} }}}
|}
```

Please note:
Regardless of the formatting pattern set via the `date format=...` property, `datepicker` always returns the data in `yyyy/mm/dd` format.

[1] https://www.mediawiki.org/wiki/Topic:Put4ygy4m0kw279f
[2] https://www.mediawiki.org/wiki/Help:Extension:ParserFunctions##time
[3] https://www.mediawiki.org/wiki/Help:Extension:ParserFunctions

Using an Individual Display Format for Date and Time

This tip helps you to reformat the output of dates which is by default `YEAR/MONTH/DAY` (e.g. 2017/06/27) and still keep a proper annotation for it.

Prerequisite is that you have the extension `ParserFunctions` installed on your wiki.

1.) ISO-Style

```
[[Startdate::{{{startdate|}}}|{{#time:Y-m-d|{{{startdate}}}}}]]
```

--> 2011-06-27

2.) German-Style

```
[[Startdate::{{{startdate|}}}|{{#time:d.m.Y|{{{startdate}}}}}]]
```

--> 27.06.2011

The parser function `#time` takes a date and/or time (in the Gregorian calendar) and formats it according to the syntax given[2,3].

```
Format: {{#time: format string | date/time object | language code | local }}
```

Source: https://www.semantic-mediawiki.org/wiki/Help:Date_formatting

[2] A full list of parameters is available on the help page: https://www.mediawiki.org/wiki/Help:Extension:ParserFunctions##time

[3] In case the `date/time object` parameter is missing, the value of the magic word `{{CURRENTTIMESTAMP}}` (i.e., the time the page was last rendered into HTML) will be used.