Contents

Manual:Extension/Page Forms

With **Page Forms,** users without administrator rights can use forms to create and edit pages to query data - without programming knowledge.

The use of the extension is documented on documented on MediaWiki.

Contents

Main features

- **Definition pages in the namespace *Form*** The New forms can be created using the special page `Special: CreateForm`. Here, users select an existing template which provides the paramaters for the form. This means that before a form is created, the required template is always created first. All created form definition pages are saved in the *Form* namespace. Subsequent edits to the form definition page have to be made in source editing mode.
- **Application example: info boxes** Page Forms is often used to add and edit infoboxes on a wiki page. If Semantic MediaWiki is used, the collected data in the templates can be stored and retrieved.
- **Edit existing forms values via menu item** Existing values in a template can be updated using the menu item "Edit with form" of the page edit button, for example.
- **Automatic completion of fields** Users Users are offered existing values when entering them, depending on the form input type. This reduces problems with naming ambiguities, spelling, etc.
- **Free text field** Free text on the page that is not part of the template itself can be displayed in a separate input field called "Free text" for editing directly in forms mode.

BlueSpice input types

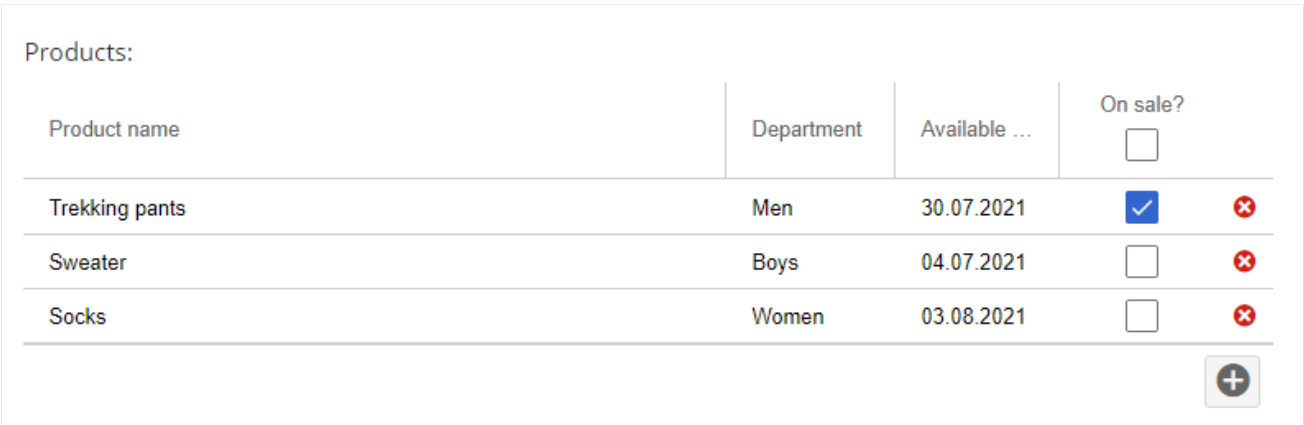
In addition to the default input types, BlueSpice offers the following additional input types:

Input type	Result	Function
bs-grid	input table	grid-based input field to combine related paramters. Table rows can be added with a "+" button. The following templates need to be created: • Template for the table row plus its related columns.json page • Template for the output of the grid on a wiki page
bs-usercombo	User name (with link to the profile page)	(Single selection).
bs-usertags	Comma-separated user name	Menu that allows to select existing wiki users (multiple selections possible). Note: To link to the profile page, the corresponding parameter in the template needs to be formated accordingly: `{{#arraymap:{{{myParameter|}}}|,|@@@|[[User:@@@|@@@]]}}`
bs-		

Input type	Result	Function
mvvisualeditor	Formatted text	Text box with simplified VisualEditor.
bsvisualeditor	-	*Obsolete - replaced by bs-mvvisualeditor*

bs-grid

Bs-grid provides the possibility to use table rows to collect combined values for a particular parameter:



entering grid-data for a single form field

To create the grid-based form field and its output:

1. **Create** the template *Template:Products/Row*.
    1. **Define** the parameters, for which to collect values. Gere we create a table row, so that we can later display the collected data as a table:

```
<noinclude>Table row for the output of the product data</noinclude><includeonly>
|-
|{{{product|}}}
|{{{department|}}}
|{{{availDate|}}}
|{{{sale|}}}
</includeonly>
```

2. **Define** the grid in the page *Template:Products/Row/Columns.json*:

```
[
{"header":"Product name","dataIndex":"product", "flex":1, "editor":{"allowBlank":fal
se}},
{"header":"Department","dataIndex":"department","editor":{"xtype":"combo","typeAhead
":true,"triggerAction":"all","store":[["Toddler","Toddler"],["Boys","Boys"],["Girls"
,"Girls"],["Men","Men"],["Women","Women"]]}},
{"xtype":"datecolumn","header":"Available from","dataIndex":"availDate","format":"d.
m.Y","editor":{"xtype":"datefield","format":"d/m/y","minValue":"01/01/21"}},
{"xtype":"checkcolumn","header":"On sale?","dataIndex":"sale","headerCheckbox":true,
"stopSelection":false}
]
```

Note: The syntax of this json file is produced by the Ext JS framework. Links to the documentation of the grid syntax are provided under *Related info* at the end of this page (JS-knowledge required).

3. **Create** the page *Template:Products.* It contains the output format for the data. It is also formatted as a filterable table.

```
<noinclude>Output table for product data.
The parameter "productdate" is processed in the form Form:Products.
</noinclude><includeonly>{{#default_form: Products}}
{| class="wikitable filterable"
|+Product overview for our current collection
!Product name
!Department
!Available from
!On sale?
{{{productdata}}}
|}
</includeonly>
```

4. **Create** the data entry form *Form:Products*. The form field *productdata* defines the data entry type as a table (bs-grid):

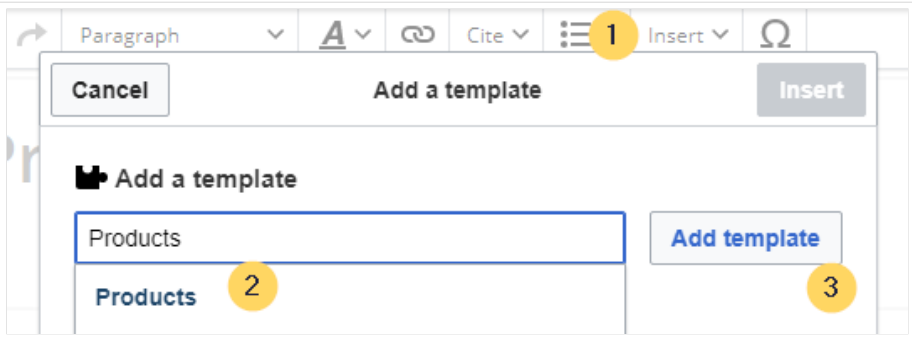
```
<noinclude>This is the form "Products".It is being used with the template Template:
Products.</noinclude><includeonly>
<div id="wikiPreview" style="display: none; padding-bottom: 25px; margin-bottom:
25px; border-bottom: 1px solid #AAAAAA;"></div>
{{{for template|Products}}}
Products:

{{{field|productdata|input type=bs-grid|colDef=Template:Products/Row/Columns.
json|template=Products/Row}}}

{{{end template}}}

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

5. **Include** the template "Products" on a wiki page.
   1. **Click** *Insert > Template* in the editor toolbar.
   2. **Enter** "Products.



add the Products template to a page

   3. **Click** *Add template.*
   4. **Save** the page.
   5. **Open** the page in form-edit mode (using the drop-down menu next to the "+"-button in the top toolbar.The page open in forms mode. Enter your data in the products grid.
   6. **Save** the page again. The filterable products table is now shown.

Product overview for our current collection

Product name	Department	Available from	On sale?
Trekking pants	Men	30.07.2021	true
Sweater	Boys	04.07.2021	
Socks	Women	03.08.2021	

Output of the grid data for parameter productdata

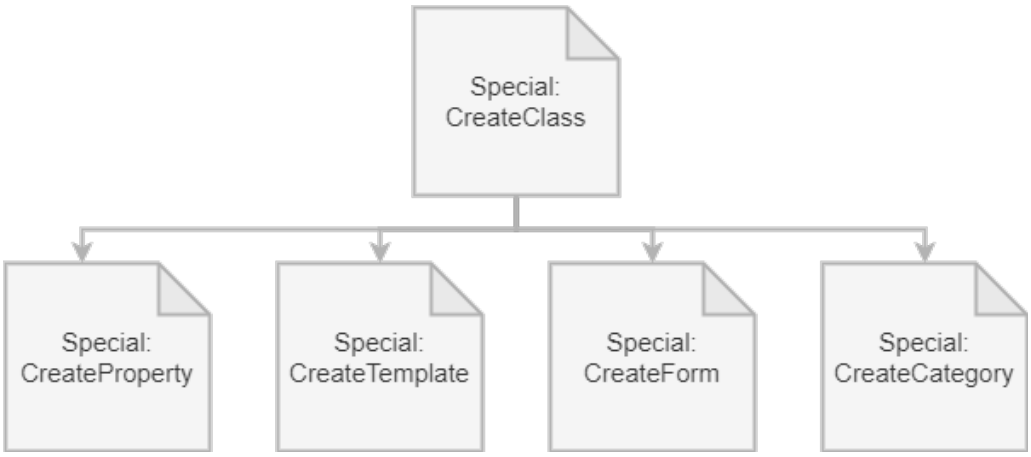
In source code view, the *productdata* parameter looks like this:

```
{{Products
|productdata={{Products/Row|product=Trekking pants|department=Men|availDate=30.07.2021
|sale=true}}
{{Products/Row|product=Sweater|department=Boys|availDate=04.07.2021}}
{{Products/Row|product=Socks|department=Women|availDate=03.08.2021}}
}}
```

Special pages

Page Forms defines some special pages that are used for data input and data maintenance.

Among others, the following special pages are important for data collection:



Related info

- https://www.mediawiki.org/wiki/Extension:Page_Forms/en
- Reference:Page Forms

Technical Reference: Page Forms

Reference:Page Forms

Extension: Page Forms

all extensions

Overview			
Description:	Forms for creating and editing wiki pages		
State:	stable	**Dependency:**	MediaWiki
Developer:	Yaron Koren, Stephan Gambke, and others	**License:**	GPL v2+
Type:	MediaWiki	**Category:**	Content Structuring
Edition:	BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	**Version:**	4.1+
View help page			

Features

Page Forms (known until November 2016 as **Semantic Forms**) is an extension to MediaWiki that allows users to add, edit and query data using forms.

It was originally created as an offshoot of the Semantic MediaWiki extension to be able to edit templates that store their parameters via SMW. However, it can now also work with the alternative Cargo extension, or with neither extension installed.

Very simply, Page Forms allows you to have **forms for creating and editing pages** on your wiki, as well as **for querying data**, all **without any programming**. Forms can be created and edited not just by administrators, but by users themselves.

The main components of Page Forms functionality are form definition pages, which exist in a separate namespace, 'Form:'. These are pages consisting of markup code which gets parsed when a user goes to a form. Since forms are defined strictly through these definition pages, users can themselves create and modify forms, without the need for any actual programming.

The Page Forms extension is mostly used to add and edit calls to infobox-style templates within pages. A form allows a user to populate a pre-defined set of templates, as well as page sections, for any page; if Cargo or Semantic MediaWiki are used, the data within the templates can then be stored and queried.

Forms can also be used to edit the data in an existing page, and you can enable an "formedit" tab to show up on any page.

Page Forms also supports **autocompletion of fields**, so users can easily see what the previously-entered values were for a given field. This greatly helps to avoid issues of naming ambiguity, spelling, etc.

Data in a page that doesn't fit into the form, like a free-form text description of the page's subject, isn't ignored when the page is edited with a form; rather, it is placed into a separate input box called "free text".

Page Forms also includes a wide variety of other functionality related to forms.

Accessibility

Test status:	2-testing complete
Checked for:	Web, Authoring tool
Last test date:	2022-08-05
WCAG level:	AA
WCAG support:	partially supports (workaround: no)
	some input types are not keyboard or screenreader accessible. E.g., tokens, bs-msvisualeditor.

Comments:	Standard input elements such as text fields, drop downs, textareas, etc. are accessible. Therefore, accessible forms can be created.
Extension type:	extended
Extension focus:	editor