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

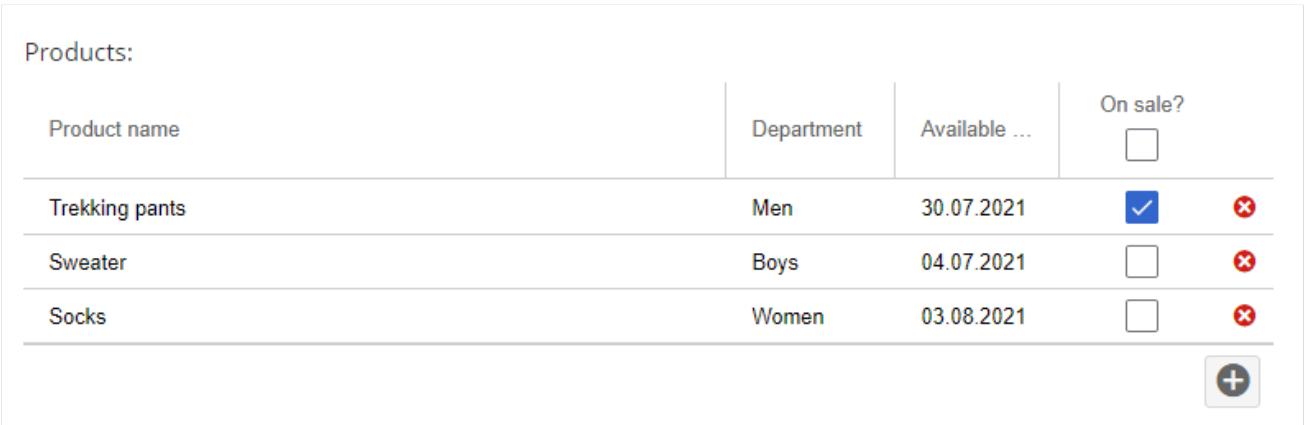
In addition to the default input types, BlueSpice offers the following additional input types:

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):

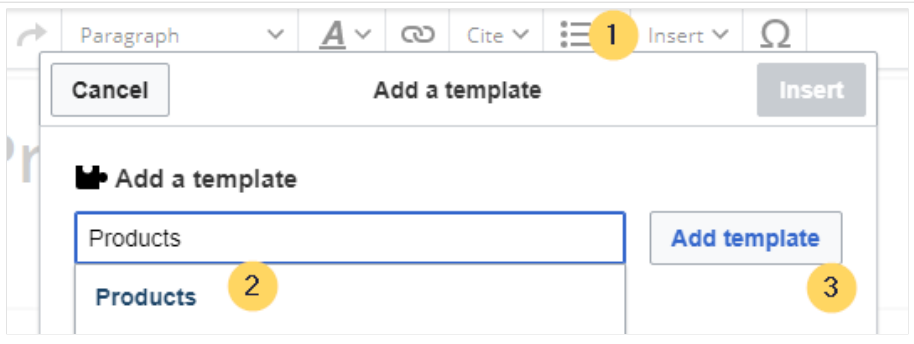
```
<noinclude>This is the form "Products".It is being used with the template Template:
Products.</noinclude><includeonly>
<div id="wikiPreview" style="display: none; padding-bottom: 25px; margin-bottom:
25px; border-bottom: 1px solid #AAAAAA;"></div>
{{{for template|Products}}}
Products:

{{{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

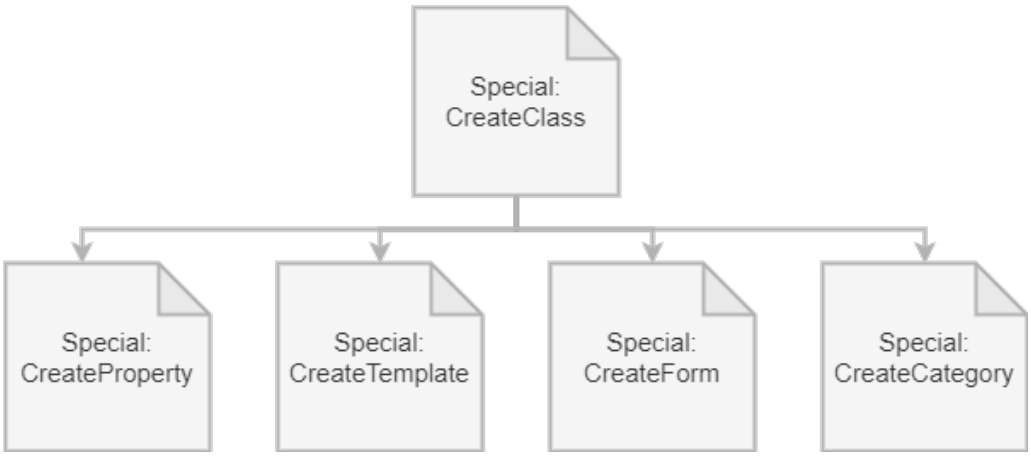
In source code view, the *productdata* parameter looks like this:

```
{{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

Manual:Extension/Page Forms/Snippets

Contents

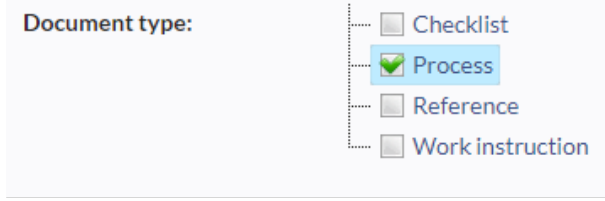
Providing input values from a (sub-) category

Using input type "tree"

(see official documentation: mediawiki.org/wiki/Extension:Page_Forms/Input_types#tree)

```
{{{field|doctype|input type=tree|top category=Document type|depth=1|hideroot|list}}}
```

Form display:



Field type tree

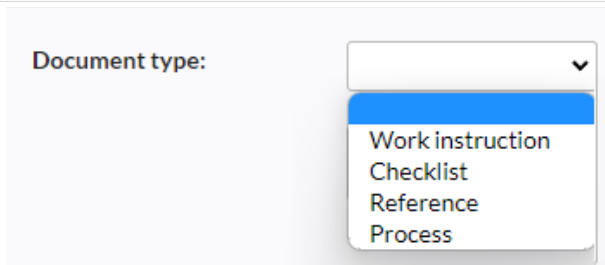
Example output of template parameter: `doctype=Process`

Using DPL query (dropdown)

(Solution found at: https://www.mediawiki.org/wiki/Extension_talk:DynamicPageList3#Return_page_titles_as_plain_text?

```
{{{field
 | doctype
 | input type=dropdown
 | values = {{#dpl: namespace = Category
 | category = Document type
 | mode = inline
 | format = ,¶%TITLE%,,
 | inlinetext = ,
  }}
}}}
```

Form field:



values provided by dpl3 parser function

Example template output:

`doctype=Process`

Using an SMW inline query

```
{{{field|doctype|input type=dropdown|values={{#ask: [[Subcategory of::Document type]]
|format=plainlist|sep=,|link=none|template=SMW Pagename}} }}}
```

The above example uses the mapping template *Template:SMW Pagename* to avoid showing the namespace prefix "Category:" in the dropdown. Create the template with the following content:

```
{{PAGENAME:{{{1|}}} }}
```

Although the category prefix is then not shown in the dropdown, the value the form passes to its associated template parameter still includes the namespace prefix.

Example template output:

`doctype=Process`

Using an SMW concept

Create the page Concept:Doctype with the following content:

```
{{#concept:
 [[Subcategory of::Documenttype]][[Modification date::+]]
 |List of sub categories of the category Documenttype
}}
```

Use this concept as the source for the values of your field:

```
{{{field|doctype|input type=dropdown|values from concept=doctype|mapping
template=catConcept}}}
```

The above example uses the mapping template *Template:catConcept* to avoid showing the namespace prefix "Category:" in the dropdown. Create the template with the following content:

```
{{PAGENAME:{{{1|}}} }}
```

Although the category prefix is then not shown in the dropdown, the value the form passes to its associated template parameter still includes the namespace prefix.

Example template output:

`doctype=Category:Process`