

Contents

1. Manual:Extension/Forms	2
2. Reference:Forms	9
3. Reference:Workflows	12

Manual:Extension/Forms

The **Forms** extension allows formatting a wiki page or a user dialog as a form. It is also used in conjunction with the [Workflows](#) extension where it provides input forms for workflow data.

Contents

1 Introduction	3
2 Create a checklist form	3
3 Create a checklist instance	6
4 Send email	6
5 Adjust the styling	7
5.1 Common.css	8
5.2 Inline styling	9
6 Related info	9

Introduction

To create a checklist as a form, for example, the following steps are necessary:

1. **Create and save the checklist form** via the *Special:Form_editor* special page. The form serves as a template for all checklists that are created with it.
2. **Create and save checklist instances** via the special page *Special:Create_form_instance*.

Optionally, the styling of the form can be adjusted.

As an example, we are creating a small checklist for an event where food-related hygiene information has to be tracked.

For each day of the event, a separate form has to be completed! The form has to be completed by 1 p.m.

Event

Event name

Fall fest 2022

Organizer

Hallo Welt!

Date

09/22/2022

Contact

John Doe

Hygiene requirements

Booth (floor, walls, ceiling)

✓

Handwash station (soap dispenser, disposable towels)

✗

Delivery processing

Refrigerated products

☒ Max. temperatur 7° C

☒ Packaging

☒ Expiration date

Baked goods

☒ Packaging

☐ Expiration date / freshness

Storage

Refrigerator 1

clean

Other comments

Event checklist example

Create a checklist form

1. **Define** the main properties of the form:
 1. **Open** the *Special:Form_editor* page.
 2. **Enter** a name for the new form, e.g. *EventChecklistHygiene*. This name is used as a unique identifier for the database. Therefore, do not use any special characters (apart from the hyphen "-").

2. **Click** the text box radio button.
3. **Enter** a name. This is used as the database ID for this field and should therefore not contain any special characters.
4. **Enter** a label. This is displayed as the label for the text field.
5. **Add** more form elements.
4. Define which form fields should be used to generate the page name of the form instances created with this form. As an example, we define the page name for the checklists as subpages to the form name with the values from the event name and date fields (both fields must exist in the form: `EventChecklistHygiene/{{ech-eventname}}-{{ech-date}}`)
 1. **Mark** *event name* and *day of event* as required so that these values always exist.
 2. **Switch** to the *Target* view in the form properties.
 3. **Select** "Json in a wiki page" as the target type.
 4. **Enter** your title syntax: `EventChecklistHygiene/{{ech-eventname}}-{{ech-date}}`

Form properties

Appearance

Behaviour

Infrastructure

Target

Listeners

Target Type

Title

Assign after action

JSON on wiki page

EventChecklistHygiene/{{ech-eventname}}-{{ech-date}} *

☐

Define the page name syntax for form instances

5. **Click** Submit at the bottom of the page to save the form. The form is now stored in the main namespace as *EventChecklistHygiene.form*.

EventChecklistHygiene

1

Event Checklist

Event

Event name

Submit

Reset

Saved form

To continue editing the form, switch to the editing mode of the page (1).

Create a checklist instance

To use the form, create your first checklist.

1. **Open** the page *Special:Create_form_instance*.
2. **Choose** the form *EventChecklistHygiene*.
3. **Fill out** the form fields.
4. **Click** *Submit*.

Special Create form instance

Create form instance

Select the form to create

Event Checklist

Event

Event name *

Day of event *

EventChecklistHygiene

Eigenkontrolle-veranstaltung

EventChecklistHygiene

Submit Reset

Create a checklist instance

The checklist is now saved in the wiki and can be edited further.

Send email

The form can also send an Email after being saved:

The screenshot shows the 'Target' configuration page in the BlueSpice Forms extension. On the left is a sidebar with icons and labels for 'Appearance', 'Behaviour', 'Infrastructure', 'Target' (which is selected and highlighted in blue), and 'Listeners'. The main area is divided into two columns. The left column contains labels for 'Target Type', 'Receiver', 'Subject', 'Content', and 'Assign after action'. The right column contains the corresponding input fields: a dropdown for 'Target Type' set to 'Email', a dropdown for 'Receiver' set to 'Marketing', a text area for 'Subject' containing 'Dear {{{name}}},', a larger text area for 'Content' containing 'This is the content: {{{content}}}', and a checkbox for 'Assign after action' which is currently unchecked. Below this main configuration area is a section titled 'Form elements' with a sub-section 'Element picker' and a dropdown arrow. The 'Element picker' contains two input fields: 'Name' and 'Content', both of which are empty.

- **Target type:** Email
- **Recipient:** Possible recipients must have been configured on the server for security reasons. Example:

```
$wgFormsTargetEmailRecipients = [  
'Marketing' => "marketing@example.com",  
'Administration' => 'WikiSysop'  
];
```

- **Subject:** Existing form fields can be used as variables here. The value is taken from the form.
- **Content:** As in the subject, existing form fields can also be used as variables here. The value is taken from the form.

[Localization messages](#) in combination with form fields can also be used as subject or email body:

```
{{int:Mailsubject| {{{name}}}|{{{surname}}} }}
```

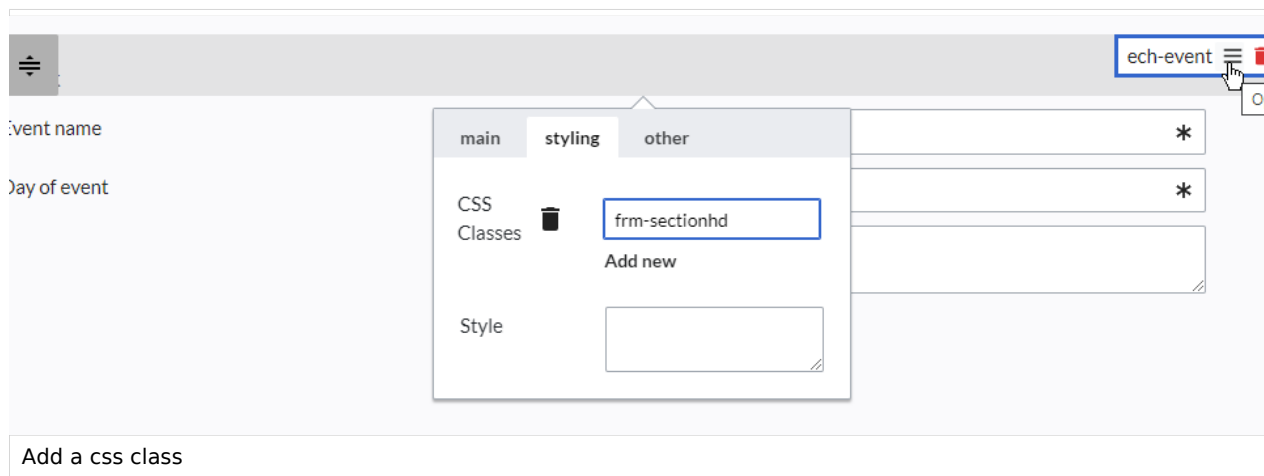
Adjust the styling

Common.css

To style individual elements in the form, go to the styling tab in the options dialog. There, you can enter a css class name. The styling is then defined on the *MediaWiki:Common.css* page (admin rights are required).

To style the section heading:

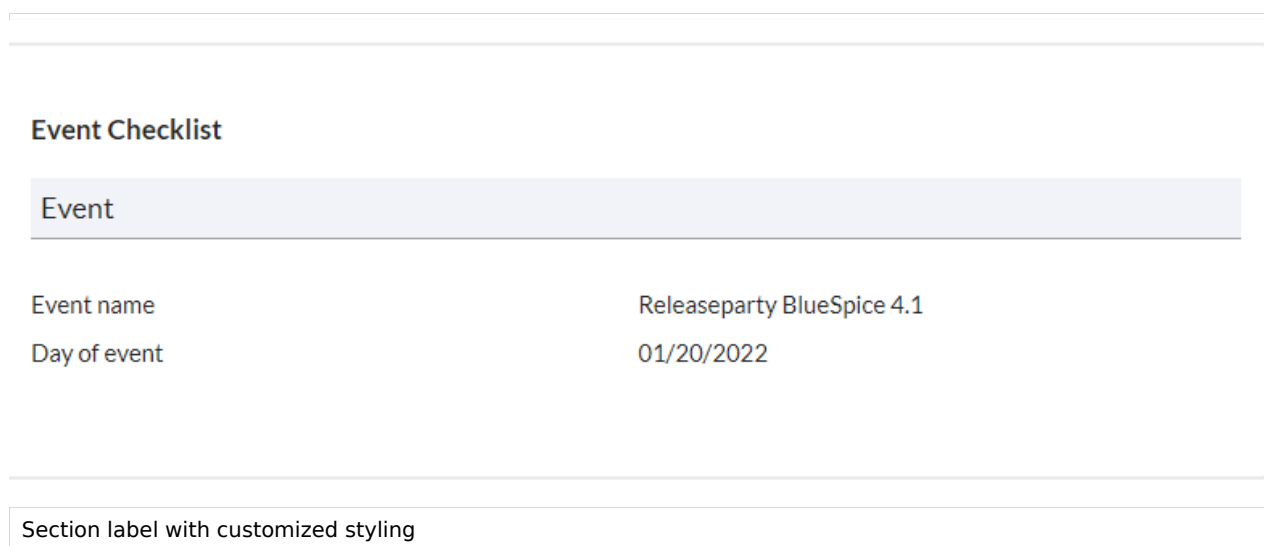
1. **Open** the styling tab in the options menu of the Event heading field.
2. **Add** a new CSS class and enter a selector name (*CSS Classes* field), e.g. *frm-sectionhd*.



The screenshot shows the 'ech-event' options dialog for the 'Event name' field. The 'styling' tab is active, displaying the 'CSS Classes' field with the value 'frm-sectionhd' and an 'Add new' button. The 'Style' field is empty. The dialog is overlaid on a form with fields for 'Event name' and 'Day of event'.

3. **Paste** the following style declarations into the *MediaWiki:Common.css* page, e.g.:

```
border-bottom: 1px solid #a6a6a7; background:#f1f3f9; padding:6px;margin:30px 0;
```



The screenshot shows the 'Event Checklist' section of the form. It contains a table with two columns: 'Event' and 'Releaseparty BlueSpice 4.1'. The 'Event' column has a header 'Event' and a row 'Event name'. The 'Releaseparty BlueSpice 4.1' column has a row 'Day of event' with the value '01/20/2022'.

If you do not have permission to view this page, you can enter style information directly for each element in the form via inline styling.

Inline styling

Without access to *MediaWiki:Common.css*, the style information can be entered directly in the form. However, with many recurring style declarations this is not efficient in contrast to the definition of CSS classes in *MediaWiki:Common.css*.

As an example, let's apply the previous section heading example directly as inline styling. To do this, open the options menu of the *ech-event* form element again. Enter the following statement in the *Styling* field and save the change:

```
border-bottom: 1px solid #a6a6a7; background:#f1f3f9; padding:6px;margin:30px 0;
```

The screenshot shows the MediaWiki Forms extension interface. At the top, there is a header bar with a hamburger menu icon and the text 't'. Below this, the form element 'ech-event' is visible. The 'Options' menu is open, showing three tabs: 'main', 'styling', and 'other'. The 'styling' tab is selected. In the 'styling' tab, there is a 'CSS Classes' section with a trash icon and an 'Add new' button. Below this is a 'Style' section with a text area containing the CSS code: `border-bottom:1px solid #a6a6a7; background:#f1f3f9; padding:6px;margin:30px 0;`. To the right of the 'styling' tab, there are two input fields, each with an asterisk (*) indicating they are required. At the bottom right of the options menu, there are 'Submit' and 'Cancel' buttons. Below the form element, the text 'Inline styling of a form element' is displayed.

Related info

- [<https://www.mediawiki.org/wiki/HTMLForm> Full documentation on mediawiki.org]



Technical Reference: Forms

Reference:Forms

Extension: Forms

→ [all extensions](#)

Overview			
Description:	Allround form framework for MediaWiki		
State:	stable	Dependency:	MediaWiki
Developer:	Hallo Welt!	License:	GPL-3.0-only
Type:	MediaWiki	Category:	Rich Articles
Edition:	BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	Version:	4.1+
? View user help page			

Features

This extension allows the formatting of a wiki page or a dialog window as a form. Besides being useful for creating form instances as wiki pages, this extension is also used in conjunction with the [Workflows](#) extension.

[Full documentation on mediawiki.org](#)

Technical Information

This information applies to BlueSpice 4. Technical details for BlueSpice Cloud can differ in some cases.

Requirements

MediaWiki: 1.39.0

Integrates into

- Forms

Special pages

- [CreateFormInstance](#)
- [FormEditor](#)

Permissions

Name	Description	Role
forms-create-form	Create forms	editor
forms-edit-form-definition	Edit form definitions	admin

Configuration

Name	Value
FormsTargetEMailRecipients	array ()

API Modules

- [forms-form-submit](#)
- [forms-get-definitions](#)

Hooks

- [BlueSpiceDiscoveryTemplateDataProviderAfterInit](#)
- [ChameleonSkinTemplateOutputPageBeforeExec](#)
- [CodeEditorGetPageLanguage](#)
- [ContentHandlerDefaultModelFor](#)
- [LoadExtensionSchemaUpdates](#)
- [ParserFirstCallInit](#)
- [SkinTemplateNavigation::Universal](#)

Accessibility

Test status:	2-testing complete
Checked for:	Web, Authoring tool
Last test date:	2022-08-08
WCAG level:	AA
WCAG support:	does not support (workaround: no)

Comments:	Form editor: - drag-and-drop; no keyboard support Form instance edit: - missing field label associations Read instance: - missing field label associations, checkboxes are read multiple times.
Extension type:	extended
Extension focus:	editor

Reference:Workflows

Extension: Workflows

→ [all extensions](#)

Overview			
Description:	Allows to define and execute various workflows on the wiki		
State:	stable	Dependency:	MediaWiki
Developer:	Hallo Welt!	License:	GPL-3.0-only
Type:	MediaWiki	Category:	Quality Assurance
Edition:	BlueSpice pro, BlueSpice Farm, BlueSpice Cloud	Version:	4.1+
? View user help page			

Features

Workflow functionality based on BPMN. BlueSpice included some standard workflows. Custom workflows can be created additionally.

Technical Information

This information applies to BlueSpice 4. Technical details for BlueSpice Cloud can differ in some cases.

Requirements

MediaWiki: 1.39.0

Forms: *

- OOJSPlus: *

Integrates into

- BlueSpiceDiscovery
- UnifiedTaskOverview
- Workflows

Special pages

- WorkflowsOverview

Permissions

Name	Description	Role
workflows-admin	Manage workflows	admin
workflows-execute	Initiate and advance workflows	admin, reviewer, editor
workflows-view	View workflow items	reader

User options

Name	Value
echo-subscriptions-email-workflow-cat	1

Hooks

- [ArticleDeleteComplete](#)
- [BeforePageDisplay](#)
- [BlueSpiceDiscoveryTemplateDataProviderAfterInit](#)

- [CodeEditorGetPageLanguage](#)
- [ContentHandlerDefaultModelFor](#)
- [LoadExtensionSchemaUpdates](#)
- [MWStakeCommonUIRegisterSkinSlotComponents](#)
- [MWStakeRunJobsTriggerRegisterHandlers](#)
- [PageSaveComplete](#)
- [SetupAfterCache](#)
- [SkinTemplateNavigation::Universal](#)
- [UnifiedTaskOverviewGetTaskDescriptors](#)