Alkacon Software GmbH

An der Wachsfabrik 13
DE - 50996 Köln

Geschäftsführer
Alexander Kandzior

Amtsgericht Köln
HRB 54613

Tel: +49 (0)2236 3826 - 0
Fax: +49 (0)2236 3826 - 20

http://www.alkacon.com

OpenCms 8.5.1 Documentation

Thursday, February, 7, 2013

Document version 1.2

Contents

1 Introduction

This documentation has been developed by Alkacon Software - The OpenCms Experts. We offer standard service and support packages for OpenCms, providing an optional layer of security and convenience often required for mission critical OpenCms projects. For more information about our services, please contact us at info@alkacon.com.

It aims to help developers from getting started with OpenCms up to getting familiar with advanced development topics regarding OpenCms. It uses links referring a locally installed OpenCms version >= 8.5.1 including the OpenCms v8 Modules (modules-v8) together with the OpenCms Developer Modules (dev-demo). In order to use the interactive development demo from OpenCms 8 together with this documentation we assume you have OpenCms up and running with a the OpenCms main servlet is reachable at: `http://localhost:8080/opencms/opencms`. If so, you are able to use the links inside this documentation that will open the according DEMO pages inside the "Wonderful world of flowers".

If you don't have a running OpenCms including the Alkacon OpenCms Demo Modules yet, you can download the latest OpenCms release here and read the OpenCms server installation instructions that are located in the distributed ZIP (installation.html) and you will see how easy it is to setup OpenCms.

This documentation is published under the GNU FDL Free Documentation License by Alkacon Software GmbH. We certainly welcome all contributions and feedback to this documentation.

2 Page editor

The Page editor is the main view that allows you to add, arrange and edit the contents of a page while previewing it. Moreover, it provides access to all necessary user tasks like sitemap manipulation, publishing and file attribute or property handling.

2.1 Toolbar

The toolbar can be toggled by clicking the colored edit point top-right, when browsing the offline preview of the webpage. The toolbar is docked on the window top and shows available options for editing the currently displayed page.

Available options

Publish: Opens the publish dialog.

Clipboard: Opens the clipboard, where favorites and recent changes can be accessed.

Add content: The "Add content" allows adding content to a page by drag & drop.

Edit point: Clicking "Edit point" marks editable elements on the page with grey points.

Context: Shows the Context menu for the current page.

Sitemap: Switches to the sitemap editor.

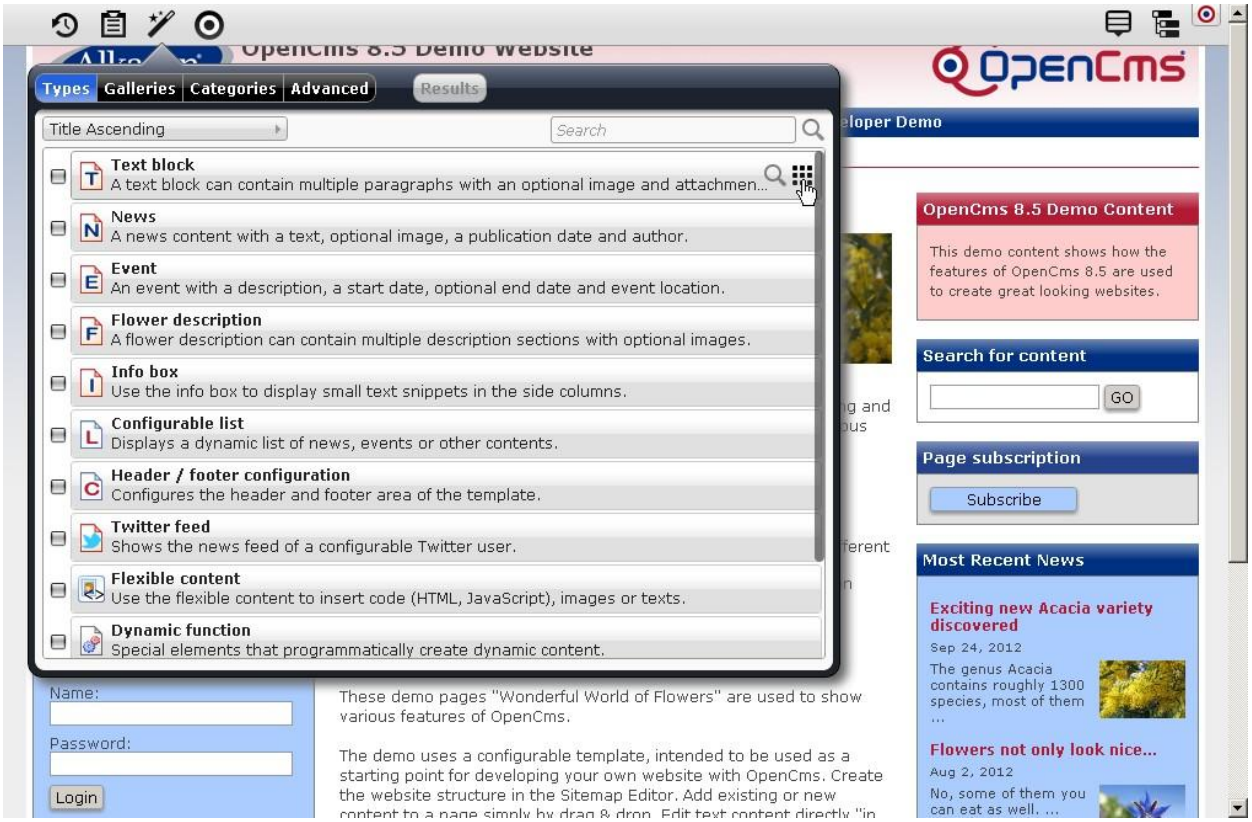
2.2 Edit points

By clicking on the **Edit point** on the toolbar, you can turn on / off the edit mode for all elements on the page. The edit points appear on all editable elements.

2.3 Add content

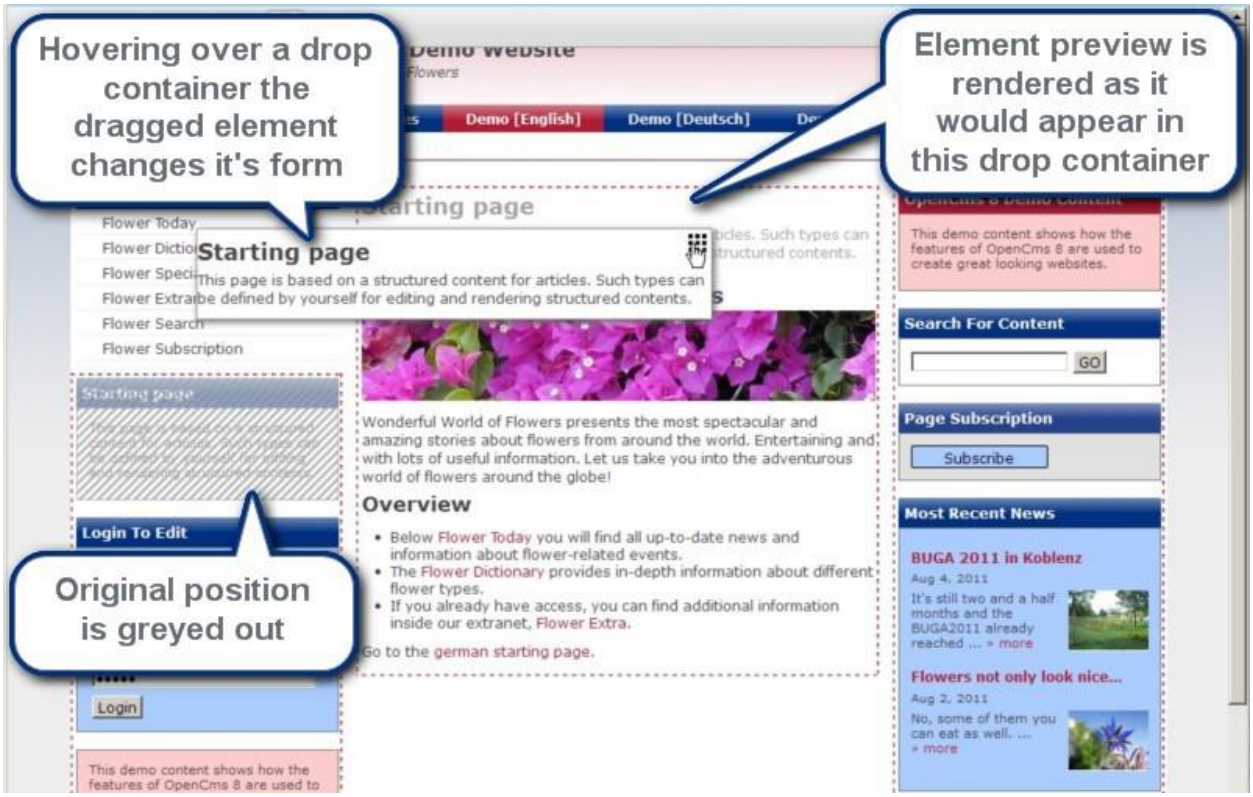
2.3.1 Creating new content

Clicking the **Add content** opens the gallery dialog and shows a list of all possible content types that can be added to the current container page.



This dialog enables the content manager to create new content and also to retrieve existing contents. In order to create a new content he simply drags a content type of his choice onto the page and a new instance of that type is created internally.

While dragging an element those sections are highlighted where the element can be dropped:

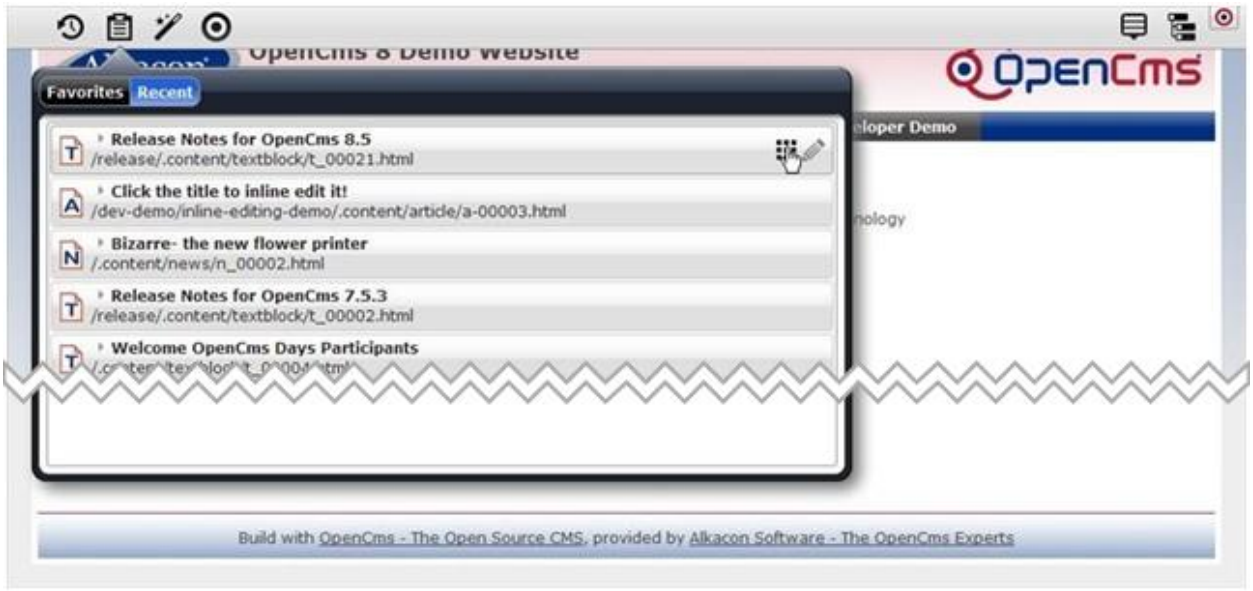


2.3.2 Searching existing content

In order to reuse existing content, click on the **Add content** icon, select one or more resource types and click on the results tab. All elements displayed can be dragged onto the page. You can refine your search by selecting single or multiple criteria in each tab (Types, Galleries, Categories and Advanced). A click on the results tab will refresh the result list containing all contents matching your given criteria.

2.4 Clipboard

By clicking on the on the **Clipboard** icon in the toolbar you can access your personal favorites and a list of elements you have recently used. You can directly add content from the clipboard to your page by drag & drop.



You can add a content element to your favorites by using the **Add to favorites.**

2.5 Context menu 

By selecting the context menu icon from the toolbar you can access:

- **Properties:** Display and edit the OpenCms properties of current container page.
- **Attributes:** Container page attributes, like date created, date last modified, etc.
- **Availability:** Opens the availability dialog
- **Lock report:** Lock state report of current container page
- **Assign categories:** Dialog to assign categories to current container page
- **SEO options:** Search engine optimization
- **Undo changes:** Recovers the last published version of the container page
- **Show workplace:** Open the OpenCms workplace
- **Logout:** Logout from the OpenCms workplace

2.5.1 Properties

To edit the container page's properties, select the **Properties** option from the Context menu. The **newly designed property dialog** allows changing of all properties of the. There are 3 different subsets of properties available: **Basic**, **Individual** and **Shared Properties**. The basic properties are a configurable subset of available properties.

2.5.2 Attributes

The attributes dialog shows next to some general information about the container page more additional metadata including: the page's title, its resource type, file size, resource state (new, changed, deleted, etc.) and the information about who and when the page was created. Moreover it shows the last modification date and the project the page belongs to. The language and the permissions the current user has are display also.

2.5.3 Availability

By selecting the Availability option from the context menu you can set several concerning the container page's online visibility. Available options are:

- **Publish scheduled date**: Using the publish scheduled function, this page will be published automatically on the set date/time.
- **Date released**: By setting the release date, this page will be visible online from the chosen date / time on (if published).
- **Date expired**: By setting the expiration date, this page will be online (if published) until this date / time and the will disappear from the public website but still remains as resource in OpenCms.

Dates can be set by keyboard input or by using the calendar widget that appear, when the date field is clicked.

2.5.4 Lock report

The **Lock report** dialog displays the locking state for the current page. If it has been locked by another user you can retrieve information about the user and the according project here. This dialog is also able to consider inherited locks.

2.5.5 Assign categories

This option opens a comfortable widget for assigning categories to the current page.

2.5.6 SEO options

Pops up a dialog that offers a mask that enables the content manager to set the most relevant properties for SEO of a container page, meaning: Title, Description and Keywords. Additionally it is possible to define aliases for the page. As for rewrite rules you can choose between sending a temporary redirect (302 returned) or sending a permanent redirect (301) or just show the page (200). Already set aliases are displayed below.

2.5.7 Undo changes

Choosing this entry will recover the last published state of the current container page.

2.5.8 Show workplace

To open the OpenCms workplace in a new window, select the **Show workplace** option from the context menu. The OpenCms workplace features more advanced options to manage content and resources and offers a lot of administration options. The workplace should only be accessed by power users meaning administrators, developers or really advanced content managers.

2.6 Publish

After you have finished your work and you want to make your changes visible to the public now, you need to publish what will make the changed content visible in the online version of the website. To publish the edited pages and also other changes you have done, click on the **Publish** icon. The publish list contains all unpublished resources that have been added, modified or deleted by the current user. Unchanged resources are never shown in the publish list. If a resource is locked by another user it will not be shown in the publish list. For a better overview the publish list is split into different sections each representing a user session. It is possible to select **single** resources with a checkbox, or select / deselect **all** displayed resources or select / deselect all resources of a certain **session**. On hovering an option – to **remove** a single resource from the user's personal publish list – appears. If an item was removed from the publish list, it will reappear if the dialog is opened again. Selecting an option will also affect siblings or related resources if existent. This behavior can be turned off/on by the option called "Include related resources" / “Include siblings”.

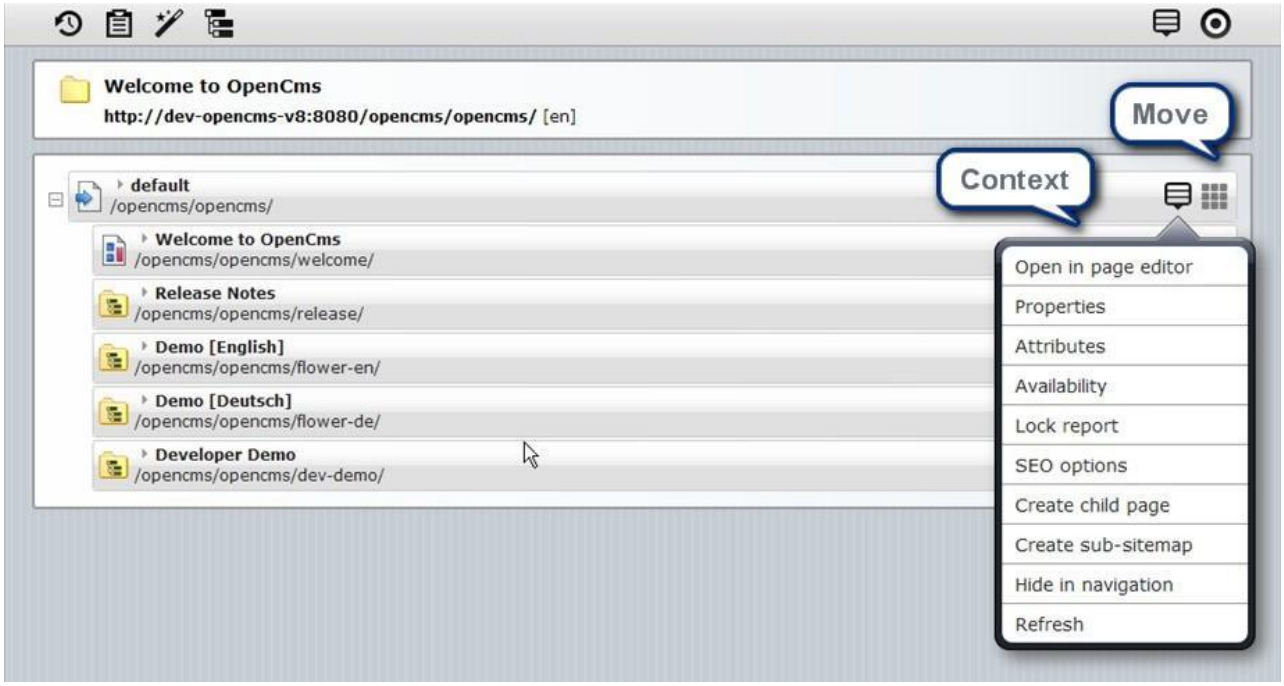
- **Include related resources** will publish all new / changed resources that are related to the original resource (e.g. images or linked resources)
- **Include siblings** will publish resources that are directly link to the original resource and that get changed when the original resource gets changed.

3 Sitemap editor

The **sitemap editor** allows you to build the site's navigation structure by creating new container pages and changing the navigation information of the existing pages. The sitemap represents the website's structure displaying the container page tree. The user can drop new container pages to different levels of the navigation tree, enter further information like title, create new sub sitemaps or change the order of the pages with drag & drop. The sitemap hierarchy describes the navigation structure of the website and is co-responsible for the URLs visible to the public. It approximates the file tree in the *Virtual File System (VFS)* of OpenCms but does not cover it exactly.

3.1 Open the editor

From the page editor click the icon **Sitemap** in the toolbar. The sitemap editor displays the page tree in the order they appear in the navigation. When moving the mouse over a page (sitemap entry) two options are displayed on the right side of the bar: **Context** and **Move**.



3.2 Sitemap editor toolbar

Publish: Opens the publish dialog.

Clipboard: Shows deleted or recently modified pages.

Create page: Enables the content manager to create new pages in the sitemap.

Display all resources: Switches to the VFS explorer view that shows all resources.

Context menu: Shows the context menu for the sitemap.

Edit point: Returns to the page editor.

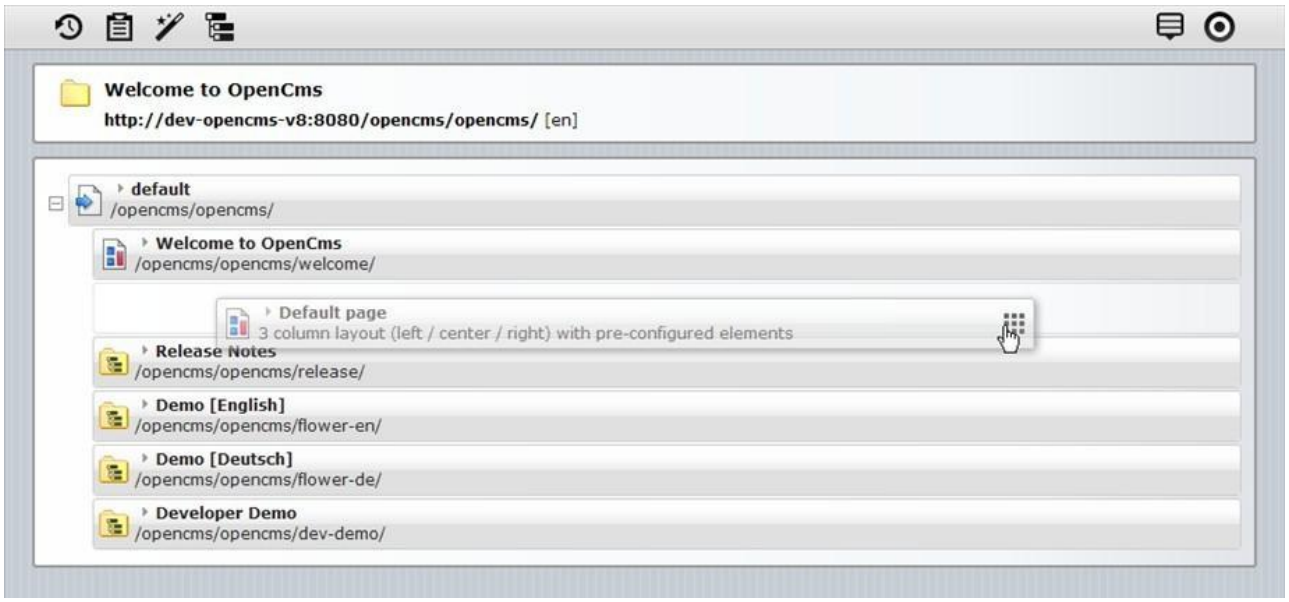
3.3 Publish

3.4 Create page

With the **Create page** option from the sitemap toolbar you can add a new by drag & drop.
The opening dialog features three options: **Container pages, Type pages, Function pages**.

3.4.1 Container pages

Depending on the configured model pages provided by the template designer you can select from different template pages meeting your requirements. You can drag a new container page to wherever you need it in the sitemap.



If the displayed page contains sub-pages itself that are currently not displayed (you might want to unfold them by clicking the (+) more symbol) you can insert the new page after the existing page at the same navigation level. The folder structure can be unfolded while drag & drop by moving the mouse over a folder icon.

3.4.2 Edit model pages

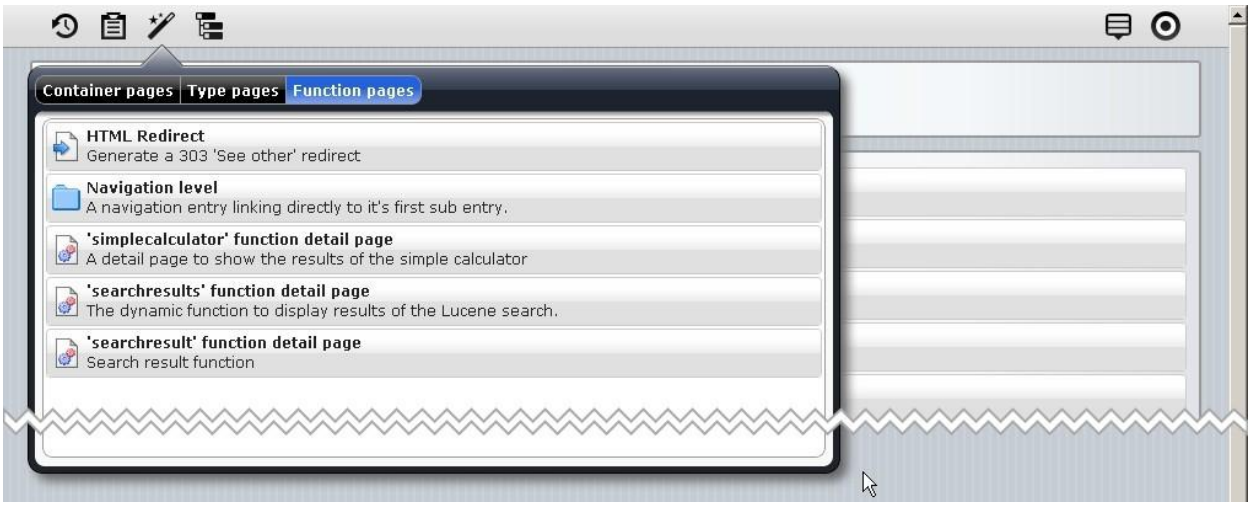
To edit model pages open the **Container pages** tab in the **Create page** dialog and select the **Edit** option from the Context menu of a new container page. You have to confirm this operation since you are about to edit the model for **all new pages**. This will not change already existent pages. This option will open the model page in the page editor, so you can add or remove default elements for new pages. Read more about model pages in section Model pages

3.4.3 Type pages

It is not necessary to create a new page in the sitemap for every instance of a resource type holding content. You should maintain a single detail page to show all contents of a specific resource type. Detail pages are used to link from search results, content collector lists or teaser elements. The detail content gets displayed in the center column of the detail page (depends on template and columns) and can be accessed by an automatically generated URI, built from the detail content's **Title** property. Read more about the type pages in section Detail pages

3.4.4 Function pages

Selecting the **Function pages** option from the create page dialog offers a selection of pages for special purpose:



3.4.4.1 HTML redirect

A HTML Redirect might be accessible through the page navigation and redirects the browser to another URI within the site or to an external link. It might also be excluded from navigation but necessary if a page / sub-site has moved to another location and to prevent dead links from users' bookmarks.

3.4.4.2 Navigation level

Read more about navigation level in section Navigation level

3.4.4.3 Function detail pages

Read more about function detail pages in section Function detail pages

3.5 Clipboard

When selecting the **Clipboard** option from the sitemap toolbar the clipboard opens in an overlay window. It features two lists of pages recently edited in the sitemap editor.

Modified: The modified list will be displayed as default and contains pages that have recently been modified in the sitemap editor. When you move the mouse over a list entry, to the right side of the bar there appears an option to show the page in the sitemap editor. If clicked the overlay window is closed and the selected page will show up blinking in the editor.

Deleted: This list holds all pages that had been deleted from the sitemap without publishing the deletion. Moving the mouse over a list entry will show up an **Undelete** option to the right side of the bar, restoring the original state of the resource before it was deleted.

3.6 Display all resources 

If necessary, the **Display all resources** option will show the site's folder tree and all contained resources in the *Virtual File System* of OpenCms. When this option is selected, the create page dialog as well as drag & drop feature is disabled. The user can browse the file tree and use the Context menu to access other options. Note that this option might display resources that are not necessarily accessible from the site as a user browses it and thus might be of resource types that could not properly be rendered. When displaying all resources the available options for each sitemap element are reduced to Context menu. Moving an element is deactivated.

3.7 Context menu 

The Context menu displays available options the current sub sitemap.

- Show parent sitemap map
- Edit aliases
- Refresh
- Show workplace
- Logout

3.8 Edit point 

The **Edit point** returns to the latest edited page with the Page editor.

3.9 Page types

3.9.1 Model pages

For OpenCms 8 templates you have to create and configure at least one or more model pages in order to enable the **Create page** dialog in the sitemap editor. When dragging a new page into the sitemap tree, the model page is used as master copy to create a new page in the site navigation. The contents dropped to the selected model page automatically appear on all new pages. Typically the template developer creates and configures model pages once he has finished the template and content type implementation but before editing the website's contents.

- Click here to open the model page demo
- Click here to open the model page with element group demo

3.9.1.1 Creation steps

1. Create a new container page in the explorer view.
2. If the template property is not set on the site folder, set the template property accordingly.
3. Create several model pages, one for each template, if the website should have pages with different template configuration.
4. Place the model pages in the following folder of the sub sitemap:
   `/{sitefolder}/{sub sitemap path}/.content/.new/`
5. Open the preview by clicking this page and drop those elements that should be copied when creating a page of this model.

3.9.1.2 Configuration

Configure the model page in the sitemap configuration file:

`/{sitefolder}/{sub sitemap path}/.content/.config`

In tab "Model pages" add a new node for each model page and select its path. Now the configured model pages appear in the list of container pages of the **Create page** dialog in the sitemap editor. In order to display a nice entry for the model, edit its **Title and Description** property. If is set as default this page is also used to create the new page, when it is added over the Context menu in the sitemap editor. Further use the check box of the page model configuration to control the visibility of the inherited page models from the upper sitemaps.

Please note, you can combine the model pages with other techniques like element groups and inheritance groups for better organizing common content parts on the website.

3.9.2 Detail pages

A detail page is a technique to create nice URLs for contents that are not dropped onto a container page but created in the `/.content/` directory, e.g. by using a collector.

- Click here to open the detail page demo

3.9.2.1 Use case

In OpenCms 8, all resources dropped into a container page are automatically created in a directory, which is configured either in the module configuration or in the sitemap configuration. Usually the resource path contains `/.content/` in it. When resources are linked from a list, the editor should not have to drop every linked resource in its own container page to get a nice URL

for it. The idea is to create a single page in the navigation, which is called **detail page**, and to use the URL of this detail page combined with the **Title** property of the resource to generate a resource link without `/.content/` in it.

Generated URL:

```
/dev-demo/collector-with-detail-page/generated_resource_title
```

where `/dev-demo/collector-with-detail-page/` is the path of the detail page and `generated_resource_title` is the url postfix generated from the mapped property.

3.9.2.2 Configuration

The configuration is related to the module containing the template and the resource type definition. Add following mapping in the XML schema definition of the content type. If the type does not have **Title** field, use a field, which comply with a title.

```
<mapping element="Title" mapto="urlName" />
```

Add the attribute `detailview="true"` to the `<cms:container>` tag in the template jsp, where the xmlcontent should be displayed. Usually it is the center column.

```
<cms:container name="centercolumn" type="center" width="450" detailview="true" .../>
```

When using collector to list the resources, surround the resource path `${content.filename}` with `<cms:link>`. The `<cms:link>` takes care of generating the proper url.

```
<%@page buffer="none" session="false" taglibs="c,cms" %>
<div>
   <cms:contentload collector="myCollector" param="..." editable="true">
       <%-- Access the content --%>
       <cms:contentaccess var="content" />
       <c:set var="link"><cms:link>${content.filename}</cms:link></c:set>
       <a href="${link}">${content.value.Title}</a>
   </cms:contentload>
</div>
```

3.9.2.3 Usage

- Open the "Create page" dialog in the sitemap editor.
- From tab "Type pages" drop the page for the resource type that is listed by a collector.
- The **detail page** is automatically configured by OpenCms in the sitemap configuration file.
- Now links should be automatically generated in the collector list.

3.9.3 Function detail page

The function detail page is a detail page mechanism for dynamic functions. This documentation part describes how to create and configure a detail page for a Dynamic function. It uses links referring a locally installed OpenCms version >= 8.5.

- Click here to open the dynamic function demo
- Click here to open the function detail page demo

3.9.3.1 Use case

Collect data in a form, which can appear on several pages of the website. Display the results of the user request on one page in the navigation. For example, integration of a search form on several pages of the website and displaying of the search results on one page in the navigation. Since OpenCms 8.0.3 you can use function detail page to define such result pages.

3.9.3.2 General steps

Following steps are required to create dynamic function and its function detail page:

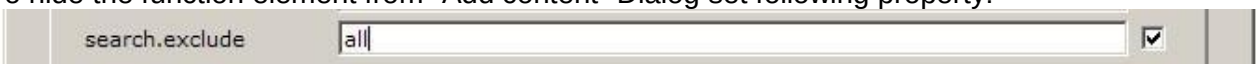
- Create dynamic function element inside the module.
- Create and configure the function detail page for the created dynamic function.
- Drop the dynamic function to container pages.
- Drop the function detail page to the website structure using the sitemap editor.

3.9.3.3 Create dynamic functions

As only users with the role TEMPLATE_DEVELOPER can edit the dynamic function elements, the function should be created inside the module in folder:

`/system/modules/[module name]/functions/`

To hide the function element from "Add content" Dialog set following property:



The drag & drop option is enabled for user with the role WORKPLACE_USERS.

3.9.3.4 Create function detail pages

- Implement the JSP with a form, which should be used on several container pages of the website. Use the following value `${cms.functionDetail['Name of the function detail page']}` as attribute **action** of the `<form>` tag.

```
<%@page buffer="none" session="false" taglibs="c,fn,cms" %>
<div class="box box_schema1">
   <form action="${cms.functionDetail['simplecalculator']}" method="post">  …
   </form>
</div>
```

- Implement another JSP, which evaluates the form data and generates the HTML output for the detail view.
- In explorer view create two dynamic function elements in folder `/system/modules/[module name]/functions/` of the module. One of the dynamic functions should point to the form JSP and another to the detail page jsp.
- Edit the module configuration file `/system/modules/[module name]/.config` and configure the function detail page in the tab "Functions". Add new field "Named function". Define a unique name for the function detail. This should be the same name, which is used in the form JSP as action parameter. After this configuration the function detail page appears in the tab "Function pages" of the "Create page" dialog in the sitemap editor.
- Edit property **container.info** on the template jsp. Set the property value `functionDetail=[container name attribute]`. Use as the value the name attribute of the container, in which the function results should be displayed.

3.9.3.5 Usage

- Drop the dynamic function element with the form into container pages.
- Open the "Create page" dialog of the sitemap editor.
- Use the configured function detail page from tab "Function pages" as page model to add new page to the navigation.

3.9.4 Navigation level

Navigation level is a special navigation folder, which redirects to the first container page inside this folder. Use this option, if you would like to list several sub pages in one navigation level without an overview page.

- Click here to open the navigation level demo

3.9.4.1 Use case

On the website there is a list of product descriptions that should be listed in the navigation under products. When the user selects the folder in the navigation, the first product item is selected directly. There is no overview page. The first item in the list is displayed, too, after the order of the items has been changed. OpenCms 8.5 introduces a new page type **Navigation level** to define a navigation entry point, which automatically redirects to the first sub page of the navigation level.

3.9.4.2 Using the navigation level

- Open the tab **Function pages** in the sitemap editor.
- Drag & drop **Navigation level** into the sitemap.
- Add per drag & drop several sub pages under the new navigation entry.

3.9.4.3 Implementation details

Navigation tag – The `<cms:navigation>` tag fully supports the navigation level. Build the link to the navigation level in a usual manner. The classes `CmsJspNavigationElement` and `CmsJspNavBuilder` automatically recognize the navigation level and directly generate the link to the first item in the list. Nothing else is required.

Styling – The class `CmsJspNavElement` provides a new method to recognize the **Navigation level** in the navigation. This information can be useful e.g. for styling. In following example the css class **current** is used to mark the selected item in navigation. The navigation level element is not marked:

Example

```
<cms:navigation type="treeForFolder" startLevel="1" endLevel="4" var="nav"/>
   <c:forEach items="${nav.items}" var="elem"> […]
   <c:set var="link"><cms:link>${elem.resourceName}</cms:link></c:set>
   <a href="${link}">
      <c:if test="${nav.isActive[elem.resourceName] and !elem.navigationLevel }">
            class="current"
      </c:if> >${elem.navText}</a>        […]
</c:forEach>
```

Please check following JSP for the complete navigation example on github.

3.9.5 Hidden entries

Hidden entries are container pages or other pages which are not visible in navigation menu build with `CmsJspNavBuilder`.

3.9.5.1 Description

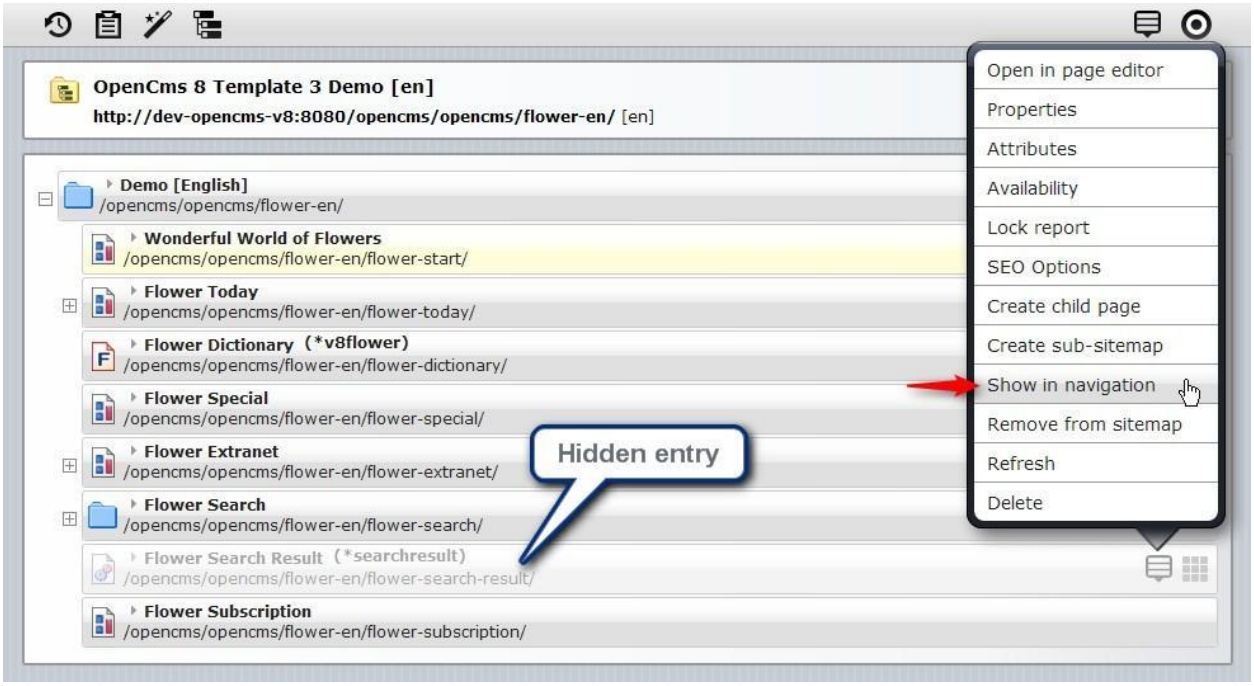
Since OpenCms 8.5 resources, which should not be included in the navigation menu on the website, can be directly hidden in the sitemap editor. The hidden pages are still visible in the sitemap editor and can be edited.

This new feature is supposed to be useful for pages, which are hidden in navigation but

- appear directly under predefined url like contacts, search, location pages etc.
- detail pages for the resources

3.9.5.2 Usage

- Open the sitemap
- Open the Context menu and select "Hide in navigation"
- Use context menu "Show in navigation" to enable the navigation properties for the page

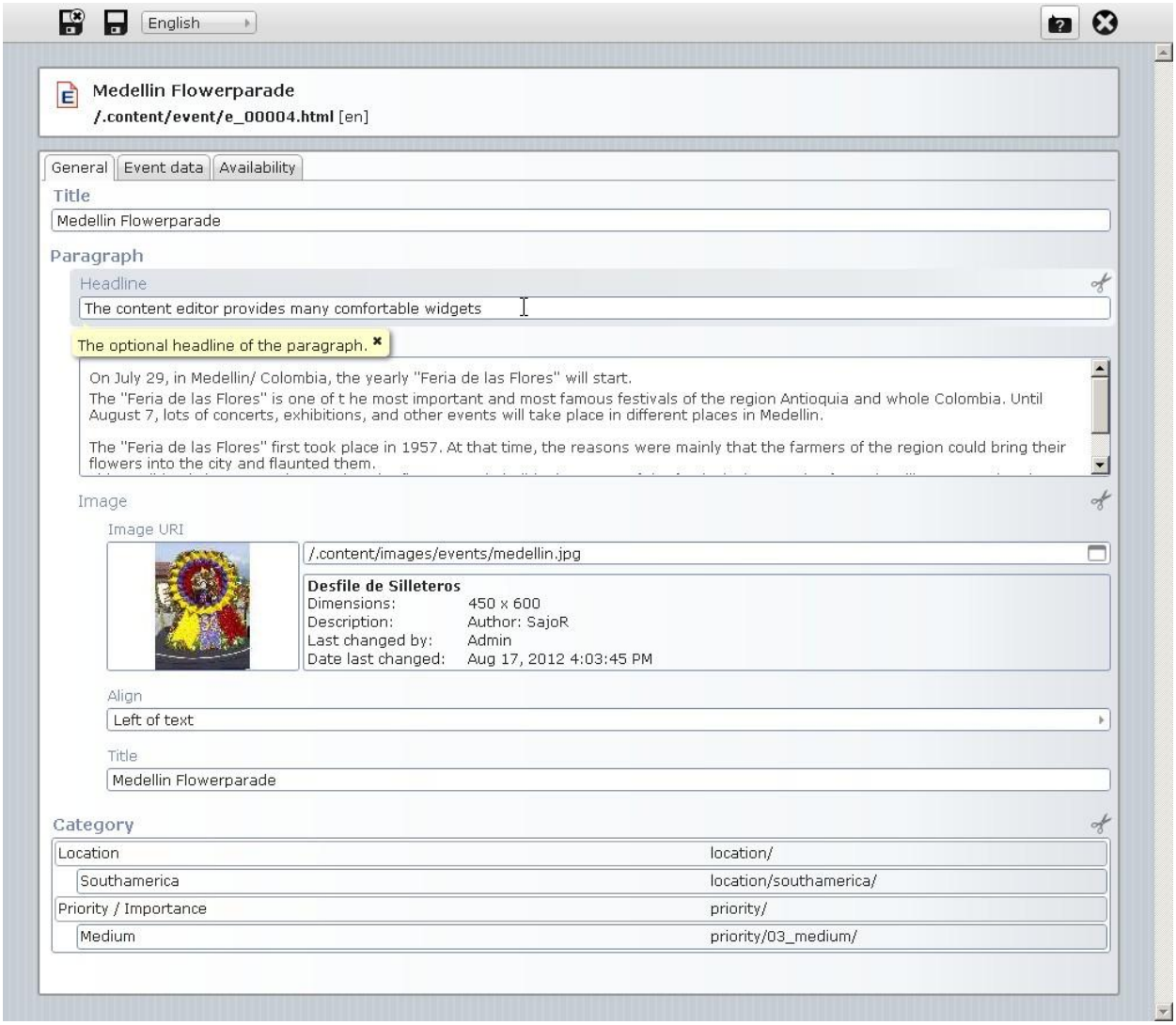


4 Content editor

The content editor in OpenCms 8.5 has been completely rewritten in HTML 5 for ease-of-use and speed. The editor is automatically generated based on XML schemas describing the structure of content types. It provides a rich user interface for the content managers that are used to generate XML files based on the XML schemas. The generated XML is then stored in the OpenCms repository.

Main features of the content editor are:

- Runs on the client side and provides in a highly responsive user experience.
- Automatically generated from an XML schema, do programming required.
- Provides a large selection of rich user interface widgets.
- Can be extended with custom user interface widgets for special uses cases.
- Generates XML files that are validated against the XML schema and stored in the database.



4.1 Widgets

Widgets are used to create a suitable content editor for XML content. The widgets to be used are defined in the XSD (schema definition) of the content type. Compared to the old XML content editor some configuration parameters have been added, changed or removed. The following sections will describe each widget and its parameters with a short example.

Attention: The widget configuration only tells the content editor which GUI component to use for value selection – not the data type to use. E.g. you can select a date with the date picker, but if you don't tell OpenCms to use the type **`OpenCmsDateTime`** it won't be handled as thus. The data type of an element defined in the XSD is responsible for the internal handling of the values. The use of data types increases the data quality what affects not only content retrieval and performance but also the overall data integrity e.g. keeping internal links intact.

In the following sections you will find code snippets showing how to use and configure the distributed widgets. If a specific data type makes sense for the usage of a special widget the code snippets will contain the **`<xsd:element>`**- node, otherwise the examples only show the widgets configuration made in the **`<layout>`**-node assuming that the corresponding element uses **`OpenCmsString`** as type attribute.

Available XML field types

`OpenCmsBoolean` – Represents a Boolean value (TRUE/FALSE)
`OpenCmsCategory` – Can handle one or more selected categories
`OpenCmsColor` – Stores a hexadecimal color value
`OpenCmsDateTime` – Handles a date value
`OpenCmsHtml` – Used for rich text editor fields (keeps internal relations e.g. text links intact)
`OpenCmsLocale` – Stores a locale value
`OpenCmsPlainTextString` – Stores the extracted text for a field containing HTML
`OpenCmsString` – Stores a simple String 1:1
`OpenCmsVarLink` – Can handle internal and external at once
`OpenCmsVfsFile` – Designated for storing internal file references
`OpenCmsVfsImage` – Refers internal images (handles image format, scaling, description …)

4.1.1 String widget

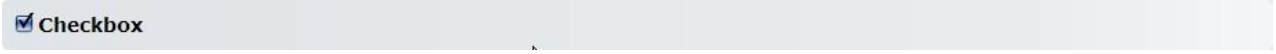
The **String widget** provides a simple text input field to enter plain text that is stored as String value internally.

This widget does not need any configuration and can be defined as follows:

```
<xsd:element name="Input" type="OpenCmsString" />
```

4.1.2 Boolean widget

The Boolean widget provides a checkbox and stores a Boolean value internally.



This widget does not need any configuration and can be defined as follows:

```
<xsd:element name="Checkbox" type="OpenCmsBoolean" />
```

4.1.3 Display widget

The display widget only shows a value without the possibility of modification. The value this widget displays configured as attribute inside its layout node:

A configuration could look like:

```
<layout element="Di" widget="DisplayWidget" configuration="This is no default string"/>
```

4.1.4 Select widget

The select widget provides a drop down with a set of options.

The layout for this widget could look like:

```
<layout element="Select" widget="SelectorWidget" configuration="1*|2|3|4|5|6|7" />
```

Read section select widget configuration to get more information.

4.1.5 Text area widget

The text area widget provides a HTML text area input field that can be scrolled if the text is too large to be displayed completely. The field can also be resized within the editor.

In the configuration parameter you can define the count of rows the text area should have when it is opened initially. If no configuration is set, the default count of rows of fife is taken.

```
<layout element="Textarea" widget="TextareaWidget" configuration="7" />
```

4.1.6 Radio button widget

The radio button widget provides a group of radio buttons for single value selection.

The configuration could look like:

```
<layout element="Radiobutton" widget="RadioSelectWidget"
        configuration="Radiobutton1*|Radiobutton2|Radiobutton3|
                       Radiobutton4|Radiobutton5|Radiobutton6" />
```

The configuration syntax is identical to the select widget configuration.

4.1.7 Multi select widget

The multi select widget provides a group of check boxes to select one or more value.

The configuration could look like:

```
<layout element="Multi" widget="MultiSelectWidget" configuration="1*|2*|3*|4|5|6|7"/>
```

The syntax is identical to the select widget configuration.

4.1.8 Combo widget

A combo widget offers a pre-defined set of select options with the ability for individual text input.

The configuration could look like:

```
<layout element="Combobox" widget="ComboWidget" configuration="1|2|3|4|5|6|7" />
```

Its syntax is identical to the select widget configuration.

4.1.9 Resource type combo widget

This special combo box widget provides a list of all resource types configured for OpenCms.

This widget does not need a configuration String and the layout node should look like:

```
<layout element="TypeCombo" widget="TypeComboWidget" />
```

4.1.10 HTML widget

The HTML widget uses the TinyMCE that has been extended for a lot of special CMS features.

Options can be defined for each element of the type OpenCmsHtml using the widget HtmlWidget. They have to be placed in the annotation section of a XSD. The configuration attribute in the layout must contain the activated options as a comma separated String value:

```
<xsd:element name="Text" type="OpenCmsHtml" />

<layout element="Text" widget="HtmlWidget"
        configuration=" link,anchor,imagegallery,downloadgallery,formatselect"/>
```

Available options are:

- anchor: the anchor dialog button
- buttonbar:${button bar items, separated by ';'}: an individual button bar configuration.
- css:/vfs/path/to/cssfile.css: the absolute path in the OpenCms VFS to the CSS style sheet to use to render the contents in the editor (availability depends on the integrated editor)
- formatselect: the format selector for selecting text format like paragraph or headings
- formatselect.options:${list of options, separated by ';'}: the options that should be available in the format selector, e.g. formatselect.options:p;h1;h2
- fullpage: the editor creates an entire HTML page code
- ${gallerytype}: Shows a gallery dialog button, e.g. imagegallery displays the image gallery button or downloadgallery displays the download gallery button
- height:${editorheight}: the editor height, specified in ‘px’ or ‘%’, e.g. 400px
- hidebuttons:${list of buttons to hide, separated by ';'}: the buttons to hide that usually appear in the default button bar, e.g. hidebuttons:bold;italic;underline;strikethrough hides some formatting buttons
- image: the image dialog button (availability depends on the integrated editor)
- link: the link dialog button
- source: shows the source code toggle button(s)
- stylesxml:/vfs/path/to/stylefile.xml: the absolute path in the OpenCms VFS to the user defined styles that should be displayed in the style selector (availability depends on the integrated editor)
- stylesformat:/vfs/path/to/stylefile.xml: the absolute path in the OpenCms VFS to the user defined styles format that should be displayed in the style selector (availability depends on the integrated editor)
- table: the table dialog button (availability depends on the integrated editor)

Some configurations like the button bar items should be defined in the global widget configuration in the opencms-vfs.xml.

4.1.11 Localization widget

The localization widget provides a standard HTML form input field for overwriting localized values of a resource bundle. The resource bundle is configured with the widget configuration attribute. An optional key name to look up in the bundle can be given, too, in case it is different from the element name: key=mykey.

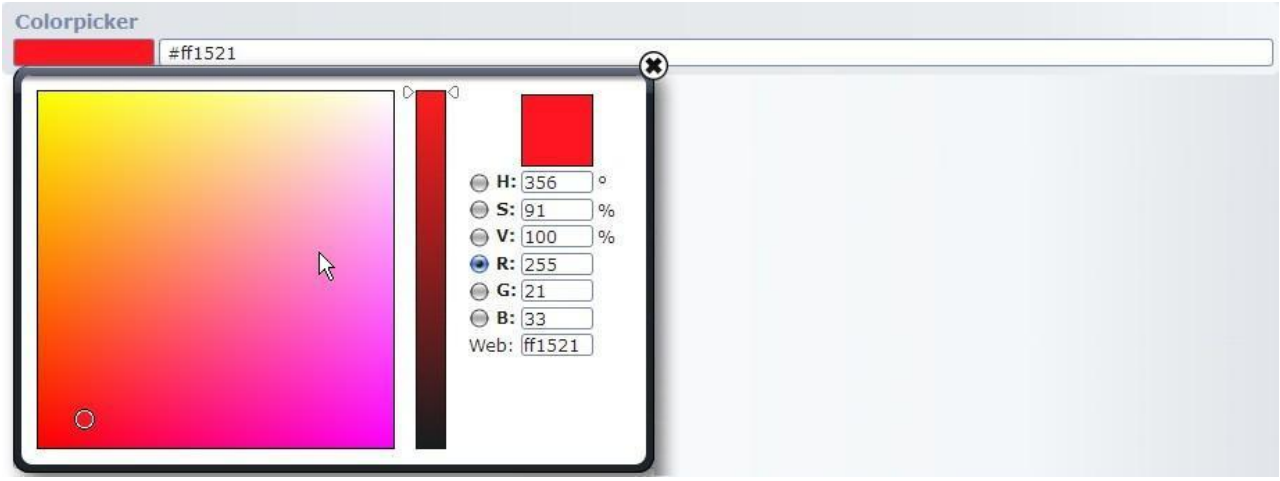
The locale to get the value for can be configured, by adding a configuration directive: locale=en.

```
<layout element="Localization" widget="LocalizationWidget"
        configuration="org.opencms.workplace.messages|key=mykey|locale=en" />
```

To use the stored localization values and have the values of the resource bundles as fallback, use the CmsXmlMessages object.

4.1.12 Color picker widget

With the color picker widget you can choose a color.



This widget does not need any configuration, just use the element type `OpenCmsColor`

```
<xsd:element name="Colorpicker" type="OpenCmsColor" />
```

4.1.13 Date picker widget

The date picker widget is an easy way specifying dates.

This widget does not need any configuration, just use the type `OpenCmsDateTime`

```
<xsd:element name="Calendar" type="OpenCmsDateTime" />
```

4.1.14 Category widget

The category widget provides a category selection in a comfortable way.

This widget has some optional configuration parameters:

- **category**: Path to the root category to restrict the shown categories to its children.
- **onlyleafs**: Starts the selection in a list view

- **parentSelection**: Signals if to store the parents of the selected categories also

The category widget should be assigned to elements of the type `OpenCmsCategory`, what will enable to store the multiple selections. For legacy reasons it is still possible to layout elements of the type `OpenCmsVfsFile` with the category widget, but then you won't be able to make a multi selection. A configuration could look like:

```
<xsd:element name="Category" type="OpenCmsCategory" />

<layout element="Category" widget="CategoryWidget"
    configuration="category=onlyleafs=false|parentSelection"/>
```

4.1.15 Group widget

The group widget is a special select box that offers o list of groups within OpenCms.

This widget does not need any configuration and can be configured like this:

```
<layout element="Group" widget="GroupWidget" />
```

4.1.16 Multi group widget

With the multi group widget you can select multiple groups at once.

This widget is configurable with the following options:

- **groupfilter**: regular expression to filter available groups
- **groups**: comma separated list of group names to show in the select box. Please note, if this configuration option is used, groupfilter and includesubous are not considered anymore.
- **includesubous**: boolean flag to indicate if sub OUs should be scanned for groups to select
- **oufqn**: the fully qualified name of the OU to read the groups from

To map the selected group to a permission to set, use the following mapping configuration:

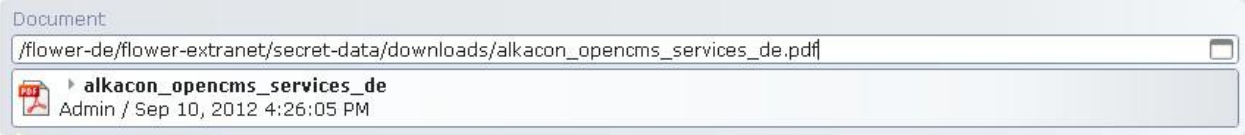
```
<mapping element="..."
mapto="permission:GROUP:+r+v|GROUP.ALL_OTHERS:|GROUP.Projectmanagers:+r+v+w+c" />
```

This means that the +r+v permission is written for the principal GROUP on the resource. Additionally two permissions are written as default: for ALL_OTHERS, no allowed permission is set, for Projectmanagers, "+r+v+w+c" is set.
This widget does not need any configuration and can be this way:

```
<layout element="GroupMulti" widget="GroupMultiSelectorWidget" />
```

4.1.17 VFS file widget

The VFS file widget provides a selection of a file in the VFS. It either can be typed in directly or be choosing with the gallery. This gallery will come up by pushing the button on the right side of the input field. A detail description of the popup can you find here: "GallerySelection"



This widget has this configuration options:

- **hidesiteselector, showsiteselector**: The site selector is hidden or shown (default)
- **excludefiles, includefiles**: Files are hidden in the popup tree or shown (default)
- **notprojectaware, projectaware**: To show (default: not) only files of the current project
- **startsite**: The site the popup tree should be opened with

Configuration example:

```
<xsd:element name="VfsFile" type="OpenCmsVfsFile" />
<layout element="VfsFile" widget="VfsFileWidget"
        configuration="startsite=/sites/www.domain.org/repository|hidesiteselector" />
```

4.1.18 VFS image widget

The VFS image widget allows image and offers a couple of functions like resizing, cropping, etc.

Configuration

The configuration has to be formatted as JSON object, with the following possible keys:

- **class**: optional class name that implements a dynamic startup configuration and special format values, must be a full qualified class name
- **formatnames**: list of format names to select, with pairs of selectable value and selectable text, e.g. value1:optiontext1|value2:optiontext2
- **formatvalues**: corresponding format values to the format names list, can be dynamically generated by the dynamic configuration class. The list of values should contain width and height information, with a '?' as sign for dynamic size and with an 'x' as separator for width and height. Example: ['200x?', '800x600']
- **scaleparams**: default scale parameters (no width, height or crop should be provided!)
- **startup**: the startup folder, can be dynamically generated by the provided class, in that case, use 'dynamic' as value
- **type**: the startup folder type, can be 'gallery' or 'category'. Can be dynamically generated by the provided class, in that case, use 'dynamic' as value
- **usedescription**: indicates if the description input field for the image should be
- **useformat**: indicates if the format select box for the image should be shown or not

An example configuration JSON:

```
{scaleparams: 'q:70,r:2,c:CCCC00',
    type: 'gallery',startup: '/demo_en/images/',usedescription: true,useformat: true,
    formatnames: 'imageleft:Image left|imageright:Image right|imagetop:Image top',
    formatvalues: ['150x?', '250x300', '?x250']}
```

Complete example for configuring the VFS image widget a XSD:

```
<xsd:element name="VfsImage" type="OpenCmsVfsImage" />
<layout element="Widget" widget="VfsImageWidget" configuration="
    {useformat:true,usedescription:true,formatnames:'left:Left|right:Right|top:Top',
     ['150x?','250x300','?x250']}"/>
```

4.1.19 Image gallery widget

The image gallery widget provides a selection of a file in the image gallery. This gallery will come up by pushing the button on the right side of the input field.

The configuration options are read from the configuration String of the widget. For nested XML schemas the configuration String must be defined inside the nested content. The configuration String has to be formatted as JSON object, with the following possible keys:

- **class**: optional class implementing dynamic startup configurations and format values
- **startup**: the startup folder, set to ‘dynamic’ if you want to use the custom class
- **type**: can be 'gallery' or 'category', set to ‘dynamic’ if you want to use the custom

XSD configuration example:

```
<xsd:element name="ImageGallery" type="OpenCmsVfsFile" minOccurs="1"/>

<layout element="ImageGallery" widget="ImageGalleryWidget"
          configuration="{type: 'gallery', startup: '/demo_en/images/'}"/>
```

4.1.20 Download gallery widget

The download gallery widget provides the selection of VSF resources.

The configuration options are read from the configuration String of the widget. For nested XML schemas the configuration String must be defined inside the nested content. The configuration String has to be formatted as JSON object, with the following possible keys:

- **class**: optional class implementing dynamic startup configurations and format values
- **startup**: the startup folder, set to ‘dynamic’ if you want to use the custom class
- **type**: can be 'gallery' or 'category', set to ‘dynamic’ if you want to use the custom

Example configuration:

```
<xsd:element name="DownloadGallery" type="OpenCmsVsfFile" minOccurs="1"/>

<layout element="DownloadGallery" widget="LegacyDownloadGalleryWidget"
        configuration="{type: 'gallery', startup: '/demo_en/images/'}"/>
```

4.1.21 HTML gallery widget

The HTML gallery widget provides a selection of a file in the HTML gallery. This gallery will come up by pushing the button on the right side of the input field.

The configuration options are read from the configuration String of the widget. For nested XML schemas the configuration String must be defined inside the nested content. The configuration String has to be formatted as JSON object, with the following possible keys:

- **class**: optional class implementing dynamic startup configurations and format values
- **startup**: the startup folder, set to ‘dynamic’ if you want to use the custom class
- **type**: can be 'gallery' or 'category', set to ‘dynamic’ if you want to use the custom

Example configuration

```
<layout element="HtmlGallery" widget="HtmlGalleryWidget"
        configuration="{type: 'gallery', startup: '/demo_en/html/'}"/>
```

4.1.22 Table gallery widget

The table gallery widget provides a selection of a file in the table gallery. This gallery will come up by pushing the button on the right side of the input field.

The configuration options are read from the configuration String of the widget. For nested XML schemas the configuration String must be defined inside the nested content. The configuration String has to be formatted as JSON object, with the following possible keys:

- **class**: optional class implementing dynamic startup configurations and format values
- **startup**: the startup folder, set to ‘dynamic’ if you want to use the custom class
- **type**: can be 'gallery' or 'category', set to ‘dynamic’ if you want to use the custom

Example configuration:

```
<layout element="TableGallery" widget="TableGalleryWidget"
        configuration="{type: 'gallery', startup: '/demo_en/table/'}"/>
```

4.1.23 Link gallery widget

The link gallery widget provides a selection of a file in the link gallery. This gallery will come up by pushing the button on the right side of the input field.

The configuration options are read from the configuration String of the widget. For nested XML schemas the configuration String must be defined inside the nested content. The configuration String has to be formatted as JSON object, with the following possible keys:

- **class**: optional class implementing dynamic startup configurations and format values
- **startup**: the startup folder, set to ‘dynamic’ if you want to use the custom class
- **type**: can be 'gallery' or 'category', set to ‘dynamic’ if you want to use the custom

```
{type: 'gallery', startup: '/demo_en/link/'}
```

Example of the xsd declaration:

```
<xsd:sequence>
    <xsd:element name="LinkGallery" type="OpenCmsVfsFile" minOccurs="1"/>
    […]
</xsd:sequence>
[…]
<layouts>
    <layout element="LinkGallery" widget="LinkGalleryWidget"
          configuration="{type: 'gallery',
          startup: '/demo_en/link/'}"/>
</layouts>
```

4.2 Select widget configuration

If options are passed from XML content schema definitions as widget configuration options, the following syntax is used for defining the option values:

```
value='{text}' default='{true|false}' option='{text}'
help='{text}|{more option definitions}
```

For example:

```
value='value1' default='true' option='option1' help='help1'|value='value2'
option='option2' help='help2'
```

The elements default, option and help are all optional, only a value must be present in the input. There should be only one default set to true in the input, if more than one is detected, only the first default found is actually used. If no option is given, the value of option defaults to the value of the given value. If no help is given, the default is null.
Shortcut syntax options:
If you don't specify the value key, the value is assumed to start at the first position of an option definition. In this case the value must not be surrounded by the ' chars. Example: `value='some value' default='true'` can also be written as some value `default='true'`.
Only if you use the short value definition as described above, a default value can be marked with a * at the end of the value definition. Example: `value='some value' default='true'` can also be written as `some value*`.
Only if you use the short value definition as described above, you can also append the option to the value using a :. In this case no ' must surround the option. Please keep in mind that in this case the value itself can not longer contain a : char, since it would then be interpreted as a delimiter. Example: `value='some value' option='some option'` can also be written as `some value:some` option.
Any combinations of the above described shortcuts are allowed in the configuration option String. Here are some more examples of valid configuration option Strings:

```
1*|2|3|4|5|6|7
1 default='true'|2|3|4|5|6|7
value='1' default='true'|value='2'|value='3'
value='1'|2*|value='3'
1*:option text|2|3|4
1* option='option text' help='some'|2|3|4
```

Please note: If an entry in the configuration String is malformed, this error is silently ignored (but written to the log channel of this class at INFOlevel.

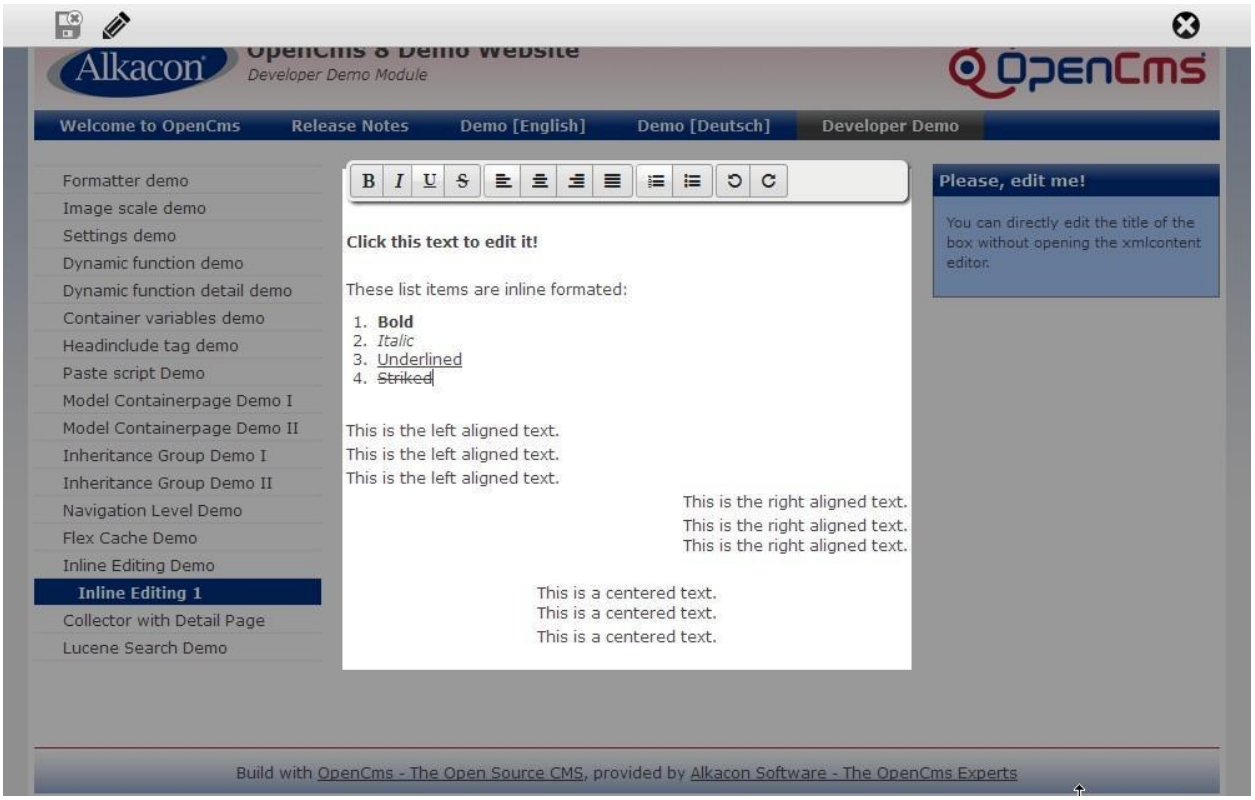
4.3 Implement Custom Widgets

If the provided widgets do not fulfill your entire requirements, it is possible to implement custom widgets for the content editor. The module named **`'org.opencms.dev.demo.customwidget'`** that is shipped with the standard OpenCms distribution shows how to implement custom widgets for the GWT based content editor without the need to write/compile any line of GWT code.
The server side implementation is located at:
`org.opencms.dev.demo.customwidget.CustomWidget`
and the client side implementation is you will find in the VFS at:
`/system/modules/org.opencms.dev.demo.customwidget/resources/mywidget.js`

5 Inline editor

Inline editing is a new feature introduced with OpenCms 8.5, which improves the user experience during editing of page contents. The inline editing enables the content manager to change the contents directly on the preview page, without displacing any content element. The form based content editor can still be used as usual for editing the complete content of the resource. When the cursor is hovering text, which can be edited inline, an edit cursor appears:

Clicking the inline editable field, the text is highlighted and optionally a format toolbar comes up:



- Click here to open inline editing demo

5.1 Fields supporting inline edit

Inline editing can be enabled for content fields of the type OpenCmsString and HtmlWidget. We intend to extend this list for the future versions.

5.2 Configuration

Inline editing is configured inside a formatter JSP. Each formatter of a resource type defines its own inline edit behavior. E.g. an article can be dropped to the center column as well as to the side column, but the editor should only be able to edit the article in the center column. To achieve this only the formatter used to render the article in the center column container should enable the inline editing.

5.2.1 Formatters

There is an xmlcontent with a field `Title`. The inline editig feature for this xmlcontent field can be configured in the formatter JSP in only two steps:

1 To enable the inline editing add a attribute `rdfa="rdfa"` to `<cms:formatter>` tag.

```
<cms:formatter var="content" val="value" rdfa="rdfa">
```

2 Add `${rdfa.fieldName}` as attribute to the element, which surrounds the value.

```
<span ${rdfa.Text}>${value.Text}</span>
```

5.2.2 Nested contents

With EL the nested content field can be accessed like in following example:

```
<cms:formatter var="content" val="value" rdfa="rdfa">
   <c:forEach items="${content.valueList.Paragraph}" var="paragraph">
       <h2>${paragraph.value.Headline}</h2>
       <span>${paragraph.value.Text}</span>
   </c:forEach>
</cms:formatter>
```

`${content.valueList.Paragraph}` return a list of `CmsJspContentAccessValueWrapper` objects `${paragraph.value.Headline}` and ${paragraph.value.Text} access the values of the nested fields. Just replace the `.value` with rdfa to get the css attributes required for inline editing. To enable the inline editing add `${paragraph.rdfa.Headline}` and `${paragraph.rdfa.Text}` to the HTML tag, which surround the field content:

```
<cms:formatter var="content" val="value" rdfa="rdfa">
   <c:forEach items="${content.valueList.Paragraph}" var="paragraph">
       <h2 ${paragraph.rdfa.Headline}>${paragraph.value.Headline}</h2>
       <span ${paragraph.rdfa.Text}>${paragraph.value.Text}</span>
   </c:forEach>
</cms:formatter>
```

5.2.3 Details

The `rdfa` attribute of the `<cms:formatter>` tag is used in a similar manner to the `val` attribute. Assume the content has a field with the name `Title`. In EL with `${rdfa.Title}` you get the specific css attributes, which are required to enable the inline editing for this content field. These automatically generated attributes has to be set as attribute on the HTML element, which surrounds the field content. Note that the inline editing feature cannot be applied to field contents, that are manipulated in a JSP e.g. with `cms:stripHtml()` or `cms:trimToSite()`.

6 Element group

An element group is a new content element in OpenCms 8, which references a group of other content elements. Click here to open the element group demo

6.1 Description

An element group is a content element similar to Inheritance group, which allow the user to drop other elements in it. The user can edit, delete or replace the elements inside the group as well as changing their order. All changes on the element group affect all container pages to which the element group is dropped. So the element group allows maintaining the referenced elements in one place to take effect on many pages.

6.2 Combination with model pages

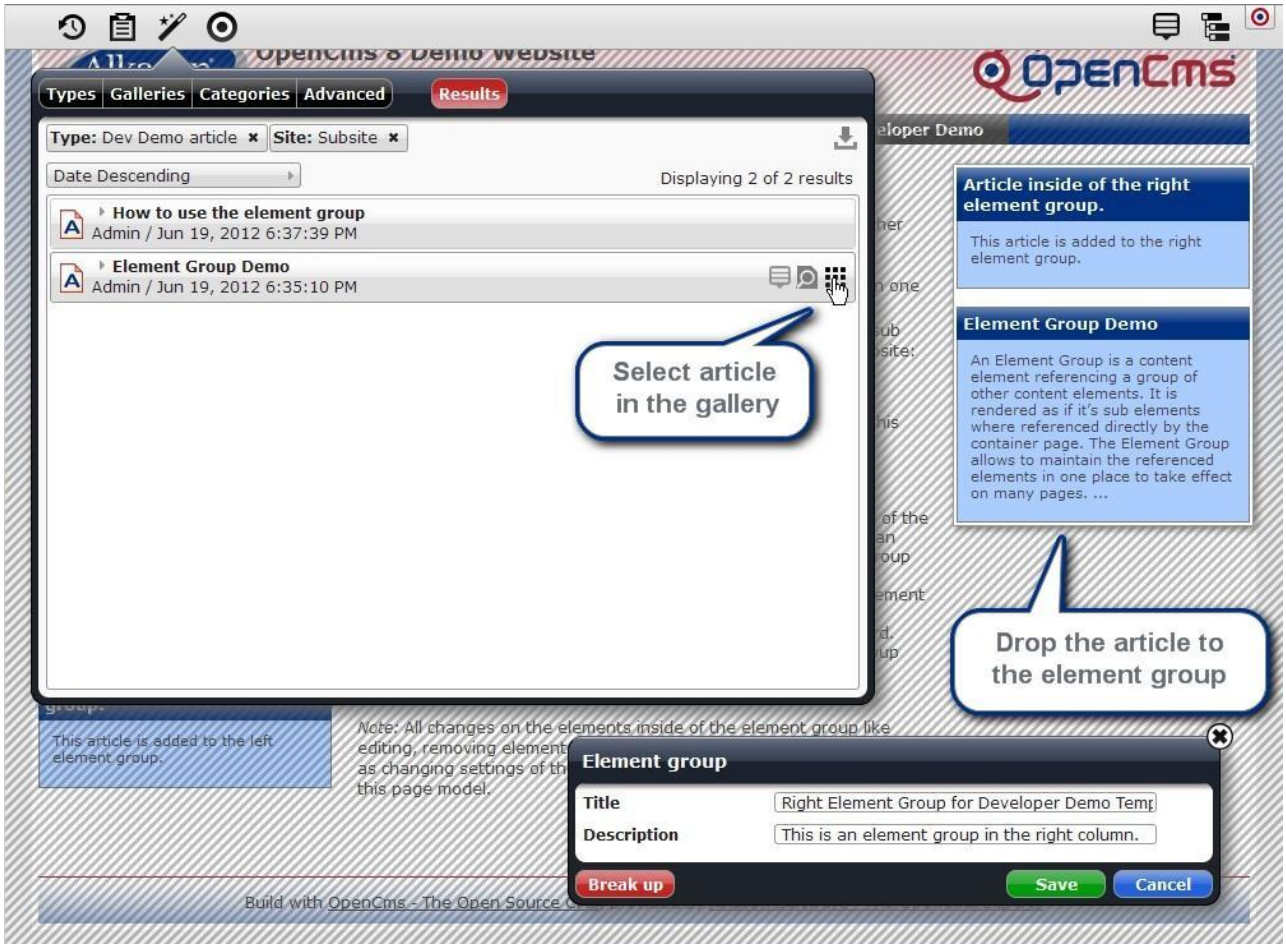
The most popular use case for element group is its combination with the model page. Drop the element group to the appropriate container in the model page and organize all elements inside the element groups, if you would like to have following effects on the site:

- Common content parts appear on all created pages, e.g. header, footer, side columns.
- All future changes in these containers effect all pages using this page model, for example:
  - editing
  - dropping new elements
  - deleting elements
  - changing order of elements

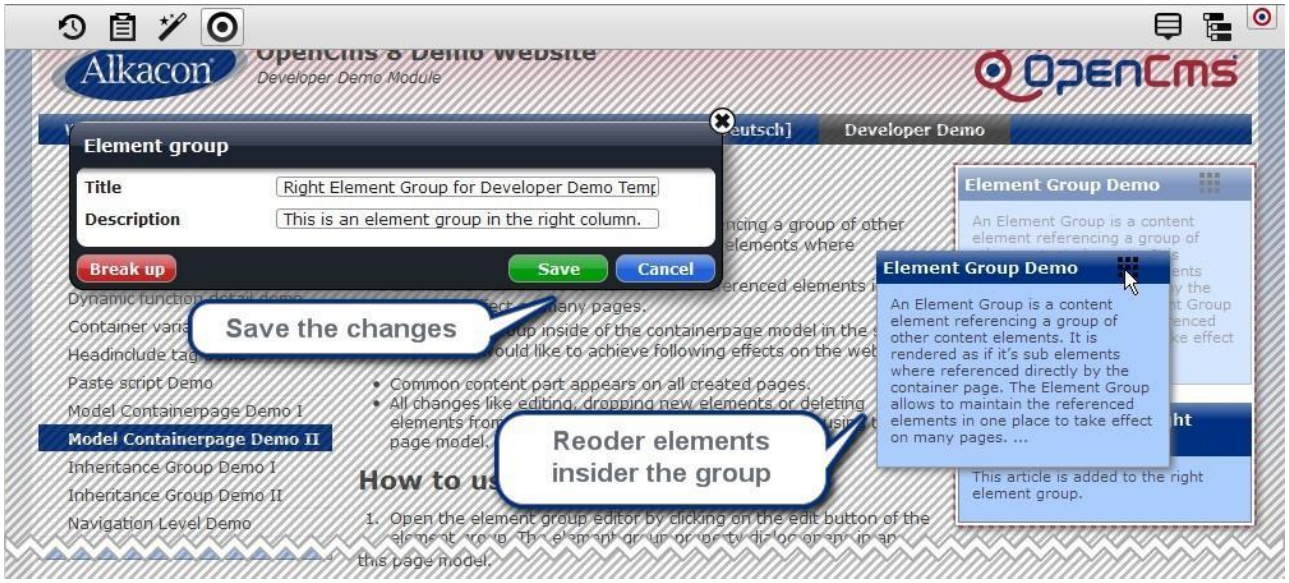
6.3 Using the element group

- Drop an element group to container page like any other content from the "Add content" dialog.
- Open the element group editor by clicking on the edit symbol. The element group dialog opens in an overlay, where the title and the description of the element group can be edited.
- During the element group is edited the ADE toolbar is active and can be used as usual.

- Drag and drop new elements from the "Add content" dialog to the element group.



- Edit or delete the existing elements of the element group.
- Reorder the elements inside the group.
- Save all changes by clicking the ok-button of the element group editor.



All changes on the elements inside of the element group like editing, removing elements, moving elements insider the group as well as changing settings of the element are populated to all pages using this page model.

7 Inheritance group

Inheritance groups are a new content type in OpenCms 8.5 for use in container pages.
Click here to open the inheritance-group demo

7.1 Description

An inheritance group is similar to an element group in that they allow you to drop a set of content elements into your page as a single object. But element groups don't allow you to change the set of elements only for a specific subset of pages; when you make a change to an element group, this change is visible across all pages using that element group. This is the problem that Inheritance groups are meant to solve:
When changing an Inheritance group on a page, the content of the same group will only be changed on the page and any pages which are descendants of the page's parent folder; In other words, changes will be "inherited" by the group in child pages, but nowhere else. For example, you may want to define an Inheritance group for the right column of your template, define some common content which you want to be visible everywhere on your site, and then add more specific contents for each subsection of your site.

7.2 Basic definitions

We will call a container page a.html a **descendant page** of another container page b.html if the parent folder of a.html is a direct or indirect subfolder of the parent folder of b.html. Conversely, we call b.html an **ancestor page** of a.html. When editing a container page, we call the parent folder of that page the *current folder*. This is important because Inheritance group data is attached to folders, not to individual container pages. So container pages in the same folder can't have different content for the same Inheritance group.

7.3 Usage

7.3.1 Creating new Inheritance groups

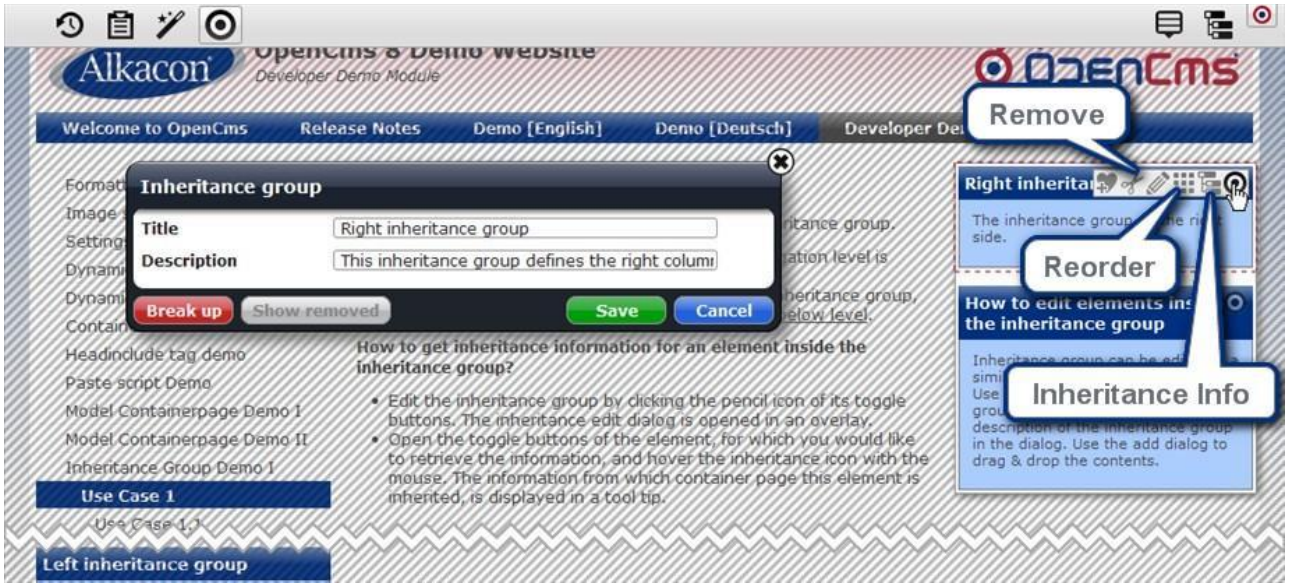
You can create a new Inheritance group in the container Page editor by opening the galleries menu and dragging the item "Inheritance group" from the "Types" tab onto your page. As with other resources, the Inheritance group will only be created in the VFS once you edit it.

7.3.2 Using existing Inheritance groups

You can use an existing Inheritance group on your container page by opening the galleries menu, selecting the type "Inheritance group" from the
Once created, Inheritance groups can be freely dropped onto container pages.

7.3.3 Changing Inheritance groups

To edit an Inheritance group, first you need to open a container page that contains the Inheritance group. Hover over the **Edit point** of the Inheritance group and click the edit symbol. Now the Inheritance group editor will pop up. You can now perform various actions:

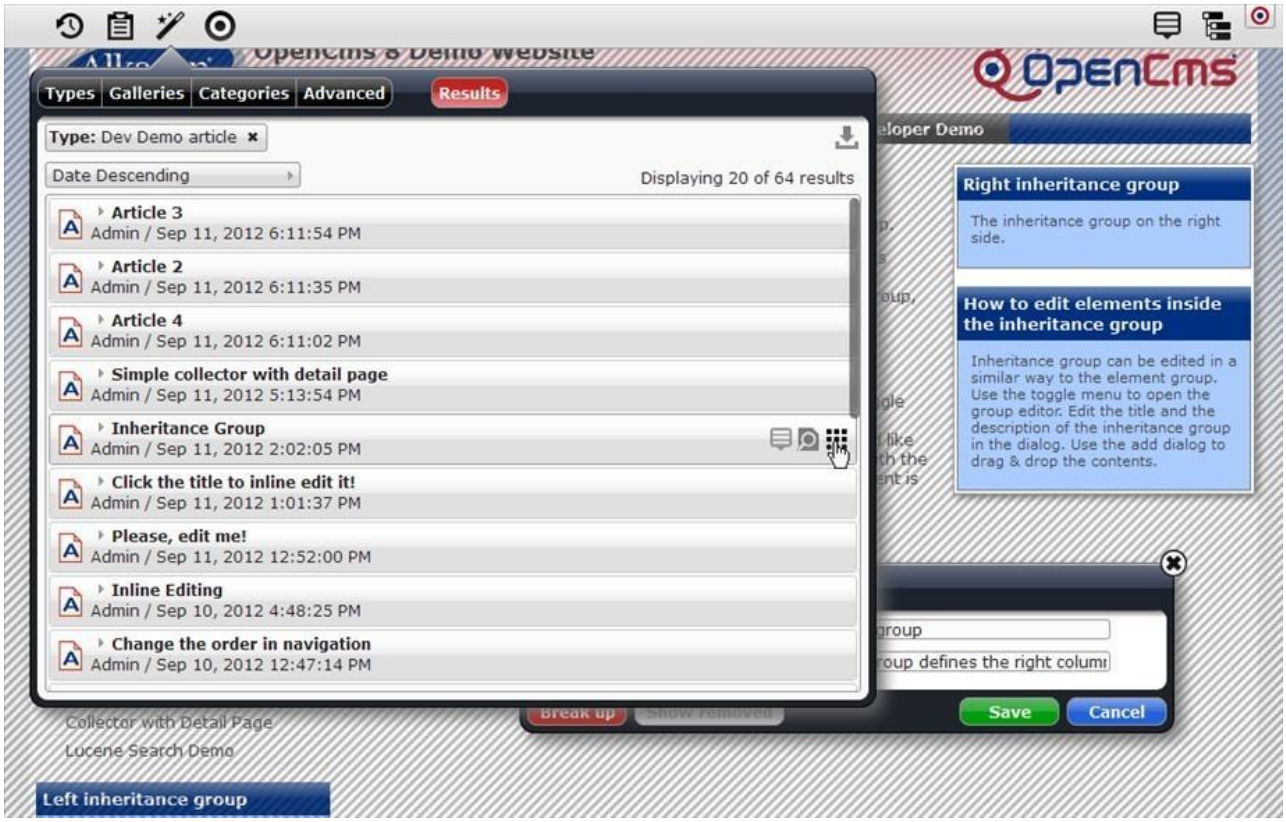


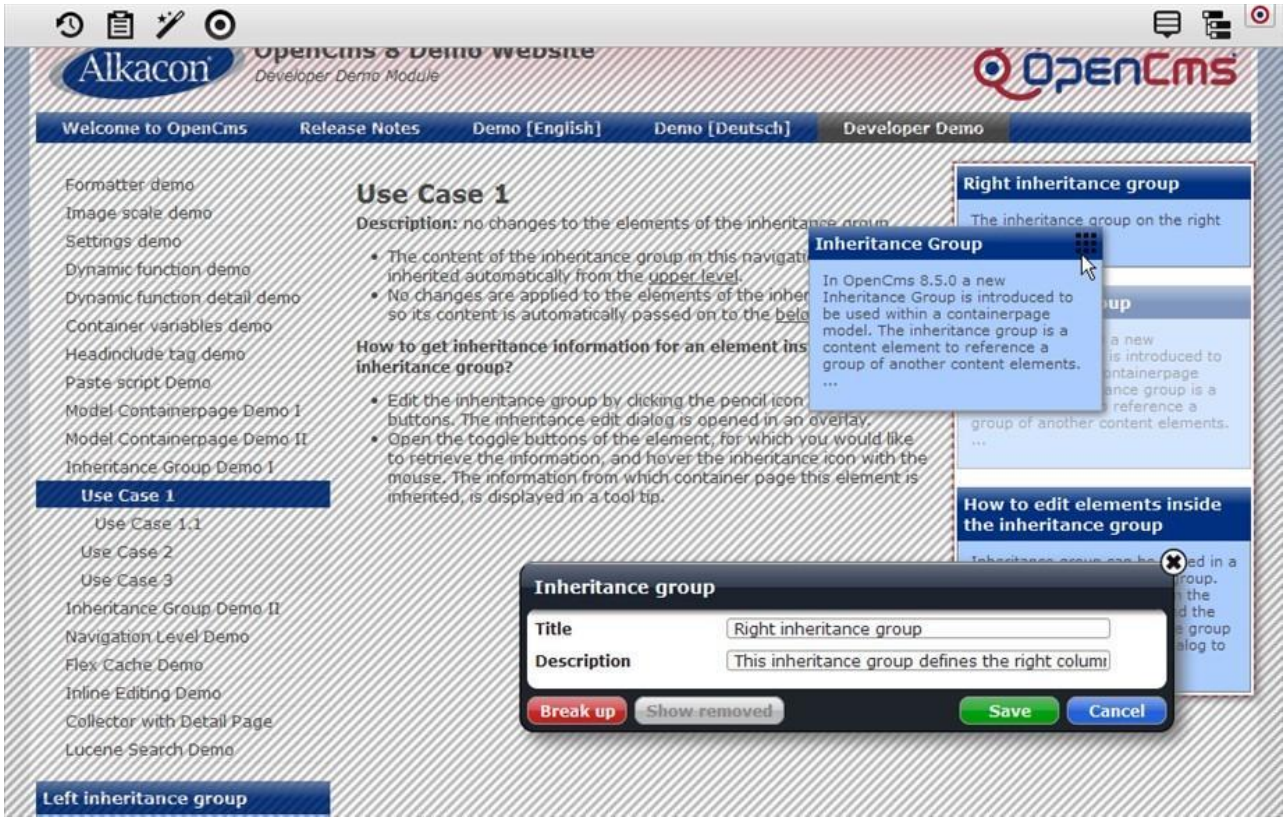
7.3.3.1 Changing the title and description

Using the text fields in the dialog, you can change the title and description of the Inheritance group. The title and description are not displayed when rendering the Inheritance group, but are mainly used for finding the Inheritance group using the galleries dialog.

7.3.3.2 Adding content elements

You can add new content elements to the Inheritance group by opening the Add content dialog, and drag it inside the Inheritance group.





Note that when adding new elements to Inheritance groups on a page, those elements will inherited by the same Inheritance group on any descendant page. By default, new elements will appear on the bottom of the Inheritance group on descendant pages.

7.3.3.3 Removing content elements

Clicking on the 'Remove' symbol in the Inheritance group editor will perform one of two actions, depending on where the content element was added to the Inheritance group.
If the content element was added to the Inheritance group of the currently edited page or a page in the current folder, the element will just be removed from the Inheritance group, and will not appear in the Inheritance group on any descendant pages. But if the content element was added to the Inheritance group in an ancestor folder, the element will be hidden, and will also be hidden in the Inheritance group on descendant pages. The difference is that a hidden element can be set to "visible" again in a descendant page.

7.3.3.4 Reordering content elements

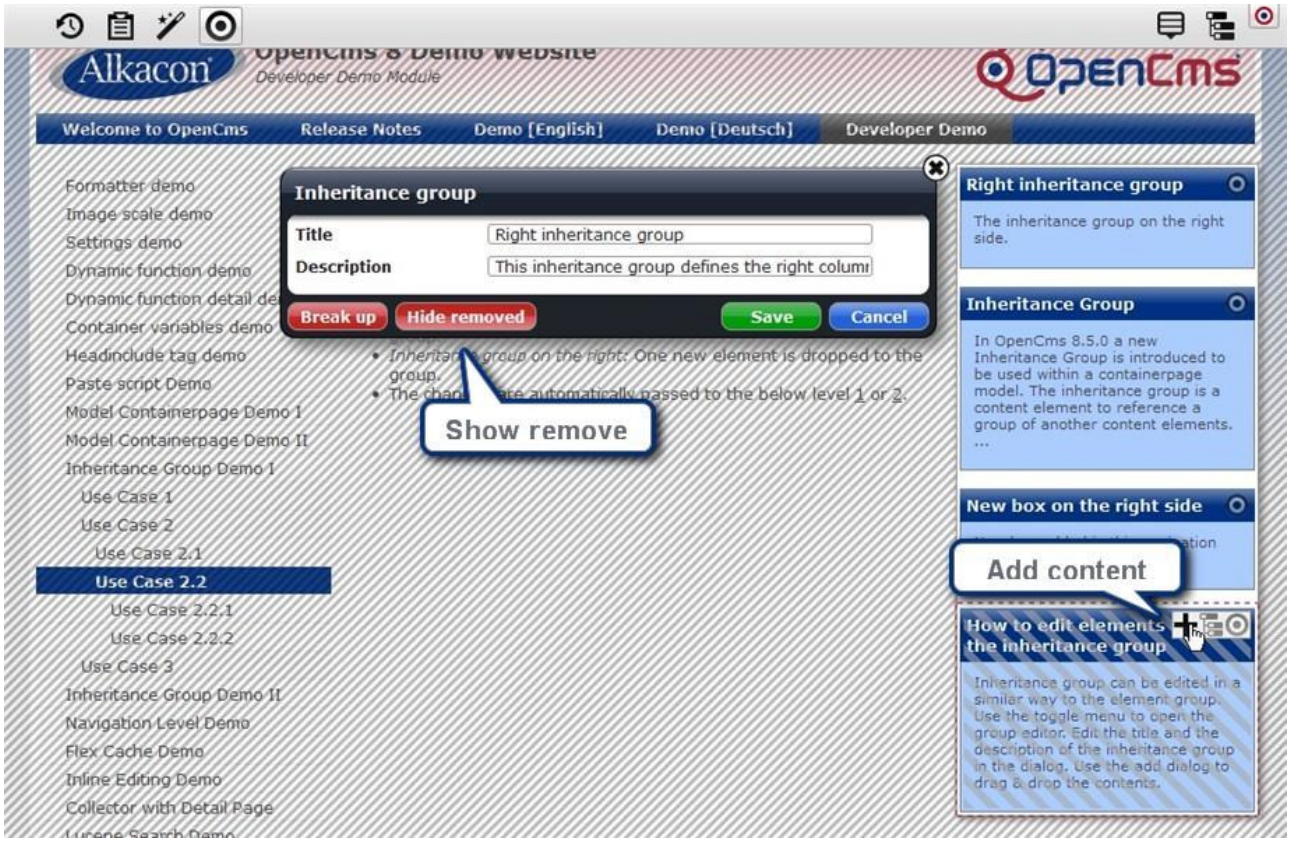
Inheritance groups can be reordered via drag-and-drop. Just pull the elements by their move icons to rearrange them inside the Inheritance group.

7.3.3.5 Inheritance status

If an element of an Inheritance group was not added for the current folder, a tree-like "Inheritance info" icon will be available from the element's toggle menu. Hovering your mouse over this icon will tell you from which folder this element was added to the Inheritance group.

7.3.3.6 Showing previously hidden elements

The Inheritance group editor's "Show removed" button will toggle the display of elements which have been hidden from the Inheritance group in an ancestor folder in the Inheritance group editor. To show a hidden element in the current page, first click on the "Show removed" button that displays the hidden elements. They are displayed with diagonal stripes on top to distinguish them from the other elements. Now hover the "**Edit point"** of the element you want to show and click the "Add content" (plus) button from the appearing toggle menu.



7.3.3.7 Save

All changes to an Inheritance group will only take effect if you click on the "Save" button. The Inheritance group editor will then be closed.

7.3.3.8 Break up

When using this button, the Inheritance group will be replaced by its current, individual content elements in the container page, just as with group containers. Note that this action takes effect immediately, not just after pressing the "Save" button.

7.3.3.9 Edit settings

Just like with other container page elements, the element settings of Inheritance group elements can be edited. But note that element settings can only be changed for an element if that element was added to the Inheritance group in the current folder. If the element was added to the Inheritance group in a different folder, the option for editing element settings will not appear.

7.4 Internals

There are two resource types used for the implementation of Inheritance groups: inheritance_group and inheritance_config. inheritance_group is the type of the contents which are actually inserted into the container page when creating new Inheritance groups or dragging existing Inheritance groups from the galleries into a page. They only contain the internal id of the Inheritance group they represent.

The resources of type inheritance_config contain the actual contents of the Inheritance groups. The Inheritance group changes for a folder will be stored in a file (of type inheritance_config) named .inherited which is contained in the same folder. Since only one file per folder is used, data for multiple Inheritance groups will be stored in the same file. The content of this config file should not be edited manually.

8 Collectors

Click here to open a simple collector demo

8.1 Implementation

In order to develop a collector, which can be used with `<cms:collector>` tag, the new collector should implement `I_CmsResourceCollector` interface. The package org.opencms.file.collectors already provides a standard implementation for this interface with `A_CmsResourceCollector`. Extend this class, if you develop your own collector. Following methods have to be implemented:

```
List<String> getCollectorNames();
```

`getCollectorNames()` return one or more collector names as list of strings.

```
String getCreateLink(CmsObject cms, String collectorName, String param)
    throws CmsException, CmsDataAccessException;
```

`getCreateLink(CmsObject, String, String)` returns the link that must be executed when a user clicks on the direct edit "new" button on a list created by the named collector. If this method returns `null`, it indicated that the selected collector implementation does not support a "create link", and so no "new" button will not be shown on lists generated with this collector.

```
String getCreateParam(CmsObject cms, String collectorName, String param)
    throws CmsDataAccessException;
```

The method `getCreateParam(CmsObject, String, String)` returns the parameter that must be passed to the `getCreateLink(CmsObject, String, String)`. If this method returns `null`, it indicates that the selected collector implementation does not support a "create link", and so no "new" button will be should shown on lists generated with this collector.

```
List<CmsResource> getResults(CmsObject cms, String collectorName, String param)
    throws CmsDataAccessException, CmsException;
```

`getResults(CmsObject, String, String)` returns a list of `org.opencms.file.CmsResource` Objects that are gathered in the VFS using the named collector.

8.2 Configuration

Edit opencms-vfs.xml and add following line to `<collectors>` node and restart the Servlet Container afterwards

```
<collector class="org.opencms.dev.demo.CmsSimpleResourceCollector" order="180" />
```

8.3 Using collectors in JSP files

Use the `<cms:contentload>` tag to collect the resources.

```
<cms:contentload collector="..." param="..." editable="true">
   <cms:contentaccess var="content" /> ...
</cms:contentlaod>
```

This tag requires following parameters:
collector – the name of the collector
param – collector parameter. The standard parameter syntax is: [path]|[resource type]|[count]

editable – Set this attribute to true, to enable the direct edit option for the list.
With `<cms:contentaccess var="content"/>` you get access to the `CmsXmlContent` of the current element inside the itteration.

```
<%@page buffer="none" session="false" taglibs="c,cms" %>
<%-- The JSP HTML should be surrounded by block element --%>
<div>
  <%-- Read collector paramter, e.g. from request --%>
  <c:set var="folder" value="${param.folder}"/>
  <c:set var="type" value="${param.type}"/>
  <c:set var="count" value="${param.count}"/>
    <ul>
    <%-- Use <cms:contentload> with new collector--%>
    <cms:contentload collector="myCollector" param="${folder}|${type}|${count}">
      <%-- Access the content --%>
      <cms:contentaccess var="content" />
      <c:set var="link"><cms:link>${content.filename}</cms:link></c:set>
      <li><a href="${link}">${content.value.Title}</a></li>
    </cms:contentload>
  </ul>
</div>
```

8.4 Making lists droppable

A common way to drop a JSP directly into a container page is to use the dynamic function.

- Create a new dynamic function. Usually this resource type is a part of the module and should be created in folder `/system/modules/mymodule/functions/`. In Dev Demo module the collector JSP is in folder:

`/system/modules/org.opencms.dev.demo/functions/`

- Select the JSP containing "myCollector". In Dev Demo it is the following jsp:

`/system/modules/org.opencms.dev.demo/pages/collector.jsp`

- To define the collector parameters set the initial request parameters in the dynamic function. These parameters are used in collector.jsp:
  - folder=/dev-demo/collector-with-detail-page/.content/article/
  - type=ddarticle
  - count=5

Now you can open a container page in ADE and drop the new dynamic function into the container page.

8.5 Using detail pages with collectors

- Go to the sitemap editor and open the "Create page" dialog.
- Drag a page from the “Types” tab for the resource type that is configured for the collector.
- OpenCms configures the new detail page automatically for the sub-sitemap.

Now when you click on the link to open the resource, the detail page of the resource is opened. In the list the detail page url is used.

8.6 Code example

Example of a simple collector class and JSP:
CmsSimpleResourceCollector.java
simple-collector.jsp

9 XSD choice element

In OpenCms the XSD choice element extends the XML Schema definition. The XSD choice provides a single or multiple choices of OpenCms types in arbitrary order. This documentation part describes how to use the choice element within XSD and how to access the values in a JSP with EL.

9.1 Definition

The `<xsd:choice>` node can be used in the same way as a `<xsd:sequnce>` to describe an OpenCms type. It has to be defined in XML schema definition of a **nested XML content**. The element in the root schema has to be **optional**. Add attribute `minOccurs="0"` on the element to make it optional. **Root XML Schema Dfinition:**

```
<%@page buffer="none" session="false" taglibs="c,cms" %>
<%-- The JSP HTML should be surround by block element --%>
<div>
      <%-- Read collector paramter, e.g. from request --%>
        <c:set var="folder" value="${param.folder}"/>
        <c:set var="type" value="${param.type}"/>
        <c:set var="count" value="${param.count}"/>
        <ul>
      <%-- Use <cms:contentload> with new collector--%>
        <cms:contentload collector="myCollector"
                         param="${folder}|${type}|${count}" editable="true">
              <%-- Access the content --%>
                <cms:contentaccess var="content" />
              <%-- Set the link to the content in the list and
                   do not forget to use <cms:link> tag --%>
                <li>
                  <c:set var="link"><cms:link>${content.filename}</cms:link></c:set>
                  <a href="${link}">
                      ${content.value.Title}
                  </a>
              </li>
        </cms:contentload>
        </ul>
</div>
```

Use the `<xsd:choice>` node to define the element of the selection. The elements, which define the selection options of the `<xsd:choice>` have to be **optional**, too.
Nested XML schema defenition with <xsd:choice>:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
        <xsd:include schemaLocation="opencms://opencms-xmlcontent.xsd"/>
        …
        <xsd:complexType name="OpenCmsDevDemoTextOption">
                <xsd:choice minOccurs="0" maxOccurs="3">
                  <xsd:element name="Text" type="OpenCmsString" minOccurs="0" />
                  <xsd:element name="Html" type="OpenCmsHtml" minOccurs="0" />
                  <xsd:element name="Link" type="OpenCmsVarLink" minOccurs="0" />
                </xsd:choice>
                <xsd:attribute name="language" type="OpenCmsLocale" use="optional"/>
        </xsd:complexType>
</xsd:schema>
```

9.2 Single and multiple choices

The `<xsd:choice>` element can be used as single or multiple choice.
To use the single choice do not use any attributes on the `<xsd:choice>` node. This is the default setting.
To use the multiple choice set following attributes:

- `minOccurs="0"` and
- `maxOccurs="[max number of elements]"`, e.g. `maxOccurs="5"`

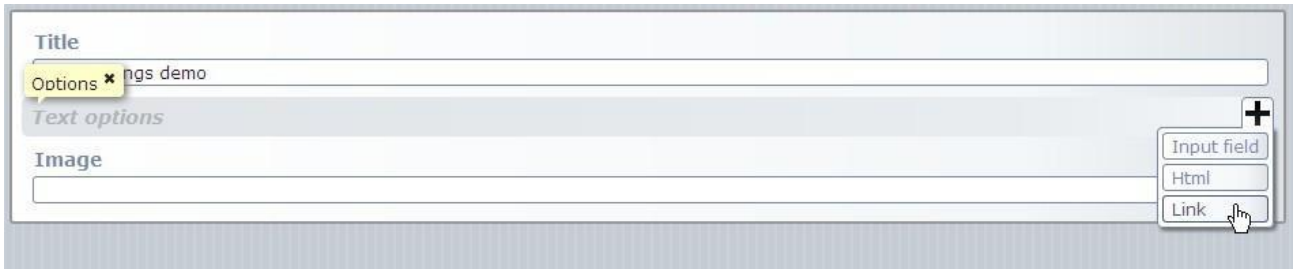
Single choice:

```
<xsd:choice>
   <xsd:element name="VariableLink" type="OpenCmsVarLink" minOccurs="0" />
   <xsd:element name="LinkGallery" type="OpenCmsVfsFile" minOccurs="0" />
   <xsd:element name="DownloadGallery" type="OpenCmsVfsFile" minOccurs="0" />
</xsd:choice>
```

Multiple choice:

```
<xsd:choice minOccurs="0" maxOccurs="5">
   <xsd:element name="VariableLink" type="OpenCmsVarLink" minOccurs="0" />
   <xsd:element name="LinkGallery" type="OpenCmsVfsFile" minOccurs="0" />
   <xsd:element name="DownloadGallery" type="OpenCmsVfsFile" minOccurs="0" />
</xsd:choice>
```

9.3 Content editor



9.4 Accessing values in JSP

The elements of a `<xsd:choice>` node can be accessed in the same way as elements of the nested content. Before the content value is read, test the root element and the choice elements for their existence. Check the existence of the nested root element:

```
<c:if test="${value.Options.exists}"> … </c:if>
```

Check the value of the choice element. Use `isSet` for the examination. `isSet` returns true, if the element exists and its value is not an empty string.
We exend the above example:

```
<c:if test="${value.Options.exists && value.Options.value.Text.isSet}">
   <div>${value.Options.value.Text}</div>
</c:if>
```

In the same way access the other choice elements:

```
${value.Options.value.Html.exists}
${value.Options.value.Html.isSet}
${value.Options.value.Html}
${value.Options.value.Link.exists}
${value.Options.value.Link.isSet}
${value.Options.value.Link}
```

Complete example with a choice element inside a nested content (v8InfoBox):

```
<c:if test="${value.FurtherInfo.value.Link.exists
            && (value.FurtherInfo.value.Link.value.VariableLink.isSet
            || value.FurtherInfo.value.Link.value.LinkGallery.isSet
            || value.FurtherInfo.value.Link.value.DownloadGallery.isSet)}">
   <c:choose>
       <c:when test="${value.FurtherInfo.value.Link.value.VariableLink.isSet}">
           <c:set var="infolink">
               ${value.FurtherInfo.value.Link.value.VariableLink}
           </c:set>
       </c:when>
       <c:when test="${value.FurtherInfo.value.Link.value.LinkGallery.isSet}">
           <c:set var="infolink">
               ${value.FurtherInfo.value.Link.value.LinkGallery}
           </c:set>
       </c:when>
       <c:when test="${value.FurtherInfo.value.Link.value.DownloadGallery.isSet}">
           <c:set var="infolink">
                ${value.FurtherInfo.value.Link.value.DownloadGallery}
           </c:set>
       </c:when>
   </c:choose>
   <c:set var="infotext">${infolink}</c:set> ...
   <div class="boxbody_listentry">
       <c:set var="link"><cms:link>${infolink}</cms:link></c:set>
       <a href="${link}">${infotext}</a><br/>
   </div>
</c:if>
```

`${value.FurtherInfo.value.Link.exists}` checks the existence of the choice element. `${value.FurtherInfo.value.Link.value.VariableLink.isSet}` checks the value of a choice element.

9.5 Examples

The full examples for the usage of the choice element can be find in the Development Demo as well as in Template III.

1. The root XSD of the info box (v8-modules)
2. The nested XSD containing further info (v8-modules)
3. The nested XSD containing choice element (v8-modules)
4. The formatter JSP (v8-modules)
5. The root XSD of the article with settings (dev-demo)
6. The nested XSD containing choice element (dev-demo)

10 ADE configuration

10.1 Sitemap configuration

The central place where the resource types available through the ADE's “Add Content” dialog and other global settings are configured is the “.content/.config file. Settings for sub-sitemaps follow the same mechanism with “.content/.config” file based in the sub-sitemap.

The Sitemap configuration file offers 4 tabs:

- **Resource types**: Defines a resource type‘s name and name pattern, default folder, formatter (optional) and more.
- **Model pages**: Defines models to be used by the sitemap editor for generating new container pages.
- **Property configuration**: Defines which properties get displayed under "Basic Properties" tab for the property dialog in ADE.
- **Detail pages**: Defines detail pages used for specified resource type in the sitemap

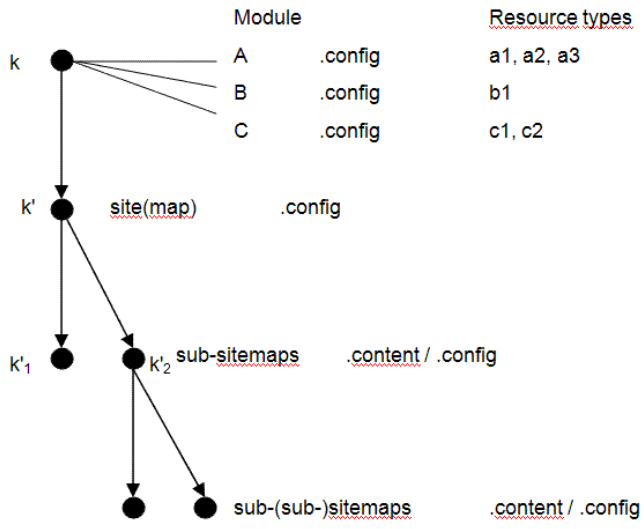
10.2 Module configuration

In case a module adds a new resource type (as defined in the module's manifest) to OpenCms and you want to make this automatically available to ADE after the module is installed, you can place “.config” file of the type “module_config” inside the root folder of your module.

The Module configuration file offers three options:

- **Resource types**: Defines a resource type‘s name and name pattern, default folder, formatter and more.
- **Property configuration**: Defines which properties get displayed under "Basic Properties" tab for the property dialog in ADE.
- **Functions**: Defines a dynamic function that is shipped with the module (A name for identification and a JSP provider is required at least)

10.3 Configuration inheritance



11 Solr search integration

11.1 Abstract

After searching with Apache's Lucene for years Apache Solr has grown and grown and can now be called an enterprise search platform that is based on Lucene. It is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via XML, JSON or binary over HTTP. You query it via HTTP GET and receive XML, JSON, or binary results. To get a more detailed knowledge what Solr exactly is and how it works, please visit the Apache Solr project website. Searching with the powerful and flexible Apache Solr's REST-like interface will drill down the development complexity. More over you can rely on existing graphical interfaces that provide comfortable AJAX based search functionality to the end user of your internet/intranet application.

11.2 Searching for content in OpenCms

OpenCms 8.5 integrates Apache Solr. And not only for full text search, but as a powerful enterprise search platform as well.

11.2.1 DEMO

The OpenCms v8 modules are covering a Solr based OpenCms Search Demonstration using the Open Source UI Ajax Solr (Ajax Solr Project on github.com):
Click here to run the OpenCms Solr Search DEMO

11.2.2 Quick start example

11.2.2.1 Send a REST-like query

Imagine you want to show a list of "all articles, that have changed since yesterday, where property 'X' has the value 'Y' :

```
http://localhost:8080/opencms/opencms/handleSolrSelect?
    fq=type:v8article
    &fq=lastmodified:[NOW-1DAY TO NOW]
    &fq=Title_prop:Flower
```

Parameter explanation:

```
http://localhost:8080/opencms/opencms/handleSolrSelect
   // The URI of the OpenCms Solr Select Handler configured in 'opencms-system.xml'
    ?fq=type:v8article                  // Filter query on the field type
                                        // with the value 'v8article'
    &fq=lastmodified:[NOW-1DAY TO NOW]  // Filter query on the field lastmodified
                                        // with a range query from 'NOW-1DAY TO NOW'
    &fq=Title_prop:Flower               // Filter query on the field Title_prop
                                        // with the value 'v8article'
```

NOTE: Solr query Syntax
If you want to get familiar with the Solr query syntax you will get a general overview at Solr query syntax. For advanced features Searching - Solr Reference Guide - Lucid Imagination will lend a hand.

Please note that many characters in the Solr Query Syntax (most notable the plus sign: "+") are special characters in URLs, so when constructing request URLs manually, you must properly URL-Encode these characters.

```
                                q=  +popularity:[10   TO   *]     +section:0
http://localhost:8983/solr/select?q=%2Bpopularity:[10%20TO%20*]%20%2Bsection:0
```

For more information, see Yonik Seeley's blog on Nested Queries in Solr.

You can pass any "Solr valid" input to the new OpenCms Solr request handler (handleSolrSelect). To get familiar with the Solr query syntax the Solr Wiki page lends itself: Search and Indexing

11.2.2.2 Retrieve the response

The response produced by Solr can be XML or JSON by default. With an additional parameter 'wt' you can specify the QueryResponseWriter that should be used by Solr. For the above shown query example a result can look like this:

```
<response>
 <lst name="responseHeader">
   <int name="status">0</int>
   <int name="QTime">7</int>
   <lst name="params">
     <str name="qt">dismax</str>
     <str name="fl">*,score</str>
     <int name="rows">50</int>
     <str name="q">*:*</str>
     <arr name="fq">
       <str>type:v8article</str>
       <str>contentdate:[NOW-1DAY TO NOW]</str>
       <str>Title_prop:Flower</str>
     </arr>
     <long name="start">0</long>
   </lst>
 </lst>
 <result name="response" numFound="2" start="0">
   <doc>
     <str name="id">51041618-77f5-11e0-be13-000c2972a6a4</str>
     <str name="contentblob">[B:[B@6c1cb5</str>
     <str name="path">/sites/default/.content/article/a_00003.html</str>
     <str name="type">v8article</str>
     <str name="suffix">.html</str>
     <date name="created">2011-05-06T15:27:13Z</date>
     <date name="lastmodified">2011-08-17T13:58:29Z</date>
     <date name="contentdate">2012-09-03T10:41:13.56Z</date>
     <date name="relased">1970-01-01T00:00:00Z</date>
     <date name="expired">292278994-08-17T07:12:55.807Z</date>
     <arr name="res_locales">
       <str>en</str>
       <str>de</str>
     </arr>
     <arr name="con_locales">
       <str>en</str>
    </arr>
     <str name="template_prop">
       /system/modules/com.alkacon.opencms.v8.template3/templates/main.jsp</str>
     <str name="style.layout_prop">/.content/style</str>
     <str name="NavText_prop">OpenCms 8 Demo</str>
     <str name="Title_prop">Flower Today</str>
     <arr name="content_en">
       <str>News from the world of flowers Flower Today In this [...]</str>
     </arr>
```

```
    <date name="timestamp">2012-09-03T10:45:47.055Z</date>
    <float name="score">1.0</float>
  </doc>
  <doc>
    <str name="id">ac56418f-77fd-11e0-be13-000c2972a6a4</str>
    <str name="contentblob">[B:[B@1d0e4a2</str>
    <str name="path">/sites/default/.content/article/a_00030.html</str>
    <str name="type">v8article</str>
    <str name="suffix">.html</str>
    <date name="created">2011-05-06T16:27:02Z</date>
    <date name="lastmodified">2011-08-17T14:03:27Z</date>
    <date name="contentdate">2012-09-03T10:41:18.155Z</date>
    <date name="relased">1970-01-01T00:00:00Z</date>
    <date name="expired">292278994-08-17T07:12:55.807Z</date>
    <arr name="res_locales">
      <str>en</str>
      <str>de</str>
    </arr>
    <arr name="con_locales">
      <str>en</str>
    </arr>
    <str name="template_prop">
      /system/modules/com.alkacon.opencms.v8.template3/templates/main.jsp
    </str>
    <str name="style.layout_prop">/.content/style</str>
    <str name="NavText_prop">OpenCms 8 Demo</str>
    <str name="Title_prop">Flower Dictionary</str>
    <arr name="content_en">
      <str>The different types of flowers Flower Dictionary There are [...]</str>
    </arr>
    <date name="timestamp">2012-09-03T10:45:49.265Z</date>
    <float name="score">1.0</float>
  </doc>
 </result>
</response>
```

11.2.2.3 Send a Java-API query

```
String query="fq=type:v8article&fq=lastmodified:[NOW-1DAY TO
NOW]&fq=Title_prop:Flower";
 CmsSolrResultList results = OpenCms.getSearchManager().getIndexSolr("Solr Online
Index").search(getCmsObject(), query);
 for (CmsSearchResource sResource : results) {
   String path = searchRes.getField(I_CmsSearchField.FIELD_PATH);
   Date date =searchRes.getMultivaluedField(I_CmsSearchField.FIELD_DATE_LASTMODIFIED);
   List<String> cats = searchRes.getMultivaluedField(I_CmsSearchField.FIELD_CATEGORY);
 }
```

The class `org.opencms.search.solr.CmsSolrResultList` encapsulates a list of 'OpenCms resource documents' (`{@link CmsSearchResource}`).
This list can be accessed exactly like an {@link ArrayList} which entries are {@link CmsSearchResource} that extend {@link CmsResource} and holds the Solr implementation of {@link I_CmsSearchDocument} as member. **This enables you to deal with the resulting list as you do with well known {@link List} and work on it's entries like you do on {@link CmsResource}.**

11.2.2.4 Use CmsSolrQuery class for querying Solr

```
CmsSolrIndex index = OpenCms.getSearchManager().getIndexSolr("Solr Online Index");
CmsSolrQuery squery = new CmsSolrQuery(getCmsObject(),
    "path:/sites/default/xmlcontent/article_0001.html");
List<CmsResource> results = index.search(getCmsObject(), squery);
```

11.2.3 Advanced search features

Auto suggestion/completion/correction: http://wiki.apache.org/solr/Suggester
Facteted search: http://wiki.apache.org/solr/SimpleFacetParameters
Highlighting: http://wiki.apache.org/solr/HighlightingParameters
Range queries: http://wiki.apache.org/solr/SolrQuerySyntax
Sorting: http://wiki.apache.org/solr/CommonQueryParameters
Spellchecking: http://wiki.apache.org/solr/SpellCheckComponent
Thesaurus/Synomys: http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters

11.2.3.1 Querying multiple cores (indexes)

'Core' is the wording in the Solr world for thinking of several indexes. Preferring the correct speech, let's say core instead index. Multiple cores should only be required if you have completely different applications but want a single Solr Server that manages all the data. See Solr Core Administration for detailed information. So assuming you have configured multiple Solr cores and you would like to query a specific one you have to tell Solr/OpenCms which core/index you want to search on. This is done by a special parameter:

```
http://localhost:8080/opencms/opencms/handleSolrSelect?
                             // The URI of the OpenCms Solr Select Handler
                             // configured in 'opencms-system.xml'
   &core=My Solr Index Name  // Searches on the core with the name 'My Solr Index Name'
   &q=content_en:Flower      // for the text 'Flower'
```

11.2.4 Using the standard OpenCms Solr collector

OpenCms version 8.5 delivers a standard Solr collector using byQuery as name to simply pass a query string and byContext as name to pass a query string and led OpenCms use the user's request context. The implementing class for this collector can be found at `org.opencms.file.collectors.CmsSolrCollector`.

```
<cms:contentload collector="byQuery" preload="true"
  param="fq=parent-folders:/sites/default/&fq=type:ddarticle&sort=lastmodified desc">
  <cms:contentinfo var="info" />
  <c:if test='${info.resultSize != 0}'>
    <cms:contentinfo var="info" />
    <c:if test='${info.resultSize != 0}'>
      <h3>Solr Collector Demo</h3>
      <cms:contentload editable="false">
        <cms:contentaccess var="content" />
        <%-- Title of the article --%>
        <h6>${content.value.Title}</h6>
        <%-- The text field of the article with image --%>
        <div class="paragraph">
          <%-- Set the requied variables for the image. --%>
          <c:if test="${content.value.Image.isSet}">
            <%-- Output of the image using cms:img tag --%>
            <c:set var="imgwidth">${(cms.container.width - 20) / 3}</c:set>
            <%-- Output the image. --%>
            <cms:img src="${content.value.Image}" />
          </c:if>
          ${cms:trimToSize(cms:stripHtml(content.value.Text), 300)}
        </div>
        <div class="clear"></div>
      </cms:contentload>
    </c:if>
  </c:if>
</cms:contentload>
```

11.3 Indexing content of OpenCms

11.3.1 Search configuration

In general the system wide search configuration for OpenCms is done in the file 'opencms-search.xml' (<CATALINA_HOME>/webapps/<OPENCMS_WEBAPP>/WEB_INF/config/opencms-search.xml).

11.3.1.1 Embedded/HTTP Solr Server

Since version 8.5 of OpenCms a new optional node with the **XPath**: opencms/search/solr is available. To simply enable the OpenCms embedded Solr Server your opencms-search.xml should start like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE opencms SYSTEM "http://www.opencms.org/dtd/6.0/opencms-search.dtd">
<opencms>
 <search>
   <solr enabled="true"/>
     [...]
 </search>
</opencms>
```

Optionally you can configure the Solr home directory and the main Solr configuration file name solr.xml. OpenCms then concats those two paths to <solr_home><configfile> an example for such a configuration would look like:

```
<solr enabled="true">
   <home>/my/solr/home/folder</home>
   <configfile>rabbit.xml</configfile>
</solr>
```

In order to disable Solr system wide remove the <solr/>-node or set the enabled attribute to 'false' like:

```
<solr enabled="false"/>
```

It is also possible to connect with an external HTTP Solr server, to do so replace the line <solr enabled="true"/> with the following:

```
<solr enabled="true" serverUrl="http://mySolrServer" />
```

The OpenCms SolrSelect request handler does not support the external HTTP Solr Server. So if your HTTP Solr Server is directly reachable by http://<your_server> there will no permission check performed and indexed data that is secret will be accessible. What means that you are **self-responsible** for resources that have permission restrictions set on the VFS of OpenCms. But of course you can use the method org.opencms.search.solr.CmsSolrIndex.search(CmsObject, SolrQuery) or org.opencms.search.solr.CmsSolrIndex.search(CmsObject, String) and be sure permissions are checked also for HTTP Solr Servers. Maybe a future version of OpenCms will feature a secure access on HTTP Solr server.

11.3.1.2 Search index(es)

By default OpenCms comes along with a "Solr Online" index. To add a new Solr index you can use the default configuration as copy template.

```
<index class="org.opencms.search.solr.CmsSolrIndex">
 <name>Solr Online</name>
 <rebuild>auto</rebuild>
 <project>Online</project>
 <locale>all</locale>
 <configuration>solr_fields</configuration>
 <sources>
   < source >solr_source< /source >
 </sources>
</index>
```

11.3.1.3 Index sources

Index sources for Solr can be configured in the file `opencms-search.cml` exactly the same way as you do for Lucene indexes. In order to use the advanced XSD field mapping for XML contents, you must add the new document type `xmlcontent-solr` to the list of document types that are indexed:

```
<indexsource>
 <name>solr_source</name>
 <indexer class="org.opencms.search.CmsVfsIndexer"/>
 <resources>
   <resource>/sites/default/</resource>
 </resources>
 <documenttypes-indexed>
   <name>xmlcontent-solr</name>
   <name>containerpage</name>
   <name>xmlpage</name>
   <name>text</name>
   <name>pdf</name>
   <name>image</name>
   <name>msoffice-ole2</name>
   <name>msoffice-ooxml</name>
   <name>openoffice</name>
 </documenttypes-indexed>
</indexsource>
```

11.3.1.4 A new document type

With OpenCms version 8.5 there is a new document type called `xmlcontent-solr`. Its implementation `(CmsSolrDocumentXmlContent)` performs a localized content extraction that is used later on to fill the Solr input document. As explained in section Custom fields for XML contents it is possible to define a mapping between elements defined in the XSD of an XML resource type and a field of the Solr document. The values for those defined XSD field mappings are also extracted by the document type named `xmlcontent-solr`.

```
<documenttype>
 <name>xmlcontent-solr</name>
 <class>org.opencms.search.solr.CmsSolrDocumentXmlContent</class>
 <mimetypes>
   <mimetype>text/html</mimetype>
 </mimetypes>
 <resourcetypes>
   <resourcetype>xmlcontent-solr</resourcetype>
 </resourcetypes>
</documenttype>
```

11.3.1.5 The Solr default field configuration

By default the field configuration for OpenCms Solr indexes is implemented by the class `org.opencms.search.solr.CmsSolrFieldConfiguration`. The most easiest Solr field configuration declared in `opencms-search.xml` looks like the following. See also Extend the CmsSolrFieldConfiguration

```
<fieldconfiguration class="org.opencms.search.solr.CmsSolrFieldConfiguration">
 <name>solr_fields</name>
 <description>The Solr search index field configuration.</description>
 <fields/>
</fieldconfiguration>
```

11.3.1.6 Migrating a Lucene index to a Solr index

An existing Lucene field configuration can easily be transformed into a Solr index. To do so create a new Solr field configuration. Therefore you can use the snippet shown in section The Solr default field configuration as template and copy the list of fields from the Lucene index you want to convert into that skeleton.
There exists a specific strategy to map the Lucene field names to Solr field names:

- **Exact name matching:** OpenCms tries to determine an explicit Solr field that has the exact name like the value of the name-attribute. E.g. OpenCms tries to find an explicit Solr filed definition named meta for `<field name="meta"> ... </field>`. To make use of this strategy you have to edit the `schema.xml` of Solr manually and add an explicit field definition named according to the exact Lucene field names.
- **Type specific fields:** In the existing Lucene configuration type specific field definitions are not designated, but the Solr schema.xml defines different data types for fields. If you are interested in making use of those type specific advantages (like language specific field analyzing/tokenizing) without manipulating the schema.xml of Solr, you have to define a type attribute for those fields at least. The value of the attribute type can be any name of each <dynamicField> configured in the schema.xml that starts with a *_. The resulting field inside the Solr document is then named <luceneFieldName>_<dynamicFieldSuffix>.
- **Fallback:** If you don't have defined a type attribute and there does not exist an explicit field in the schema.xml named according to the Lucene field name OpenCms uses text_general as fallback. E.g. a Lucene field `<field name="title" index="true"> ... </field>` will be stored as a dynamic field named `title_txt` in the Solr index.

An originally field configuration like:

```
<fieldconfiguration>
 <name>standard</name>
 <description>The standard OpenCms 8.0 search field configuration.</description>
 <fields>
   <field name="content" store="compress" index="true" excerpt="true">
     <mapping type="content"/>
   </field>
   <field name="title-key" display="-" store="true" index="untokenized" boost="0.0">
     <mapping type="property">Title</mapping>
   </field>
   <field name="title" display="%(key.field.title)" store="false" index="true">
     <mapping type="property">Title</mapping>
   </field>
   <field name="keywords" display="%(key.field.keywords)" store="true" index="true">
```

```
      <mapping type="property">Keywords</mapping>
    </field>
    <field name="description" store="true" index="true">
      <mapping type="property">Description</mapping>
    </field>
    <field name="meta" display="%(key.field.meta)" store="false" index="true">
      <mapping type="property">Title</mapping>
      <mapping type="property">Keywords</mapping>
      <mapping type="property">Description</mapping>
    </field>
  </fields>
</fieldconfiguration>
```

Could look after the conversion like this:

```
<fieldconfiguration class="org.opencms.search.solr.CmsSolrFieldConfiguration">
 <name>standard</name>
 <description>The standard OpenCms 8.0 Solr search field configuration.</description>
 <fields>
   <field name="content" store="compress" index="true" excerpt="true">
     <mapping type="content"/>
   </field>
   <field name="title-key" store="true" index="untokenized" boost="0.0" type="s">
     <mapping type="property">Title</mapping>
   </field>
   <field name="title" store="false" index="true" type="prop">
     <mapping type="property">Title</mapping>
   </field>
   <field name="keywords" store="true" index="true" type="prop">
     <mapping type="property">Keywords</mapping>
   </field>
   <field name="description" store="true" index="true" type="prop">
     <mapping type="property">Description</mapping>
   </field>
   <field name="meta" store="false" index="true" type="en">
     <mapping type="property">Title</mapping>
     <mapping type="property">Keywords</mapping>
     <mapping type="property">Description</mapping>
   </field>
 </fields>
</fieldconfiguration>
```

11.3.2 Indexed data

The following sections will show what data is indexed by default and what possibilities are offered by OpenCms to configure / implement additional field configurations / mappings.

11.3.2.1 The Solr index schema (schema.xml)

Have a look at the Solr `schema.xml` first. In the file `<CATALINA_HOME>/webapps/<OPENCMS>/WEB-INF/solr/conf/schema.xml` you will find the field definitions that will be used by OpenCms that were briefly summarized before.

11.3.2.2 Default index fields

OpenCms indexes several information for each resource by default:

- **id** - Structure id used as unique identifier for an document (The structure id of the resource)
- **path** - Full root path (The root path of the resource e.g. /sites/default/flower_en/.content/article.html)
- **path_hierarchy** - The full path as (path tokenized field type: text_path)

- **parent-folders** - Parent folders (multi-valued field containing an entry for each parent path)
- **type** - Type name (the resource type name)
- **res_locales** - Existing locale nodes for XML content and all available locales in case of binary files
- **created** - The creation date (The date when the resource itself has being created)
- **lastmodified** - The date last modified (The last modification date of the resource itself)
- **contentdate** - The content date (The date when the resource's content has been modified)
- **released** - The release and expiration date of the resource
- **content** - A general content field that holds all extracted resource data (all languages, type text_general)
- **contentblob** - The serialized extraction result (content_blob) to improve the extraction performance while indexing
- **category** - All categories as general text
- **category_exact** - All categories as exact string for faceting reasons
- **text_<locale>** - Extracted textual content optimized for the language specific search (Default languages: en, de, el, es, fr, hu, it)
- **timestamp** - The time when the document was indexed last time
- ***_prop** - All properties of a resource as searchable and stored text (field name: <Property_Definition_Name>_prop as text_general)
- ***_exact** - All properties of a resource as exact not stored string (field name: <Property_Definition_Name>_exact as string)

11.3.2.3 Custom fields for XML contents

Declarative field configuration with field mappings can also be bone via the **`XSD-Content-Definition`** of an XML resource type as defined in the **`DefaultAppinfoTypes.xsd`**

```
<xsd:complexType name="OpenCmsDefaultAppinfoSearchsetting">
   <xsd:sequence>
     <xsd:element name="solrfield"
                  type="OpenCmsDefaultAppinfoSolrField"
                  minOccurs="0" maxOccurs="unbounded" />
   </xsd:sequence>
   <xsd:attribute name="element" type="xsd:string" use="required" />
   <xsd:attribute name="searchcontent"
                  type="xsd:boolean" use="optional" default="true" />
 </xsd:complexType>

 <xsd:complexType name="OpenCmsDefaultAppinfoSolrField">
   <xsd:sequence>
     <xsd:element name="mapping"
                  type="OpenCmsDefaultAppinfoSolrFieldMapping"
                  minOccurs="0" maxOccurs="unbounded" />
   </xsd:sequence>
   <xsd:attribute name="targetfield" type="xsd:string" use="required" />
   <xsd:attribute name="sourcefield" type="xsd:string" use="optional" />
   <xsd:attribute name="copyfields" type="xsd:string" use="optional" />
   <xsd:attribute name="locale" type="xsd:string" use="optional" />
   <xsd:attribute name="default" type="xsd:string" use="optional" />
   <xsd:attribute name="boost" type="xsd:string" use="optional" />
 </xsd:complexType>
```

You are able to declare search field mappings for XML content elements directly in the XSD Content Definition. A XSD using this feature can then look like:

```
<searchsettings>
  <searchsetting element="Title" searchcontent="true">
    <solrfield targetfield="atitle">
      <mapping type="property">Author</mapping>
    </solrfield>
  </searchsetting>
  <searchsetting element="Teaser">
    <solrfield targetfield="ateaser">
      <mapping type="item" default="Homepage n.a.">Homepage</mapping>
      <mapping type="content"/>
      <mapping type="property-search">search.special</mapping>
      <mapping type="attribute">dateReleased</mapping>
      <mapping type="dynamic"
               class="org.opencms.search.solr.CmsDynamicDummyField">special
      </mapping>
    </solrfield>
  </searchsetting>
  <searchsetting element="Text" searchcontent="true">
    <solrfield targetfield="ahtml" boost="2.0"/>
  </searchsetting>
  <searchsetting element="Release" searchcontent="false">
    <solrfield targetfield="arelease" sourcefield="*_dt" />
  </searchsetting>
  <searchsetting element="Author" searchcontent="true">
    <solrfield targetfield="aauthor" locale="de"
               copyfields="test_text_de,test_text_en" />
  </searchsetting>
  <searchsetting element="Homepage" searchcontent="true">
    <solrfield targetfield="ahomepage" default="Homepage n.a." />
  </searchsetting>
</searchsettings>
```

11.3.2.4 Dynamic field mappings

If the requirements for the field mapping are more "dynamic" than just: **`'static piece of content' -> 'specified field defined in the Solr schema'`**, you are able to implement the the interface **`org.opencms.search.fields.I_CmsSearchFieldMapping`**.

11.3.2.5 Custom field configuration

Declarative field configurations with field mappings can be defined in the file **`opencms-search.xml`**. You can use exactly the same features as already known for OpenCms Lucene field configurations.

- **Please see** Migrating a Lucene index to a Solr index

11.3.2.6 Extend the CmsSolrFieldConfiguration

If the standard configuration options are still not flexible enough you are able to extends from the class: **`org.opencms.search.solr.CmsSolrFieldConfiguration`** and define a custom Solr field configuration in the **`opencms-search.xml`**:

```
<fieldconfiguration class="your.package.YourSolrFieldConfiguration">
  <name>solr_fields</name>
  <description>The Solr search index field configuration.</description>
  <fields/>
</fieldconfiguration>
```

11.4 Behind the walls

11.4.1 The request handler

The class **`org.opencms.main.OpenCmsSolrHandler`** offers the same functionality as the default select request handler of an standard Solr server installation. In the OpenCms default system configuration (opencms-system.xml) the Solr request handler is configured:

```
<requesthandlers>
      <requesthandler class="org.opencms.main.OpenCmsSolrHandler" />
</requesthandlers>
```

Alternativly the request handler class can be used as Servlet, therefore add the handler class to the **`WEB-INF/web.xml`** of your OpenCms application:

```
   <servlet>
       <description>
           The OpenCms Solr servlet.
       </description>
       <servlet-name>OpenCmsSolrServlet</servlet-name>
       <servlet-class>org.opencms.main.OpenCmsSolrHandler</servlet-class>
       <load-on-startup>1</load-on-startup>
   </servlet>
     [...]
   <servlet-mapping>
       <servlet-name>OpenCmsSolrServlet</servlet-name>
       <url-pattern>/solr/*</url-pattern>
   </servlet-mapping>
```

11.4.2 Permission check

OpenCms performs a permission check for all resulting documents and throws those away that the current user is not allowed to retrieve and expands the result for the next best matching documents on the fly. This security check is very cost intensive and should be replaced/improved with a pure index based permission check.

11.4.3 Configurable post processor

OpenCms offers the capability for post search processing Solr documents after the document has been checked for permissions. This capability allows you to add fields to the found document before the search result is returned. In order to make use of the post processor you have to add an optional parameter for the search index as follows:

```
<index class="org.opencms.search.solr.CmsSolrIndex">
   <name>Solr Offline</name>
   <rebuild>offline</rebuild>
   <project>Offline</project>
   <locale>all</locale>
   <configuration>solr_fields</configuration>
   <sources>
     [...]
   </sources>
   <param name="org.opencms.search.solr.CmsSolrIndex.postProcessor">
      my.package.MyPostProcessor
  </param>
</index>
```

The specified class for the parameter **`org.opencms.search.solr.CmsSolrIndex.postProcessor`** must be an implementation of **`org.opencms.search.solr.I_CmsSolrPostSearchProcessor`**.

11.4.4 Multilingual support

There is a default strategy implemented for the multi-language support within OpenCms Solr search index. For binary documents the language is determined automatically based on the extracted text. The default mechanism is implemented with: http://code.google.com/p/language-detection/
For XML contents we have the concrete language/locale information and the localized fields are ending with underscore followed by the locale. E.g.: content_en, content_de or text_en, text_de. By default all the field mappings definied within the XSD of a resource type are extended by the **`'_<locale>'`**.

11.4.5 Multilingual dependency resolving

Based on the file name of a resource in OpenCms there exists a concept to index documents that are distributed over more than one resource in OpenCms. The standard implementation can be found at: **`org.opencms.search.documents.CmsDocumentDependency`**

11.4.6 Extraction result cache

For better index performance the extracted result is cached for siblings @see **`org.opencms.search.extractors.I_CmsExtractionResult`**

11.5 Frequently asked questions

11.5.1 How is Solr integrated in general?

Independent from OpenCms a standard Solr Server offers a HTTP-Interface that is reachable at: http://localhost:8983/solr/select in a default Apache Solr Installation:
You are able to attach each valid Solr query documented at http://wiki.apache.org/solr/SolrQuerySyntax to this URL.
The HTTP response can either be JSON or XML and the answer of the query http://localhost:8983/solr/select?q=*:*&rows=2 could look like:

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">32</int>
    <lst name="params">
    <str name="q">*:*</str>
    <int name="rows">2</int>
    <long name="start">0</long>
  </lst>
  <result name="response" numFound="139" start="0">
    <doc>...</doc>
    <doc>...</doc>
  </result>
</response>
```

Solr is implemented in Java and there exists a Apache library called solrj, that enables to access a running Solr server by writing Java code:
The Solr integration in OpenCms offers both Interfaces: Java and HTTP. The default URL for sending Solr queries against OpenCms is:
http://localhost:8080/opencms/opencms/handleSolrSelect
this handler can answer any syntactically correct Solr query (http://wiki.apache.org/solr/SolrQuerySyntax).
The following code shows a simple example how to program against the Java API:

```
//////////////////
// SEARCH START //
//////////////////
CmsObject cmsO = new CmsJspActionElement(pageContext, request,
response).getCmsObject();
String query = ((request.getParameter("query") != null && request.getParameter("query")
!= "") ? "q=" + request.getParameter("query") : "") + "&fq=type:ddarticle&sort=path
asc&rows=5";
CmsSolrResultList hits = OpenCms.getSearchManager().getIndexSolr("Solr
Offline").search(cmsO, query);
if (hits.size() > 0) { %>
  <h4>New way:
    <fmt:message key="v8.solr.results" />
    <%= hits.getNumFound() %> found / rows <%= hits.getRows() %></h4>
      <div class="boxbody"><%
        //////////////////
        // RESULTS LOOP //
        //////////////////
        for (CmsSearchResource resource : hits) { %>
          <div class="boxbody_listentry">
            <div class="twocols">
            <div>Path: <strong><%= resource.getRootPath() %></strong></div>
            <div>German: <strong>
              <%= resource.getDocument().getFieldValueAsString("Title_de") %></strong>
            </div>
            <div>English: <strong>
              <%= resource.getDocument().getFieldValueAsString("Title_en") %></strong>
            </div>
          </div>
        </div> <%
      }
    %>
  </div><%
}
```

11.5.2 How to sort text for specific languages?

In this example, text is sorted according to the default German rules provided by Java. The rules for sorting German in Java are defined in a package called a Java Locale.
Locales are typically defined as a combination of language and country, but you can specify just the language if you want. For example, if you specify "de" as the language, you will get sorting that works well for German language. If you specify "de" as the language and "CH" as the country, you will get German sorting specifically tailored for Switzerland. You can see a list of supported Locales here. And in drder to get more general information about how text analysis is working with Solr have a look at Language Analysis page.

```
<!-- define a field type for German collation -->
<fieldType name="collatedGERMAN" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.KeywordTokenizerFactory"/>
    <filter class="solr.CollationKeyFilterFactory"
        language="de"
        strength="primary"
    />
  </analyzer>
</fieldType>
...
<!-- define a field to store the German collated manufacturer names -->
<field name="manuGERMAN" type="collatedGERMAN" indexed="true" stored="false" />
...
<!-- copy the text to this field. We could create French, English, Spanish versions
     too, and sort differently for different users! -->
<copyField source="manu" dest="manuGERMAN"/>
```

11.5.3 How to highlight the search query in results?

11.5.3.1 Does OpenCms support result highlighting?

Yes, use the OpenCms Solr Select handler at:
localhost:8080/opencms/opencms/handleSolrSelect
and you will find the highlighting section below the list of documents within the returned XML/JSON:

```
<lst name="highlighting">
  <lst name="a710bb16-1e04-11e2-b767-6805ca037347">
    <arr name="content_en">
      <str><em>YIPI</em> <em>YOHO</em> text text text</str>
    </arr>
  </lst>
  [...]
</lst>
```

11.5.3.2 Does the Java API of OpenCms support highlighting?

Currently the OpenCms search API does not support full featured Solr highlighting. But you can make use of the Solr default highlighting mechanism or course @see [1] or [2] and:
1. Call org.opencms.search.solr.CmsSolrResultList#getSolrQueryResponse() that returns a SolrQueryResponse that is documented at: http://lucene.apache.org/solr/api-3_6_1/org/apache/solr/response/SolrQueryResponse.html
2. Or you can use the above mentioned OpenCms Solr Select handler at:
localhost:8080/opencms/opencms/handleSolrSelect

11.5.3.3 Is highlighting a performance killer?

Yes, for this reason highlighting is turned off before the first search is executed. After all not permitted resources are filtered out of the result list, the highlighting is performed again.

11.5.3.4 Solr indexing questions

11.5.3.5 Please explain the differences between the "Solr Online and Offline"?

As the name of the indexes let assume Offline indexes are also containing changes that have not yet been published and Online indexes only contain thoses resources that have already been published. The "Online EN VFS" is a Lucene based index and also contains only those resources that have been published.

11.5.3.6 When executing a Solr query, does only the solr index get used?

No, permissions are checked by OpenCms API afterwards.

11.5.3.7 Where to find general information about Solr?

If you are interested in Solr in general the Solr wiki is a good starting point: http://wiki.apache.org/solr/ The Documentation from CMS side you will find within the distributed PDF file.

11.5.3.8 Is there a way to create a full backup of the complete index?

You can copy the index folder 'WEB-INF/index/${INDEX_NAME}'' by hand.

11.5.3.9 How to rebuild indexes with a fail-safe?

Edit the opencms-search.xml within your WEB-INF/config directory and add the following node to your index:

```
<param name="org.opencms.search.CmsSearchIndex.useBackupReindexing">true</param>
```

This will create a snapshot as explained here: http://wiki.apache.org/solr/CollectionDistribution

11.5.3.10 Solr result size gets limited to 50 by default, how to get more than 50 results?

In order to return only permission checked resources (what is an expensive task) we only return this limited number of results. For paging over results please have a look at at the Solr parameters: rows and start: http://wiki.apache.org/solr/CommonQueryParameters since version 8.5.x you can increase the resulting documents to a size of your choice.

11.5.4 Solr mailing list questions

11.5.4.1 A class cast exception is thrown, what can I do?

You have to set the right classes for the index, and the field configuration otherwise the Lucene search index implementation is used.

```
<index class="org.opencms.search.solr.CmsSolrIndex">[...]</index>
<fieldconfiguration class="org.opencms.search.solr.CmsSolrFieldConfiguration">[...]
</fieldconfiguration>
```

11.5.4.2 Is it possible to map elements with maxoccurres > 1?

Since version >= 8.5.1 they are mapped to a multivalved field.

11.5.4.3 How to index OpenCmsDateTime elements?

```
<searchsetting element="Release" searchcontent="false">
    <solrfield targetfield="arelease" sourcefield="*_dt" />
</searchsetting>
```

12 SEO

12.1 Introduction

OpenCms 8.5 contains two new features for the purpose of Search Engine Optimization: Aliases and SEO files. Aliases allow you to have alternative URLs for your pages which don't correspond to actual paths in the VFS. SEO files can be used to automatically generate XML sitemaps and robots.txt files for your site.

12.2 Aliases

12.2.1 Simple aliases

There are two types of aliases which can be defined: **Simple aliases** and **Rewrite aliases**. Simple aliases consist of an **alias path**, a **target resource**, and an **action** that determines what happens when the path is requested from OpenCms. Simple aliases are directly connected to their target resource, so they will continue to point to that resource even if it is moved or renamed. Valid alias paths consist of one or more segments which each consist of a "/" and one or more characters. This means that they may not end with a trailing "/". There are three possible actions which can be peformed if the path of an alias is requested from OpenCms:

- Temporary redirect (302): A temporary redirect will be sent to the browser, with the current link to the target resource as the new URL
- Permanent redirect (301): A permanent redirect will be sent to the browser, with the current link to the target resource as the new URL
- Show page: OpenCms will try to load the target resource directly in the current request

12.2.2 Rewrite aliases

While a simple alias maps a single path to a single resource, rewrite aliases can be used to define aliases for whole classes of paths by specifying a regular expression to match a path and a replacement string to apply if the pattern matches. OpenCms will test an incoming path against rewrite alias patterns and apply the first matching rewrite alias. There is no order defined for the matching, so you should not define rewrite aliases with overlapping patterns. The pattern for a rewrite alias follows standard Java regular expression syntax, while the replacement string follows the syntax for the parameter of the method java.util.regex.Matcher#replaceFirst, i.e. the content of capture groups from the regular expressions can be accessed using dollar syntax ($1, $2,...). The pattern will always be matched against the whole path. Rewrite aliases have precedence over simple aliases, i.e. if a rewrite alias matches the current request's path, a simple alias for that path will not match. There are three possible actions for rewrite aliases:

- Temporary redirect (302): A temporary redirect will be sent to the browser, with the substitution result as the new URL
- Permanent redirect (301): A permanent redirect will be sent to the browser, with the substitution result as the new URL
- Passthrough: The path resulting from the pattern replacement will be passed to the next resource handler configured in the <resourceinit> element after the alias handler.

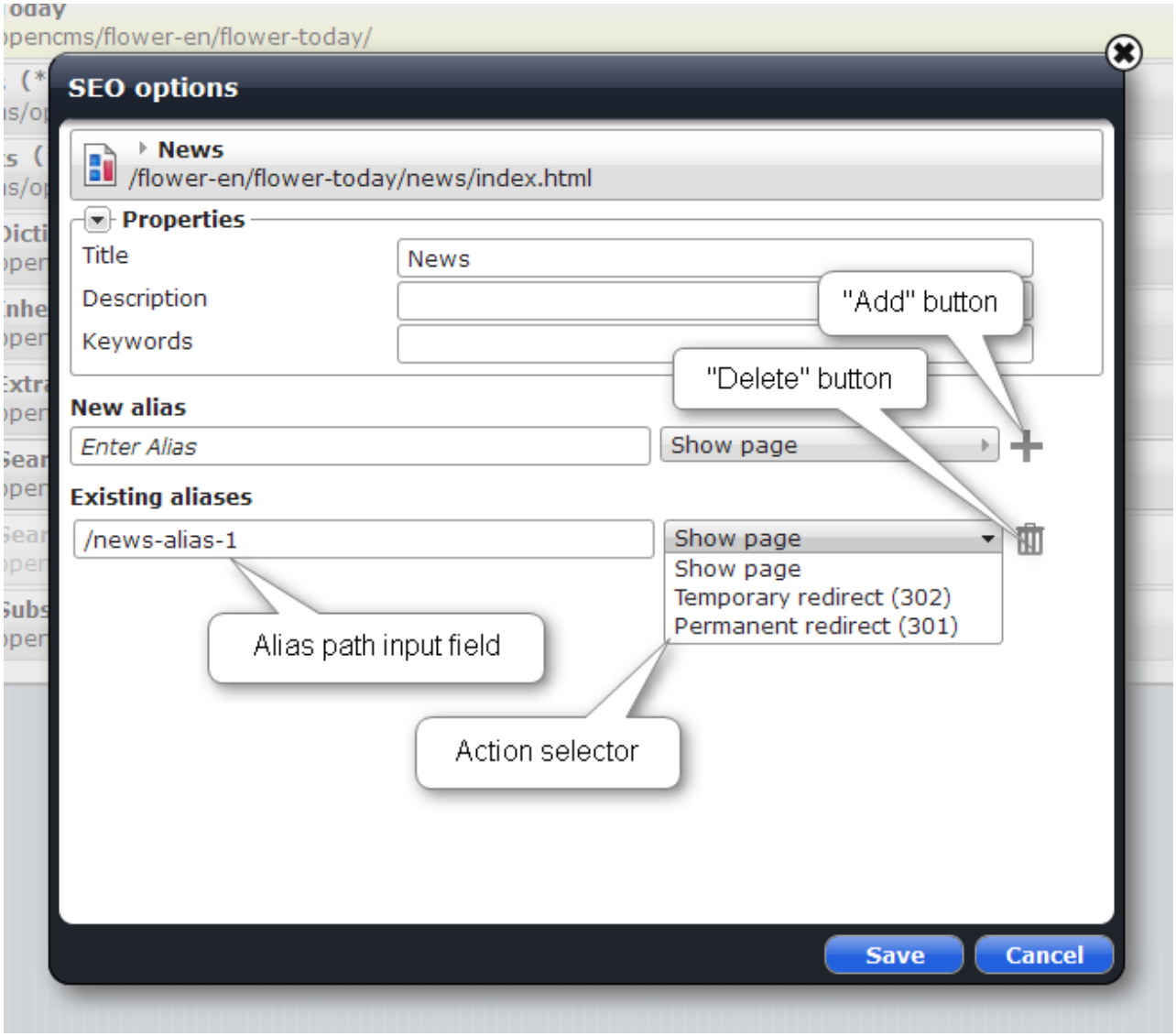
12.2.3 Internals

All alias matching for incoming requests is handled by the class org.opencms.main.CmsAliasResourceHandler, which is by default configured in the opencms-system.xml configuration file in the <resourceinit> element. Even if this handler is removed from the configuration, it is still possible to edit aliases through the user interface, although they will have no effect.
For paths which actually exist in the VFS, and for paths that start with "/system/", the alias resource handler will not be used.
Note that aliases are only active for a single site. For example, it is possible to use the same alias path for different simple aliases on different sites.
Note that, due to the limitations of redirects, request parameters will not be preserved if a POST request's path matches an alias path and a redirect action is performed.

12.2.4 SEO options dialog

A new SEO options Context menu entry is available in the Page editor's Context menu and in the Context menu for sitemap entries:

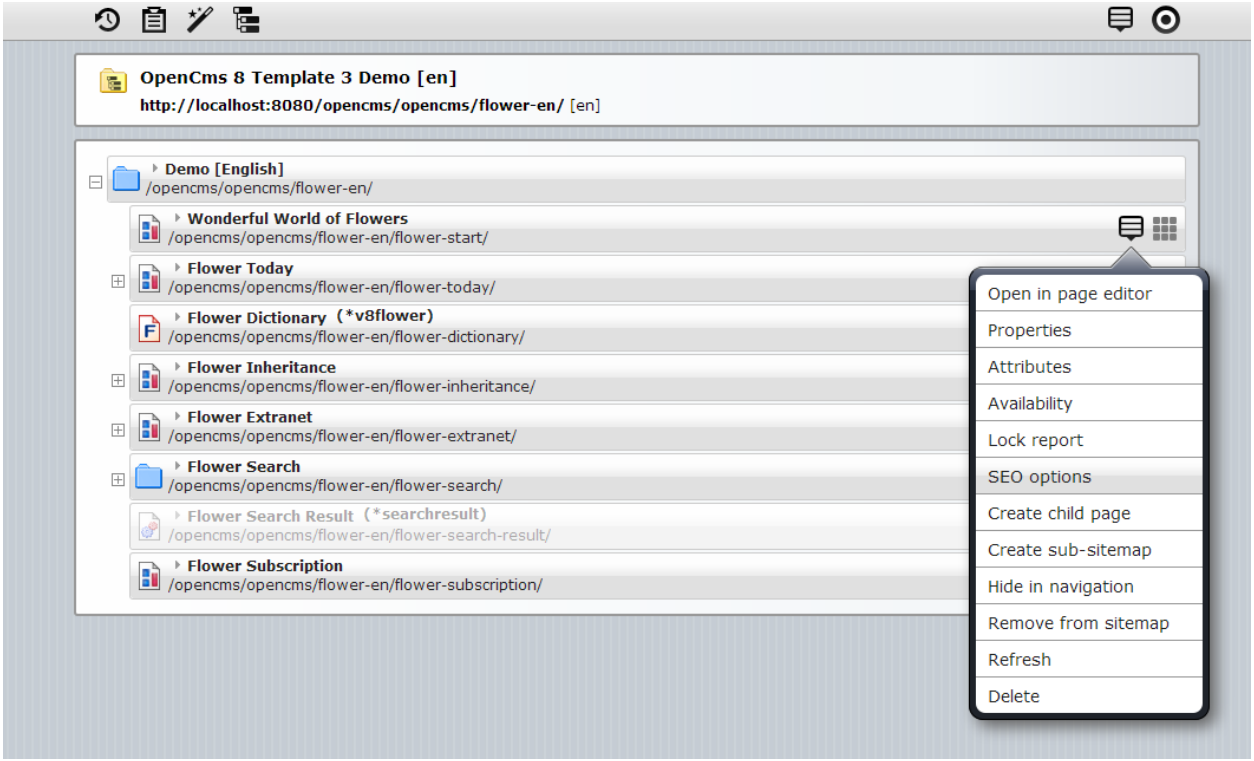
When this item is selected from the Context menu, the SEO options dialog opens:



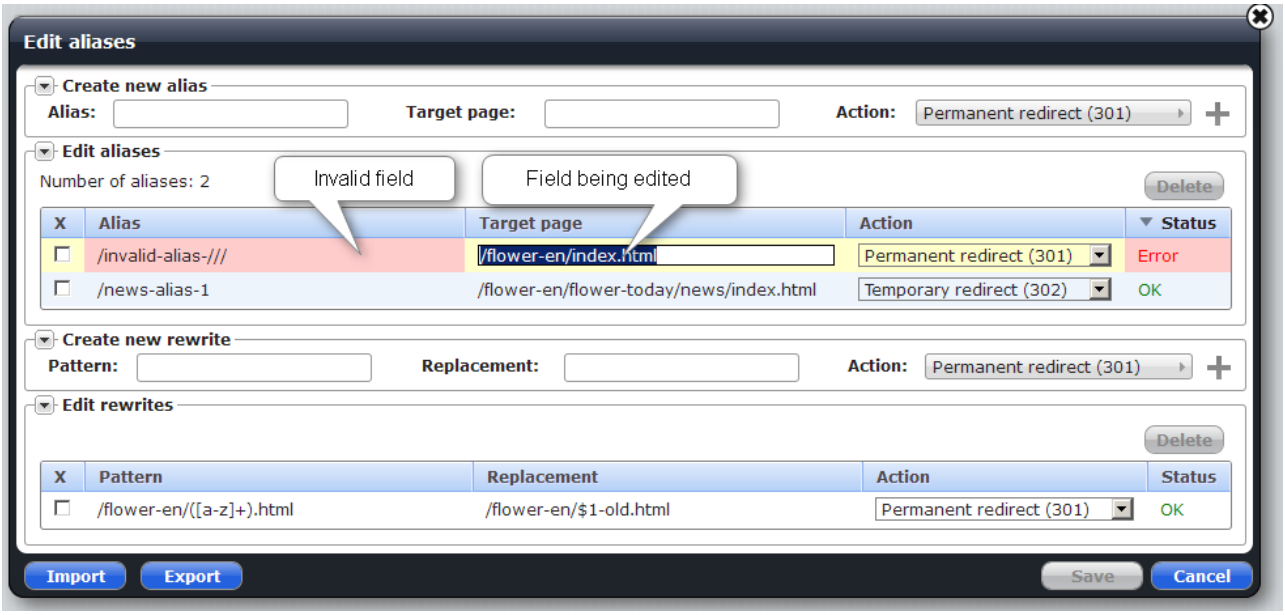
The SEO options dialog allows you to change the Title, Description and Keywords properties, and to edit the simple aliases for the resource. Note that changes will only be applied once you click the "Save" button.
You can edit an existing alias by changing its path in the text box or changing the action using the action select box. The "Delete" button can be used to remove the alias.
You can add a new alias by entering the alias path in the text box for new alias paths, selecting the action from the select box, and clicking the "Add" button.

12.2.5 Edit aliases dialog

The "Edit aliases" dialog allows you to view or edit all aliases (simple and rewrite) for the current site. You can open it from the sitemap editor's main Context menu.



When this item is selected from the Context menu, the SEO options dialog opens:



This dialog allows you to edit new aliases, edit existing aliases, and import and export aliases for the current site to/from a CSV file.

12.2.6 Creating new aliases

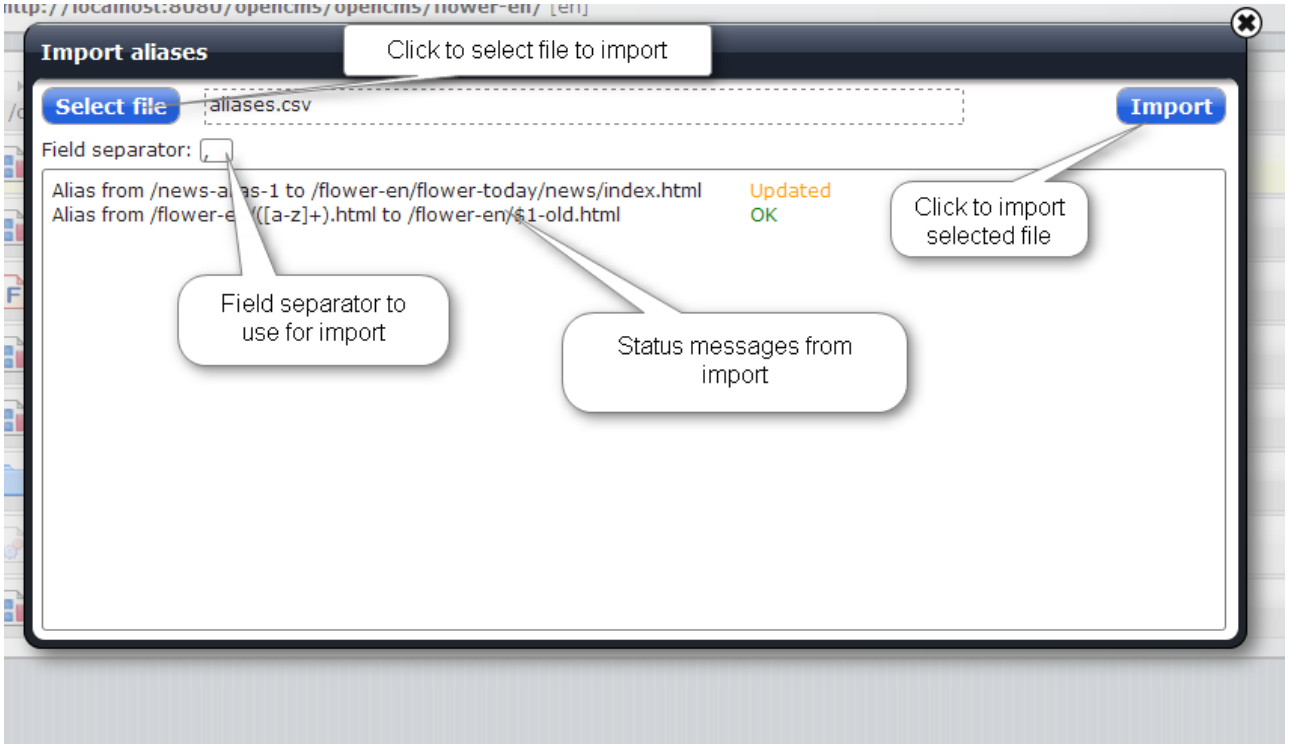
You can create a new alias by entering the alias path and target page (or pattern and replacement, respectively), selecting an action and clicking the "Add" button.

12.2.7 Editing existing aliases

Existing aliases (either simple or rewrite aliases) can be edited in their respective tables. You can change an existing alias' attributes by clicking on the corresponding table entry. If you enter invalid alias paths, the "Status" field of the corresponding table will display an "Error" label; you can get a more detailed message when hovering your mouse over this label.
You can delete existing aliases by selecting the checkboxes in the "X" column of the corresponding table rows, and then clicking the "Delete" button above the table.

12.2.8 Exporting and importing aliases

You can click the "Export" button to download a CSV file containing the currently defined aliases for the site. Note that this will not include any unsaved alias changes from the dialog.
Clicking the "Import" button will open another dialog that allows you to import a previously exported CSV file.



First click the "Select File" button to select a CSV file to import. Then click the "Import" button. It is possible that the CSV file was generated by a program which uses a different field separator than ',', so you can also set the field separator to use when parsing the file before clicking on "Import". The aliases will then be imported, and status messages will be displayed in the text field on the bottom of the dialog. The import CSV file format looks like the following:

```
"/news-alias-1","/flower-en/flower-today/news/index.html","redirect"
"/news-alias-2","/flower-de/flower-today/news/index.html"
"/flower-en/([a-z]+).html","/flower-en/$1-old.html","permanentRedirect","rewrite"
```

Each line consists of two, three or four fields. If the line contains 3 fields, it is interpreted as a simple alias, with the following field definitions:

- Field 1: The alias path
- Field 2: The site path of the alias target
- Field 3: The action for the rewrite.
    - page: Show the page
    - permanentRedirect: Send permanent redirect
    - redirect: Send temporary redirect

Lines with 2 fields will also be interpreted as simple alias paths, with the alias mode implicitly set to "permanentRedirect". If the line contains 4 fields, it is interpreted as a rewrite alias, with the following field definitions:

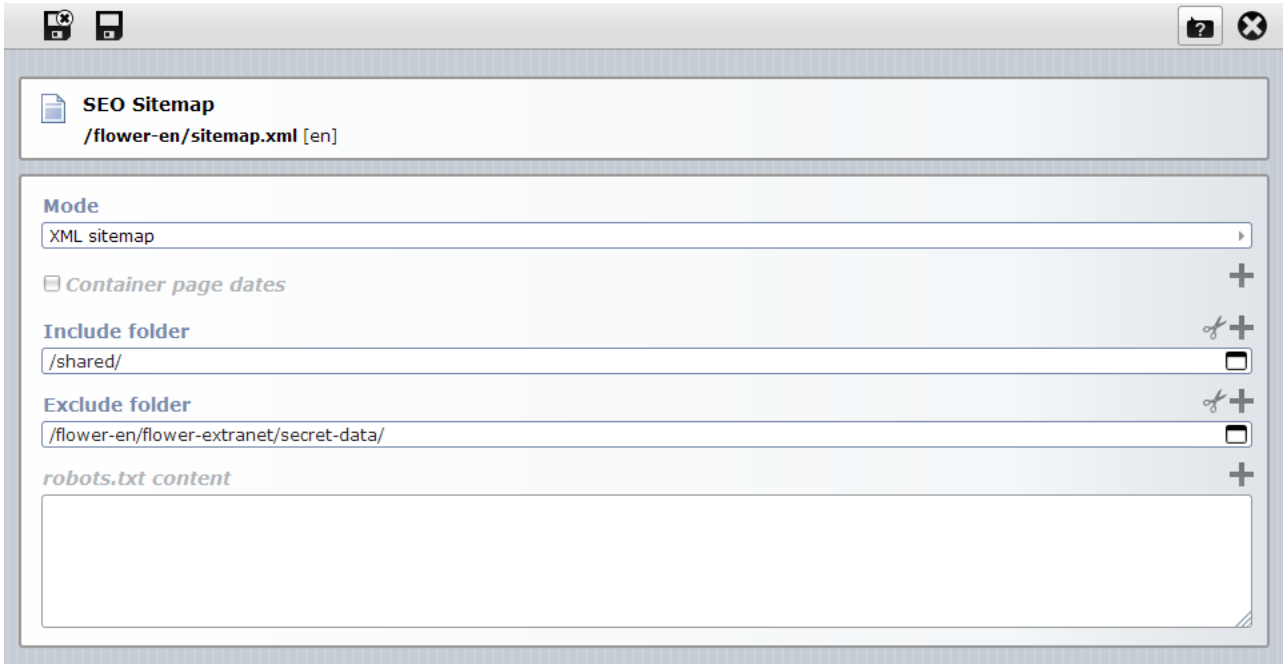
- Field 1: The alias pattern
- Field 2: The replacement string
- Field 3: The action for the rewrite
    - passthrough: Pass the rewritten path to the next resource handler
    - permanentRedirect, redirect: as above
- Field 4: Must be the constant value "rewrite"

12.3 Automatic robots.txt and XML sitemap generation

OpenCms provides automatic generation of XML sitemaps and robots.txt files which reference those sitemaps. Both of these are provided by the new seo_file resource type. When a resource of this type is requested from OpenCms, OpenCms will dynamically generate the sitemap or robots.txt data and send them to the client.

12.3.1 XML sitemap generation

Create a new resource of the type "SEO file" in the folder for which you want to generate the XML sitemap. Open this file with the content editor. Make sure the mode "XML sitemap" is selected.



By default, the XML sitemap will contain all navigation entries in and below the current folder, the URLs for all detail pages whose base container pages are in or below the current folder, and all aliases whose alias paths are below the current folder and whose action is defined as "Show page".
"Include folder" entries allows you to define folders whose whole contents, including resources from subfolders, will be included in the XML sitemap.
"Exclude folder" entries allow you to define folders whose whole contents, including resources from subfolders, will be excluded from the XML sitemap.
If you check the "Container page dates" option, the last modification date of containerpages will be computed from the last modification date of their contents.
Note that the XML sitemap output will always correspond to the Online project state of resources, and will only contain pages which are visible to the Guest user.

12.3.2 Generation of robots.txt

To use this feature, create a new file of type "SEO file" in the root folder of your site. Using this feature only makes sense if your host is set up so that the OpenCms site is mapped to the root of your domain, i.e. that http://yourhost.com/robots.txt will be retrieved from the OpenCms site's root folder.
Edit the SEO file and set the mode to "robots.txt". By default, when the robots.txt file is requested from OpenCms, it will only contain references to the existing XML sitemaps of your site. To add additional entries, you need to add them to the "robots.txt content" field of your SEO file.

13 CMIS

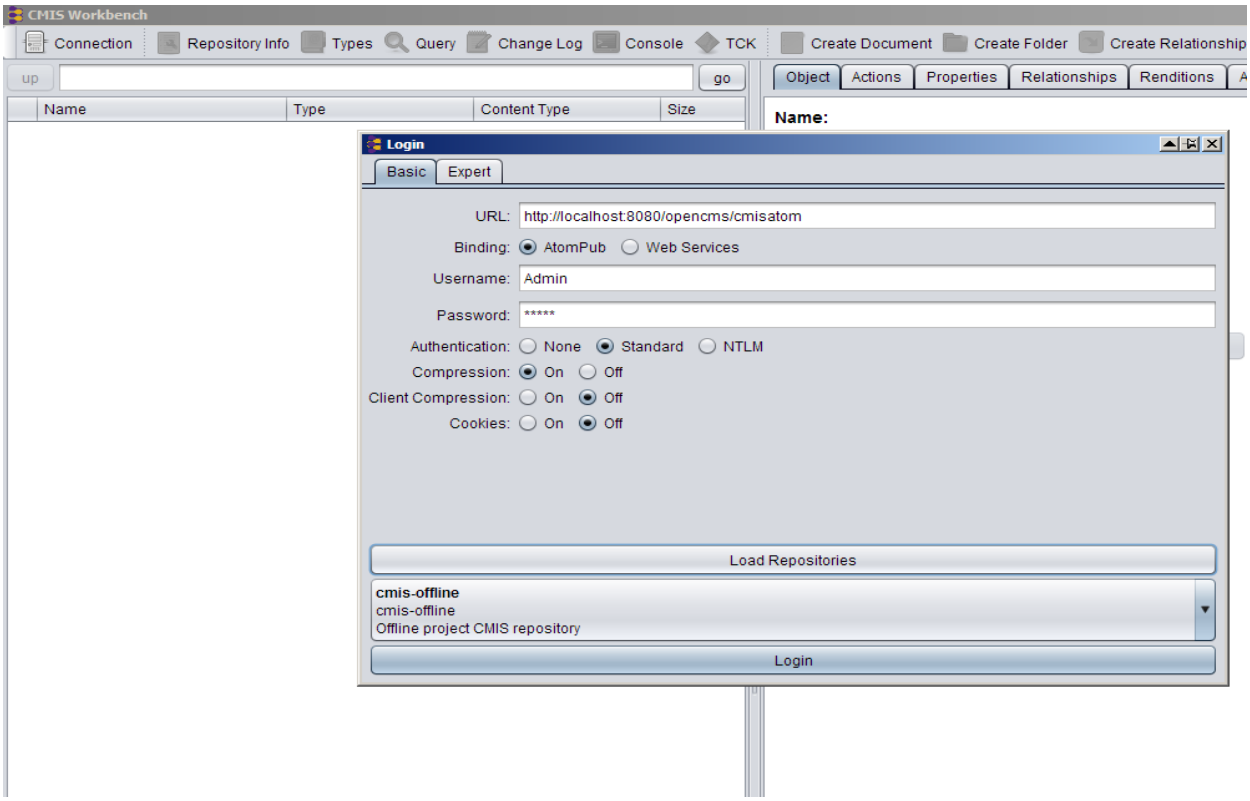
With OpenCms 8.5, it is possible to access the content inside the OpenCms Virtual File System using CMIS, the Content Management Interoperability Services. The next section will consist of a small practical example of accessing a running OpenCms instance via CMIS, while the section after that will contain more detailed information about the CMIS implementation of OpenCms. Please note that this document does not describe the CMIS standard in depth. You can read the CMIS specification at http://docs.oasis-open.org/cmis/CMIS/v1.0/cmis-spec-v1.0.html.

13.1 Accessing OpenCms via CMIS

Version 8.5 and later come with CMIS connectivity out of the box, so no configuration is required on the OpenCms side. We assume that an OpenCms instance is running locally under http://localhost:8080/opencms, where opencms is the web application name.

We will be using the Apache Chemistry Workbench for this example. This is a Java GUI client application for accessing arbitrary CMIS repositories. You can download it from http://chemistry.apache.org/java/download.html.

Start the Apache Chemistry Workbench and click the "Connection" button. The following dialog will open:



The CMIS standard defines both an AtomPub binding and a SOAP web service binding. OpenCms supports both of them, but we are using the AtomPub binding for this example, so enter the connection URL "http://localhost:8080/opencms/cmisatom" and select the "AtomPub" radio button.

Enter the user name and password of the OpenCms user as which you want to log in (all operations performed via CMIS are executed in the context of an OpenCms user). Then click on the "Load repositories" button.

Two repositories will be displayed: cmis-online, with which you can access contents from OpenCms' Online project, but not make any modifications, and cmis-offline, which allows you to access and make changes to the offline contents. Select "cmis-offline" and click the Login button.

The GUI will now display the resources from the root folder of the VFS. You can navigate to other folders by double clicking on them. Double clicking on other resources will load the resources from the VFS and display them. Please note that CMIS only allows you to access the raw content of the resources. More specifically, you cannot get the rendered output of XML content via CMIS, only its source code.

On the workbench's right side, you can select different tabs for access various information about or perform actions on the currently selected resource on the left. For example, the "Actions" tab allows you to perform actions like deleting the current resource, while in the "Properties" tab, you can view or modify the properties for the currently selected resource.

13.2 CMIS integration

CMIS is a standard for accessing content repositories over web services. OpenCms provides a CMIS interface through which CMIS client software can access the OpenCms VFS. For that purpose, OpenCms comes with two additional servlets configured in the WEB-INF/web.xml file for the two possible binding types defined by the CMIS standard, AtomPub and SOAP. The servlets are mapped to the paths /cmisatom and /services, respectively.

Additionally, some servlet context listeners are defined in the web.xml file. These servlets are provided by the Apache Chemistry server implementation framework. If you do not intend to use CMIS at all, you can just comment out the CMIS section in the web.xml file marked by "Start of/End of CMIS configuration", which will reduce the startup time of the OpenCms web application. These servlets provide access to any CMIS repositories which have been configured in opencms-importexport.xml. CMIS repositories, like WebDav repositories, are configured under the <repositories> element. By default, this configuration has two configured repositories:

```
<repositories>...
<repository name="cmis-offline" class="org.opencms.cmis.CmsCmisRepository">
 <params>
   <param name="project">Offline</param>
   <param name="description">Offline project CMIS repository</param>
   <param name="index">Solr Offline</param>
       </params>
</repository>
<repository name="cmis-online" class="org.opencms.cmis.CmsCmisRepository">
 <params>
   <param name="project">Online</param>
   <param name="description">Online project CMIS repository</param>
   <param name="index">Solr Online</param>
 </params>
</repository>
...
</repositories>
```

The <repositories> section of the configuration is also used to configure WebDAV repositories for OpenCms. OpenCms distinguishes configured CMIS repositories from WebDAV repositories by the fact that the value of the "class" attribute refers to the name of a Java class which implements the interface org.opencms.cmis.I_CmsCmisRepository.

Currently the only class implementing this interface that comes with OpenCms is org.opencms.cmis.CmsCmisRepository. This class supports various configuration parameters which are defined under the “params” element:

project:
: The "working project" for the CMIS repository. This is a required parameter. If this is the Online project, the repository will only provide read access to the Online project of the VFS.
If this is any other project, the CMIS repository will access the Offline state of the VFS, and any changes are performed inside the configured project. Note that, from the point of view of the CMIS standard, different repositories are independent objects. This means that if you are writing a client application that needs to access both offline and online contents from OpenCms through CMIS, you will need to connect to both an Offline and an Online repository.

description:
: A description text that may be displayed by CMIS clients for the repository.

index:
: The name of the OpenCms Solr index used for querying the repository. This will **not** work with a Lucene index. This uses OpenCms' standard Solr search facilities, which are described elsewhere in this manual.If this parameter is omitted, the CMIS implementation will report the repository as "not queryable".

rendition:
: The class name of a "rendition provider class". CMIS has the concept of 'renditions', which are alternative views of the the same content, e.g. thumbnails of images, and the class which is configured here can provide such renditions. The class needs to implement the interface org.opencms.cmis.I_CmsCmisRenditionProvider. No classes implementing this interface are included in the standard OpenCms distribution.

property:
: The class name of a "property provider" class. With this parameter, it is possible to add special properties to the CMIS view of the OpenCms VFS which do not directly correspond to any resource properties or attributes, but are instead read or written by calling methods on an instance of the provided class. You can use this to make custom code in OpenCms available through CMIS calls. The configured classes need to implement the interface org.opencms.cmis.I_CmsPropertyProvider. No classes implementing this interface are included in the standard OpenCms distribution.

OpenCms makes resources and relations available as CMIS objects. All files have the CMIS type "cmis:document". All folders have the CMIS type "cmis:folder". Relations have the CMIS type "opencms:RELATIONTYPE", where RELATIONTYPE is the relation type name defined in OpenCms.

Files and folders can be created, deleted and modified through CMIS, assuming the repository project is not the Online project and the current user has the permissions to perform the operation. Relations can only be created or deleted if the relation type is nor marked as “defined in content” (See the method org.opencms.relation.CmsRelationType.isDefinedInContent()).

For each OpenCms property "X", there are two CMIS properties available on both cmis:document and cmis:folder: opencms:X and opencms-inherited:X. The opencms:X property for a CMIS document or folder will contain the property value for the resource which is directly set on the resource, while the opencms-inherited:X property will contain the value which was either directly set on the resource or inherited from a parent folder. The opencms:X property can be both read and written, while the opencms-inherited:X property is read-only. Writing to the opencms:X property with CMIS will set both the structure and the resource value of the property in OpenCms.

Some standard CMIS properties are filled with resource attributes or other useful resource information from OpenCms:

cmis:objectId - the internal OpenCms structure id of the resource
cmis:path - the root path of the resource
cmis:lastModifiedBy - the name of the user who last modified the resource
cmis:lastModificationDate - the date at which the resource was last modified
cmis:createdBy - the name of the user who created the resource
cmis:creationDate - the date when the resource was created
opencms-special:resource-type - Contains the OpenCms resource type name
opencms-dynamic:PROPERTYNAME - A dynamic property which is handled by a custom property provider class configured in opencms-importexport.xml, as described above.

It is not possible to create completely new OpenCms properties through CMIS. You can only define new properties from OpenCms itself. These new properties may not be immediately visible through the CMIS interface, as the list of valid OpenCms properties in the CMIS implementation is only refreshed every 5 minutes.

When uploading new files into OpenCms via CMIS, the OpenCms resource type will be automatically determined from the file extension of the uploaded file. You can override this by specifying the opencms-special:resource-type property with the OpenCms resource type as a value when creating a new document through CMIS.

Limitations of the CMIS implementation
The CMIS implementation of OpenCms does not support all optional CMIS features, and not all OpenCms functionality. The unsupported features are:

- Versioning and PWCs
- Unfiled and multifiled resources
- Changing permissions/access control entries
- Changelogs
- Policies
- CMIS-SQL support

Although queries don't support CMIS-SQL, querying is still possible using Solr queries. If a Solr index is configured for a repository as described above, OpenCms will report that repository as queryable, but standard CMIS clients that rely on CMIS-SQL being available for queryable repositories may not work.

Publishing files is not supported, since the CMIS standard has no concept of “Offline” and “Online” mode like OpenCms. The Offline and Online repositories in the standard configuration are completely independent from the CMIS point of view. Although the CMIS implementation provides basic access control lists for resources, those are very incomplete compared to the OpenCms permission system, and are not sufficient to determine whether a user can perform an action through CMIS. So if you writing a client application to access OpenCms through CMIS, you need to use the "allowable actions" API call to determine whether an operation can be performed before actually trying to perform it.

Example CMIS client program

Thanks to CMIS, you can use any client library which correctly implements CMIS to write custom client applications that access the VFS. One such exmple application that uses the Apache Chemistry client libraries is appended below. This example application uploads documents from a folder in the local file system to the RFS. You can find more information about developing client applications with Apache Chemistry in Java at http://chemistry.apache.org/java/developing/index.html

```
/**
* Demo for uploading files to CMIS.
*/
public class CmisUploader {

  /** The repository id. */
  public static final String REPOSITORY_ID = "cmis-offline";

  /** The session factory. */
  private static SessionFactory sessionFactory = SessfromionFactoryImpl.newInstance();

  private String m_repositoryUrl;

  /** The session. */
  private Session m_session = null;

  public CmisUploader(String repositoryUrl) {

      m_repositoryUrl = repositoryUrl;
  }

  /**
   * Main method.
   */
  public static void main(String[] args) throws Exception {

      String repositoryUrl = args[0];
      String source = args[1];
      String target = args[2];
      CmisUploader uploader = new CmisUploader(repositoryUrl);
      uploader.copyFiles(source, target);
  }

  /**
   * Copies files from a local folder to a CMIS folder.
   */
  public void copyFiles(String sourceFolder, String targetFolder) throws Exception {

      File sourcefolder = new File(sourceFolder);
      for (File child : sourcefolder.listFiles()) {
```

```
            if (child.isFile()) {
                uploadFileToCmis(child.getAbsolutePath(), targetFolder);
            }
        }
    }

    /**
     * Uploads a single file from the local file system to a CMIS folder.
     */
    public void uploadFileToCmis(String fileSystemPath, String targetFolder) throws Exception {

        Session session = getSession();
        File file = new File(fileSystemPath);
        String name = file.getName();
        byte[] fileContent = readFile(file);
        Folder parent = (Folder)(session.getObjectByPath(targetFolder));
        String targetPath = (targetFolder + "/" + name).replaceAll("/+", "/");
        try {
            session.getObjectByPath(targetPath);
            System.out.println("Not creating " + targetPath + " since it already exists. ");
        } catch (CmisObjectNotFoundException e) {
            System.out.println("Creating " + targetPath);
            Map<String, Object> properties = new HashMap<String, Object>();
            properties.put(PropertyIds.OBJECT_TYPE_ID, "cmis:document");
            properties.put(PropertyIds.NAME, name);
            ContentStream contentStream = new ContentStreamImpl(
                name,
                BigInteger.valueOf(fileContent.length),
                "binary/octet-stream",
                new ByteArrayInputStream(fileContent));
            parent.createDocument(properties, contentStream, VersioningState.MAJOR);
        }
    }

    /**
     * Gets the session, creating it if possible.
     */
    private Session getSession() {

        if (m_session == null) {
            Map<String, String> parameters = new HashMap<String, String>();
            // OpenCms user name and password
            parameters.put(SessionParameter.USER, "Admin");
            parameters.put(SessionParameter.PASSWORD, "admin");
            parameters.put(SessionParameter.ATOMPUB_URL, m_repositoryUrl);
            parameters.put(SessionParameter.BINDING_TYPE, BindingType.ATOMPUB.value());
            parameters.put(SessionParameter.REPOSITORY_ID, "cmis-offline");
            m_session = sessionFactory.createSession(parameters);
        }
        return m_session;
    }

    /**
     * Utility method for reading a file's content.
     */
    private byte[] readFile(File file) throws IOException {

        byte[] fileContent = new byte[(int)file.length()];
        FileInputStream istream = new FileInputStream(file);
        istream.read(fileContent);
        istream.close();
        return fileContent;
    }
}
```

14 JSP basics

This part of the documentation describes the basics of JSP application development with OpenCms. It provides an overview and some background information about the technology used for the OpenCms JSP integration. Other sections available in the Alkacon OpenCms documentation accompany this introduction and serve as further reference material. Please note that this is not an introduction to JSP developing, there are certainly many good tutorials or books for that around.

14.1 JSP features

- Add and manage JSP in the OpenCms Workplace
- Supports WYSIWYG editable pages which use JSP templates
- Use the same JSP template for editable pages and interactive forms
- JSP Taglib for common OpenCms tasks
- JSP API to directly access OpenCms functionality
- Separate online and offline versions of the same JSP
- The FlexCache, a powerful declarative caching mechanism
- Optional output streaming on a per-page basis
- Optional static export of JSP

14.2 The JSP `<cms>-taglib`

14.2.1 How to insert the "taglib" directive?

If you like to use the OpenCms JSP taglib, you **must** insert the **`"taglib"`** directive in your JSP page to specify where the OpenCms taglib definition can be found. This can be done in two different ways:

1. With standard JSP-/Servlet-Technology:

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %>
```

This directive must be set **before** any `cms` tags are used. If used like shown in the example, the prefix `"cms"` is unique for the OpenCms Taglib. In our examples, all tags of the OpenCms taglib start with the prefix `"cms:"`.

2. With OpenCms specific short form:

```
<%@page buffer="none" session="false" taglibs="c,cms" %>
```

14.2.2 Available tags

The following list of **`<cms>-tags`** is available in the OpenCms standard taglib. An tag can be executed by calling:

```
<cms:tagName attributes...></cms:tagName>
```

- **cms:container** - Enables the template mechanism for container pages.
- **cms:contentaccess** - Provides access to the content for the JSP EL.
- **cms:contentcheck** - Provides conditional logic for checking the element of a XML content.
- **cms:contentinfo** - Provides access to the result set of a contentload.
- **cms:contentload** - Loads XML content items from the OpenCms VFS.

- **cms:contentloop** - Allows looping through XML content node element values.
- **cms:contentshow** - Provides access to individual XML content node element values.
- **cms:decorate** - Provides decoration of HTML content.
- **cms:device** - Provides different HTML output for various device types.
- **cms:editable** - Enables the direct editing within a template.
- **cms:elementsetting** - Provides access to the settings of an ADE container element.
- **cms:enable-ade** - Enables the advanced direct editing within a template.
- **cms:export** - Writes JSP code from a JSP to files in the static export.
- **cms:formatter** - Loads single XML content items for ADE formatters.
- **cms:headincludes** - Includes required css or javascript resources, placed in html/head.
- **cms:img** - Supports image scaling using the OpenCms native image scaling mechanism.
- **cms:include** - Includes other JSP elements dynamically at runtime.
- **cms:info** - Enables access to some system information like OpenCms version etc.
- **cms:jquery** - Allows to include jquery and some jquery plugins code and style sheets.
- **cms:label** - Reads localized values from the resource bundles (Java property files like workplace.properties) that hold user messages.
- **cms:link** - Builds a valid OpenCms URL for a given resource.
- **cms:navigation** - Provides access to the navigation information.
- **cms:nocache** - Sends no-cache headers to the browser.
- **cms:param** - Adds a parameter to the outer tag (if supported).
- **cms:parse** - Decorates HTML with custom A_CmsConfiguredHtmlParser implementations given in the parserClass attribute.
- **cms:property** - Enables read access to the current VFS file's properties.
- **cms:resourceaccess** - Provides access to the resources for the JSP EL.
- **cms:resourceload** - Loads resources from the OpenCms VFS.
- **cms:secureparams** - Enables automatic parameter escaping for the current flex request.
- **cms:template** - Splits a JSP page into elements to be included by the `'cms:include'` tag.
- **cms:user** - Enables access to the properties of the currently logged in user.
- **cms:usertracking** - Performs user tracking actions like visit resources or subscribe/unsubscribe.

14.2.3 Available EL functions

The following list of EL-functions is available in the OpenCms standard taglib. An EL-function can be accessed by calling:

```
${cms:functionName(parameters)}
```

- **cms:vfs** - Provides simple access to a OpenCms JSP / EL content vfs access bean.
- **cms:convertDate** - Allows conversion of Long values to Dates. Can also handle Strings that represent a Long or a Date.
- **cms:convertLocale** - Allows conversion of Objects to Locales. Can also handle Strings that are locales, or Locales itself.
- **cms:convertList** - Returns a list of attribute values specified by the attribute name of the items of the given list.
- **cms:getCmsObject** - Returns the current OpenCms user context from the page context.

- **cms:getListSize** - Returns the size of the given list.
- **cms:stripHtml** - Strips all HTML markup from the given input.
- **cms:trimToSize** - Returns a substring of the input, which is not longer then the given int value.
- **cms:convertUUID** - Allows conversion of String values to CmsUUIDs. Can also handle byte[] that are CmsUUIDs, or CmsUUID itself.
- **cms:getRequestParam** - Returns the value of a parameter from a String that is formatted for a GET request.
- **cms:getRequestLink** - Returns the link without parameters from a String that is formatted for a GET request.
- **cms:escape** - Encodes a String in a way that is compatible with the JavaScript escape function.
- **cms:unescape** - Decodes a String in a way that is compatible with the JavaScript unescape function.
- **cms:navUri** - Returns the current navigation URI.