OpenCms Days 2009

Technical Track:

Advanced XML Content & Widget Configuration

Andreas Zahner,
Alkacon Software GmbH.

Agenda

1. Introduction of new XML content features in OpenCms 7.5
   - Improved standard galleries
   - Enhanced image gallery
   - Grouping editor input fields with tabs
2. Widget Configuration
   - Standard gallery widgets
   - Enhanced image gallery widget
   - Displaying enhanced gallery widget contents on JSPs
   - HTML widget
   - Category widget
   - Select widget
   - File selector widget
3. Tab configuration
   - Configuring tabs

Introduction of new XML content features in OpenCms 7.5

Improved standard galleries

- All OpenCms galleries were completely rewritten

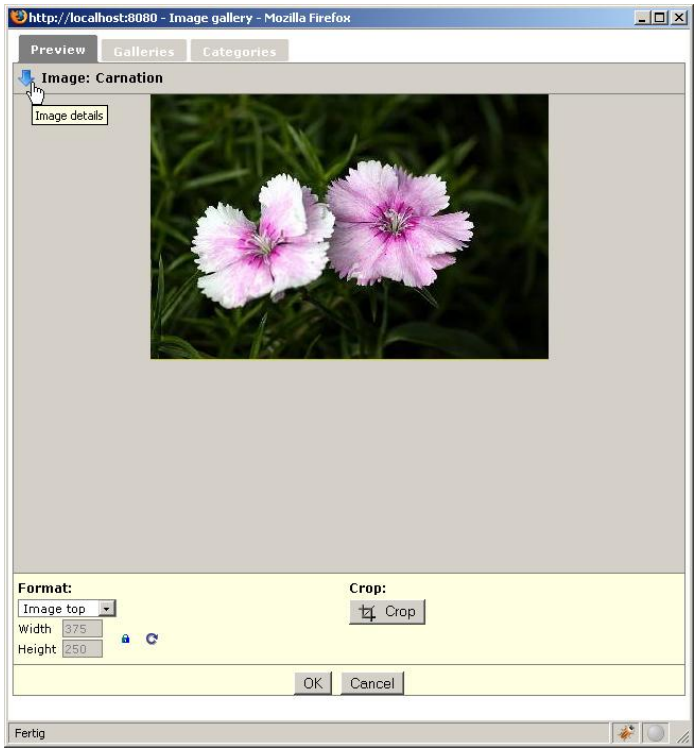
- The galleries can be used in
  - The structured content editor
  - The unstructured content editor (FCKeditor) using the FCKeditor dialog API
  - The OpenCms workplace explorer view

Improved standard galleries

- New features of the galleries
  - Select gallery items belonging to a specific category
  - Advanced AJAX features, e.g.:
    - show item information (last modification date)
    - change "Title" property value
  - Publish directly a new or changed gallery item
  - Publish changes made to gallery items like newly uploaded items or changed titles at once

Enhanced image gallery

- The enhanced image gallery can be used in
  - The structured content editor
  - The unstructured content editor (FCKeditor)
  - The OpenCms workplace explorer view

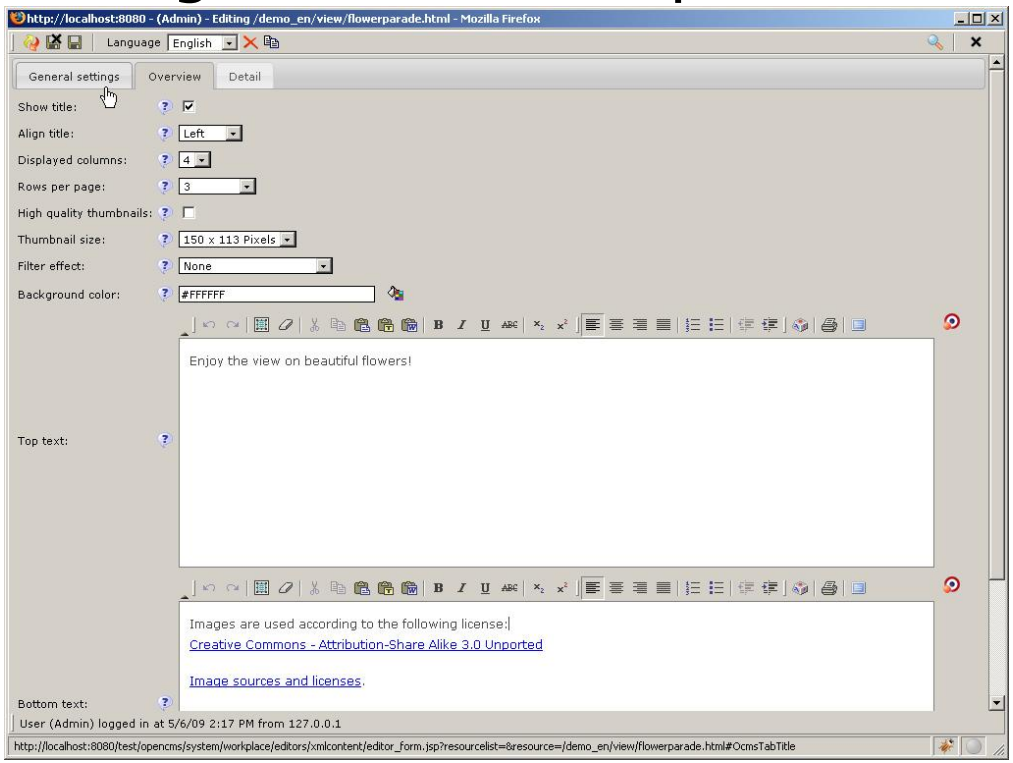


Enhanced image gallery

- New features of the enhanced image gallery
  - Select images belonging to a specific category
  - Scale images by using the integrated OpenCms image scaling capabilities
  - Crop images
  - Advanced AJAX features, e.g.:
    - show image information (image dimensions, size)
    - change "Title" property value
  - Publish directly new or changed image(s)
  - Select image alignment for unstructured content
  - Select a format from a list of predefined options for structured content

Grouping with tabs

- Improved XML content editor usability
  - Tabs can be defined for first level elements
  - Tab names can be localized
  - Configurable to collapse the first level or not



Widget configuration

Where to configure?

- Most widgets for the structured XML content editor can be configured in the XSD defining the XML content that is currently edited.

- All configuration options have to be added in the annotation section of the XSD:

```
<xsd:annotation>
  <xsd:appinfo>
   <layouts>
     <layout element="…" widget="…" configuration="…" />
```

- The configuration usually adds more functionality to the used editor widget like appearance of optional buttons or specific startup parameters

Standard gallery widgets

- The standard gallery widgets are:
  - DownloadGalleryWidget
  - LinkGalleryWidget
  - HtmlGalleryWidget
  - TableGalleryWidget

- These gallery widgets can be configured to open a specific gallery or category when they are opened from the XML content editor.

- This can be done by directly specifying a gallery/category path or by implementing an interface class.

Standard gallery widgets

- The following configuration options can be used for all standard gallery widgets:
  - **type**: defines the startup folder type that the widget should display when opened, either `'gallery'` or `'category'`
  - **startup**: defines the startup folder that the widget should display when opened, e.g. `'/demo_en/images/'`
  - **class**: optional class implementing the interface `I_CmsGalleryWidgetDynamicConfiguration` in the package `org.opencms.widgets`. This class can configure dynamic startup parameters.

Standard gallery widgets

- Using the interface
  - set `type` and `startup` configuration values to `'dynamic'` or an empty String
- Methods to implement
  - **`getStartup`**`(`
    `CmsObject cms,`
    `I_CmsWidgetDialog widgetDialog,`
    `I_CmsWidgetParameter param): String` (site path to gallery or category folder to open)
  - **`getType`**`(`
    `CmsObject cms,`
    `I_CmsWidgetDialog widgetDialog,`
    `I_CmsWidgetParameter param): String` (type of initial list to show, `'gallery'` or `'category'`)

Standard gallery widgets

- Example widget configuration

- Configuration String in XSD:

```
<layout … configuration="{
  class: 'org. … .CmsGalleryWidgetConfiguration',
  type: 'dynamic',
  startup: 'dynamic',
}"/>
```

- The configuration String is written as JSON (JavaScript Object Notation) object

Image gallery widget

- The standard image gallery widget can also be configured by using the options of the standard galleries.

- Difference:
  - The interface to implement is `I_CmsImageWidgetDynamicConfiguration` in the package `org.opencms.widgets`.

- Additional configuration option
  - **useformat**: defines if the format select box and the resize options for the image should be shown or not, with `true` or `false` (default) as possible values. Enables the possibility to change the image size or to crop the image for frontend view

Enhanced image gallery

- Usage of the enhanced image gallery widget
  - Dynamically place an image on your frontend page by offering a format selection
  - Optional description text for an image can be entered
- Configuration by setting JSON object values in the widget configuration of the XSD

```
<xsd:element name="Image" type="OpenCmsVfsImage" minOccurs="0" />
…
<xsd:annotation>
 <xsd:appinfo>
  <layouts>
    <layout element="Image" widget="VfsImageWidget" configuration="…" />
```

Enhanced image gallery

- Configuration options
  - **scaleparams**: default scale parameters used for scaled or cropped images, e.g. `'q:70,r:2,c:CCCC00'`.
    Note: No width, height or crop parameters should be defined here!
  - **type**, **startup**: defines the startup folder and startup folder type that the widget should display when opened, e.g. `type: 'gallery', startup: '/demo_en/images/'`
  - **usedescription**: indicates if the description input field for the for the image should be shown or not. Possible values are `true` or `false`, where `false` is the default.

Enhanced image gallery

- Configuration options
  - **useformat**: defines if the format select box for the image should be shown or not, with `true` or `false` (default) as possible values
  - **formatnames**: a list of possible format names that can be selected, with pairs of the value and the displayed selection text.
    Example:
    `'value1:Optiontext 1|value2:Optiontext 2'`
    The option texts can be localized using OpenCms macros like `%(key.localizedkey)`. The keys have to be defined in a workplace resource bundle.

Enhanced image gallery

- Configuration options
  - **formatvalues**: corresponding format values to the defined format names list. The list can contain width and height information, the `'?'` character means dynamic size and the `'x'` has to be used as separator, e.g. `['200x?', '400x300', '600x?']`
  - **class**: optional class implementing the interface `I_CmsImageWidgetDynamicConfiguration` in the package `org.opencms.widgets`. This class can configure dynamically startup parameters and format values.

Enhanced image gallery

- Using the interface
  - set `type` and `startup` configuration values to `'dynamic'` or an empty String
  - The `formatvalues` configuration should not be specified, it is generated by the class
- Methods to implement
  - **`getFormatValues`**`(`

```
CmsObject cms,
I_CmsWidgetDialog widgetDialog,
I_CmsWidgetParameter param,
List selectFormat,
List formatValues
```

`) : List` (of format values matching the format select options

Enhanced image gallery

- Methods to implement
  - **`getStartup`**`(`
    `CmsObject cms,`
    `I_CmsWidgetDialog widgetDialog,`
    `I_CmsWidgetParameter param): String` (site path to gallery or category folder to open)
  - **`getType`**`(`
    `CmsObject cms,`
    `I_CmsWidgetDialog widgetDialog,`
    `I_CmsWidgetParameter param): String` (type of initial image list to show, either '`gallery`' or '`category`')

Enhanced image gallery

- Example implementation
  - Dynamically sets the width and height values for the different image alignment options depending on the root folder of the edited resource (either english or german)
  - The initial gallery that is loaded also depends on the root folder
  - Configuration String in XSD:

```
<layout … configuration="{scaleparams: 't:0,q:70,r:2,c:FFFFFF',
  class: 'org.opencmsdays. … .CmsVfsImageWidgetConfigurationDemo',
  type: 'dynamic',
  startup: 'dynamic',
  usedescription: true,
  useformat: true,
  formatnames: 'imageleft:%(key.imagegallerdemo.format.left)|
    imageright:%(key.imagegallerdemo.format.right)|
    imagetop:%(key.imagegallerdemo.format.top)'
}"/>
```

Enhanced image gallery

- CmsVfsImageWidgetConfigurationDemo

```
/** Public empty constructor.<p> */
public CmsVfsImageWidgetConfigurationDemo() {

  // nothing to do here
}

public List getFormatValues(CmsObject cms, I_CmsWidgetDialog widgetDialog,
  I_CmsWidgetParameter param, List selectFormat, List formatValues) {

  // cast param to I_CmsXmlContentValue
  I_CmsXmlContentValue value = (I_CmsXmlContentValue)param;

  // now extract the absolute path of the edited resource
  String editedResource = cms.getSitePath(value.getDocument().getFile());

  // check if we are currently in the german demo folder
  boolean isGerman = editedResource.startsWith("/demo_de/");

  // this stores our result format values
  List result = new ArrayList(selectFormat.size());

  // iterate found options and determine the matching format values
  Iterator i = selectFormat.iterator();
```

Enhanced image gallery

- CmsVfsImageWidgetConfigurationDemo

```
  while (i.hasNext()) {
    CmsSelectWidgetOption currOpt = (CmsSelectWidgetOption)i.next();
    String formatName = currOpt.getValue();
    String formatValue = "400x?";
    if (formatName.indexOf("left") != -1) {
      // left image alignment
      if (isGerman) {
        formatValue = "150x?";
      } else {
        formatValue = "180x?";
      }
    } else if (formatName.indexOf("right") != -1) {
      // right image alignment
      if (isGerman) {
        formatValue = "250x?";
      } else {
        formatValue = "280x?";
      }
    } else {
      // image on top, fixed width and height value
      formatValue = "490x300";
    }
    // add the value to the result format list
    result.add(formatValue);
  }
  return result;
}
```

Enhanced image gallery

- CmsVfsImageWidgetConfigurationDemo

```
 public String getStartup(CmsObject cms, I_CmsWidgetDialog widgetDialog,
I_CmsWidgetParameter param) {

  // cast param to I_CmsXmlContentValue
  I_CmsXmlContentValue value = (I_CmsXmlContentValue)param;
  // now extract the absolute path of the edited resource
  String editedResource = cms.getSitePath(value.getDocument().getFile());
  if (editedResource.startsWith("/demo_de/") &&
      cms.existsResource("/demo_de/bilder/")) {
    // editing in the german demo folder, return german demo image gallery
    return "/demo_de/bilder/";
  } else if (cms.existsResource("/demo_en/images/")) {
    // in other cases, return english demo image gallery
    return "/demo_en/images/";
  }
  // preferred galleries do not exist, return nothing
  return null;
}

public String getType(CmsObject cms, I_CmsWidgetDialog widgetDialog,
I_CmsWidgetParameter param) {

  // for the demo, a fixed type "gallery" should be displayed
  return CmsVfsImageWidgetConfiguration.TYPE_GALLERY;
}
```

Displaying enhanced image

- Displaying images on the frontend
  - The saved values of the enhanced image gallery widget are accessed using the `CmsJspContentAccessBean`
  - Using the <cms:> taglib in combination with the expression language (EL) and the JSTL core taglib creates flexible frontend output depending on the values of the VFS image value.

Displaying enhanced image

- Accessing the stored image values

```
<cms:contentload collector="singleFile" param="%(opencms.uri)">
  <cms:contentaccess var="content" />

  <!-- Code example to access image element values -->
  <p>Accessing the image element values with
    <code>content.value.Image.xmlText['XPath']</code>:
  </p>

  <ul>

    <li>Link target:
        <cms:link>${content.value.Image.xmlText['link/target']}</cms:link></li>

    <li>Selected format: ${content.value.Image.xmlText['format']}</li>

    <li>Scale params: ${content.value.Image.xmlText['scale']}</li>

    <li>Description text: ${content.value.Image.xmlText['description']}</li>

  </ul>

</cms:contentload>
```

Displaying enhanced image

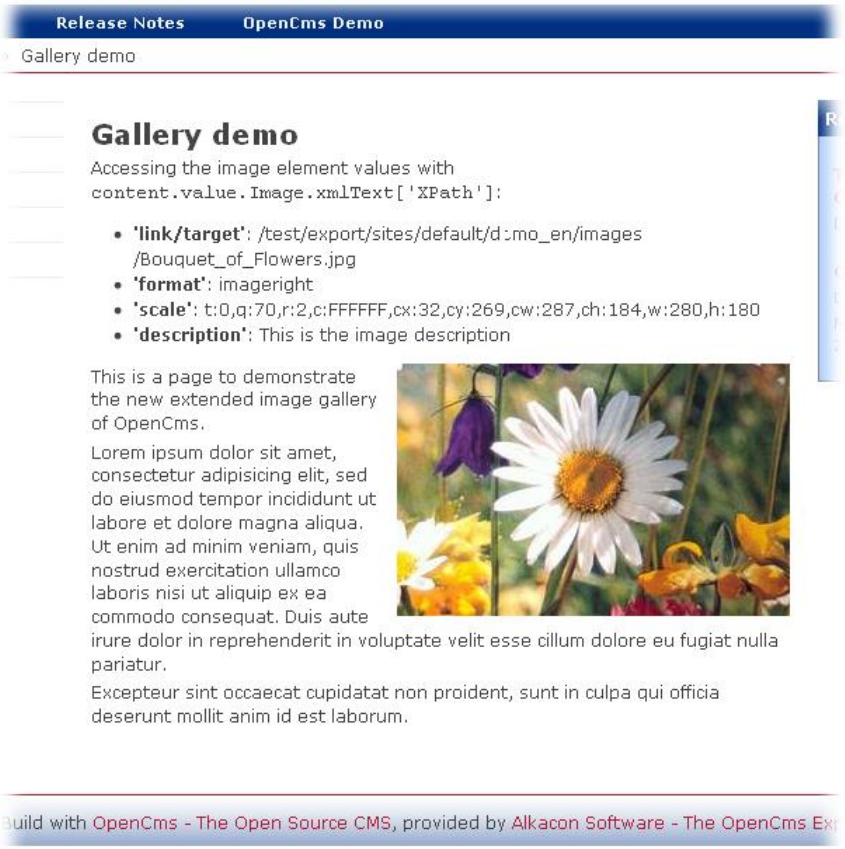
- Accessing the stored image values

```
<cms:contentload collector="singleFile" param="%(opencms.uri)">
  <cms:contentaccess var="content" />

  <!-- The text content of the paragraph with a positioned image -->
  <c:set var="imgAttrs" value="" />
  <c:choose>
    <c:when test="${content.value.Image.xmlText['format'] == 'imageleft'}">
      <%-- Left aligned image --%>
      <c:set var="imgAttrs">align="left" style="padding-right: 10px;"</c:set>
    </c:when>
    <c:when test="${content.value.Image.xmlText['format'] == 'imageright'}">
      <%-- Right aligned image --%>
      <c:set var="imgAttrs">align="right" style="padding-left: 10px;"</c:set>
    </c:when>
    <c:otherwise></c:otherwise>
  </c:choose>

  <%-- The image with scale parameter and additional alignment attributes --%>
  <img src="<cms:link>${content.value.Image.xmlText['link/target']}?__scale=
      ${content.value.Image.xmlText['scale']}</cms:link>"
    alt="${content.value.Image.xmlText['description']}"
    title="${content.value.Image.xmlText['description']}"
    ${imgAttrs}/>

  ${paragraph.value.Text}

</cms:contentload>
```

Displaying enhanced image

- Generated frontend output

```
<img src="/export/sites/default/demo_en/images/Bouquet_of_Flowers.jpg_1473387383.jpg"
   alt="This is the image description" title="This is the image description"
   align="right" style="padding-left: 10px;"/>
```



HTML widget

- The **HTML widget** can be configured to show optional buttons to edit the HTML code and to apply a different CSS to the editing area.

  The configurable buttons are:
  - **link**: shows the link dialog button
  - **anchor**: shows the anchor dialog button
  - **formatselect**: shows the format selector (headings, paragraph, etc.)
  - **fullpage**: enables the editor widget to edit a complete HTML page
  - **source**: shows the source code button
  - **image** or **imagegallery**: shows the image gallery button
  - **downloadgallery**: shows the download gallery button
  - **linkgallery**: shows the external link gallery button
  - **htmlgallery**: shows the html gallery button
  - **tablegallery**: shows the table gallery button
  - **table**: shows the insert/edit table dialog button

HTML widget

- The editor area can be configured to use a specific CSS style sheet file instead of the one specified by the template to render the contents. Additionally, a style selector can be configured if a special styles XML file is available. The following options can be added in the HTML widget configuration attribute:
  - **css:/absolute/path/to/stylesheet.css**
  - **stylesxml:/absolute/path/to/stylesfile.xml**
- The path to the files has to be an absolute path in the VFS of OpenCms.
- The height of the editor area can be chosen by adding the following to the configuration String:
  - **height:400px**

Category widget

- The category widget (CategoryWidget) allows to set categories for the edited file. It allows the following configuration options:
  - **onlyleafs**: if set to `true`, it is only allowed to select the leaf subcategory, it is not sufficient to select only a parent category
  - **category**: sets the category where the selection should start with, relative to the folder containing the categories. Example: `gewaechsarten/`
  - **property**: the name of the property where the start category should be read from, if this is set, the category option has no effect
- Configuration example:
  `onlyleafs=true|property=category`
  the starting category is read from the "category" property, only leafs can be selected

Select widget

- The options of a selector widget, multiselect widget or a combo box widget can be configured as follows:

```
<layout ... widget="SelectorWidget" configuration="Value 1:Option 1|Value 2*:Option 2" />

<layout ... widget="ComboWidget" configuration="value='Value 1' help='A help text for value 1'|value='Value 2' default='true' help='Another help text for the second value.'" />
```

- The preselected value can be marked with a "*" star symbol.
- Both syntax variants are possible to configure the widgets.

Select widget

- In most cases, it is sufficient to create a select box widget that fills the options in a customized way.
- To do so, the class **`org.opencms.widgets.CmsSelectWidget`** can be extended.
- The method
  `List parseSelectOptions(CmsObject cms, I_CmsWidgetDialog widgetDialog, I_CmsWidgetParameter param)`
  should be overwritten to create a list of `CmsSelectWidgetOption` objects.
- The constructor
  `CmsSelectWidgetOption(String value, boolean isDefault, String optionText)`
  creates a new instance of a select widget option that can be added to the list of available options

Select widget

- The created widget has to be registered in the OpenCms configuration file **`opencms-vfs.xml`**.
- The class and an alias has to be specified for each widget in OpenCms.

```
<widget class="com.mymodule.widgets.CmsMySelectWidget"
    alias="MySelectorWidget" />
```

File selector widget

- The options of the VFS file selector widget are as follows:

```
<layout ... widget="VfsFileWidget"
configuration="excludefiles|hidesiteselector|projectaware|startsit
e=/sites/default/" />
```

  - **`hidesiteselector, showsiteselector`**: the site selector is hidden or shown (default).
  - **`excludefiles, includefiles`**: files are not shown in the popup tree or included (default).
  - **`notprojectaware, projectaware`**: the folders and files are shown according to the current project or not (default).
  - **`startsite`**: the site the popup tree should be opened with.

Tab configuration

Configuring tabs

- Can be used to separate the input fields of structured contents into different tabs
- Especially useful if nested contents are used which occure exactly once
- Can be defined for XML elements on the first level only
- Tab configuration can optionally be defined in the XSD of a structured content
- Full automatic tab creation possible as well as individual configuration

Configuring tabs

- Full automatic tab creation:

```
<xsd:annotation>
 <xsd:appinfo>
 …
  <tabs useall="true" />
 …
```

- Every first level element of the XSD is used as tab
  - Name of the tab is the element name (or the localized element name from a message bundle)
  - The element name is collapsed, i.e. the element name or localized element name is omitted from the input form inside the tab
  - Makes sense if the content contains only nested elements

Configuring tabs

- Manual tab configuration

```
<xsd:annotation>
 <xsd:appinfo>
 …
  <tabs>
    <tab element="…" collapse="false" name="MyTabName" />
    <tab element="…" />
    …
   </tabs>
 …
```

- Each tab element defines with which XML element the tab should start
- When using tabs, it is a good idea to start with the first element as first tab, otherwise OpenCms will do this for you!

Configuring tabs

- The `collapse` attribute should be set to `false` if the XML element defines a simple data type, not a nested structure. This prevents that the element name is hidden from the editor form
- The name attribute allows the definition of a tab name. This can be localized in a resource bundle by adding the following key to it:

```
label.${Name of XSD complex type}.${Tab name attribute value}
```

Configuring tabs

- Example of a tab configuration for a structured content of the type photo album:

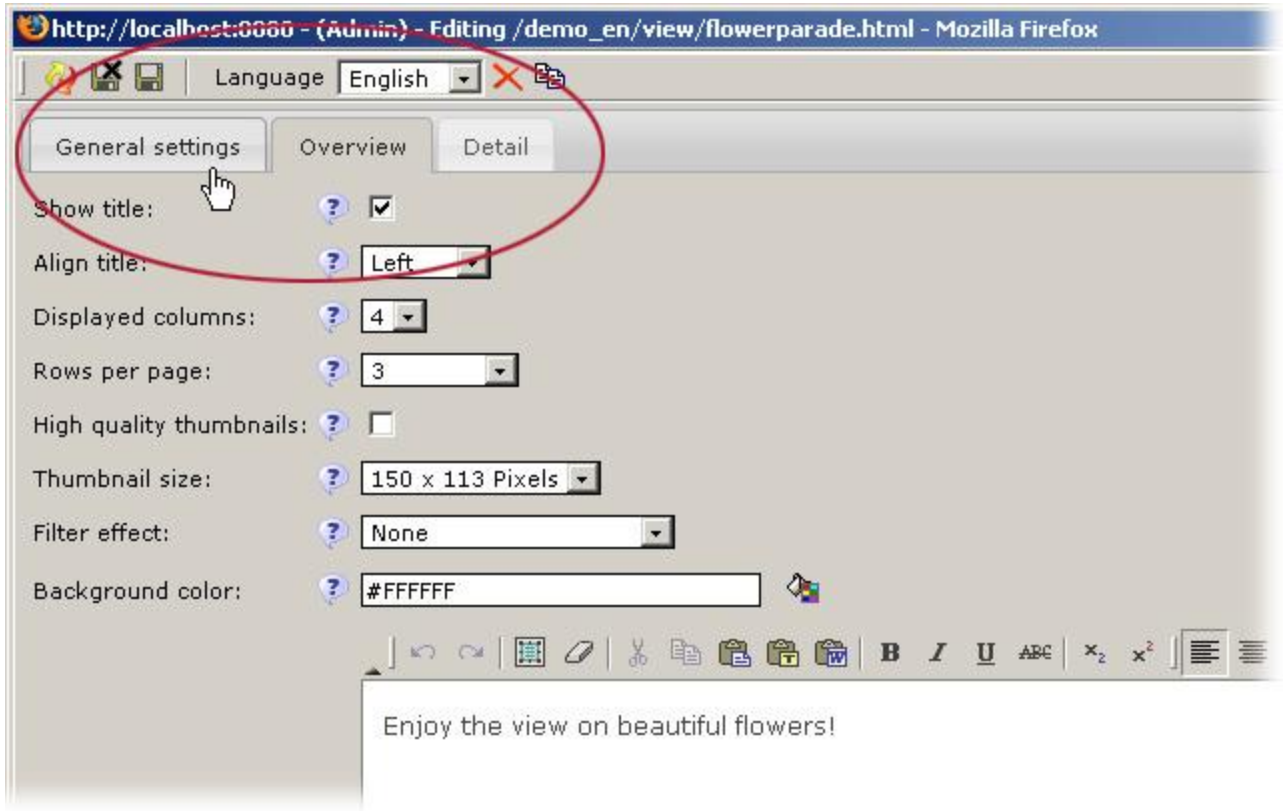
```
<xsd:complexType name="OpenCmsPhotoAlbumConfiguration">
   <xsd:sequence>
     <xsd:element name="Title" type="OpenCmsString" />
     <xsd:element name="VfsFolder" type="OpenCmsVfsFile" />
     …
     <xsd:element name="Thumbs" type="OpenCmsPhotoAlbumThumb" />
     <xsd:element name="Details" type="OpenCmsPhotoAlbumDetail" />
     …
<xsd:annotation>
   <xsd:appinfo>
   …
   <tabs>
     <tab element="Title" collapse="false" name="TabGeneral" />
     <tab element="Thumbs" />
     <tab element="Details" />
   </tabs>
 …
```

Configuring tabs

- Label key for the first tab:

```
label.PhotoAlbumConfiguration.TabGeneral = General settings
```

- Result in the editor:



Alkacon

Thank you very much for your attention!

Andreas Zahner
Alkacon Software GmbH

http://www.alkacon.com
http://www.opencms.org