Alkacon Software GmbH

Technote

Alkacon OAMP Webform Module

Version: 1.4.1

Date: Friday, November 12, 2010

1 Table of Content

2   Abstract

This document describes the installation, configuration and usage of the Alkacon OpenCms Add-On Module Package Webform version 1.4.1. With the Webform module, it is possible to create highly configurable online input forms without knowledge of HTML.

Once created, configured and published, a Webform may be filled out by website visitors. The data entered may be sent to an email account and / or stored in the OpenCms database.

3   General purpose of the Alkacon OAMP Webform Module

The module extends a basic OpenCms installation with the capability to create highly configurable online input forms. It provides the following features:

- Create Webforms by mouse clicks and simple keyboard input. A Webform is a structured XML content which offers a comfortable user interface.

- Complete configuration of a Webform is done in one file.

- Input fields, their labels, default values and options are freely configurable.

- For each input field it is possible to pick from several different field types, like Text input, Text area, Checkbox, Radio buttons, Hidden fields, File upload fields and Email Field.

- Input fields have several configuration options, like defining a default value, control if it is mandatory, define a regular expression for validation and an error message to be shown in case validation was not successful.

- To avoid that spiders use your Webforms (e.g. in order to spread spam to forums or guest books) a CAPTCHA field may be configured. This is an image containing distorted text that has to be entered in a text input to verify that a human is filling out the form.

- The collected data of a submission may be sent to a configurable email account and / or stored in the OpenCms database in separate exclusive tables, which can later be accessed by a report frontend or using the API.

4   Installation

Note: To use the Alkacon OAMP Webform module version 1.3.0, you need at least OpenCms version 7.0.3. The module is not compatible with older OpenCms versions.

Note: To use the Alkacon OAMP Webform module version 1.3.1 or later, you need at least OpenCms version 7.5.0. The module is not compatible with older OpenCms versions. Also version 1.3.0 of the OAMP Webform module is not compatible with OpenCms 7.5.0.

Note: The OpenCms module `org.opencms.frontend.templateone.form` which is the predecessor and a "lighter" version of this module (no support for database persistence) should be uninstalled before installation of this module. Old email forms that have been created with this module might still work except for the captcha image verification. The used captcha library JCaptcha of this module is a newer version and it is required that the older captcha libraries are deleted from the OpenCms web application lib folder.

Step by step installation procedure:

1. Go to the OpenCms Administration view

2. Click "Module Management" and select either "Import Module from Server" if the module was placed in the **WEB-INF/packages/modules/** folder of your OpenCms installation, or select "Import Module with HTTP" to upload the module from your local file system

3. Select the Alkacon OAMP Webform module zip file **com.alkacon.opencms.formgenerator_1.x.x.zip** to import

4. Check if the jar file **com.alkacon.opencms.formgenerator.jar** has been deployed in the **WEB-INF/lib/** folder after installation

5. In case you find the libraries
   ehcache-1.0.jar
   jcaptcha-all-1.0-RC2.0.1.jar
   in the **WEB-INF/lib/** folder please delete them

6. Restart your servlet container afterwards

4.1 Configuration

Configuration is necessary to define the database in which the Webform submissions will optionally be stored in. Furthermore the location in the real file system or OpenCms virtual file system (VFS) has to be configured where file uploads should be stored in case a Webform is set up that stores it's data in the database and has a file upload field. Also the database provider has to be configured.

4.1.1 Module Parameters

Important: it is possible that the settings to configure here are chosen wrong. Please have a look at the OpenCms log file (<tomcat>/webapps/<webappname>/WEB-INF/logs/opencms.log) for any errors or warnings if something is not working as expected.

Log in to the OpenCms Workplace.

1. Switch to the Administration View.

2. Go to the Module Management page.

3. In the list of the modules click on the module name of the module **com.alkacon.opencms.formgenerator**.

4. Click on Module Parameters. Adapt the settings for the module parameter **db-pool** and **uploadfolder**. If you want to store the uploaded files in the OpenCms VFS, check also the parameters **uploadvfs** and **uploadproject**. All other parameters are optional to change.
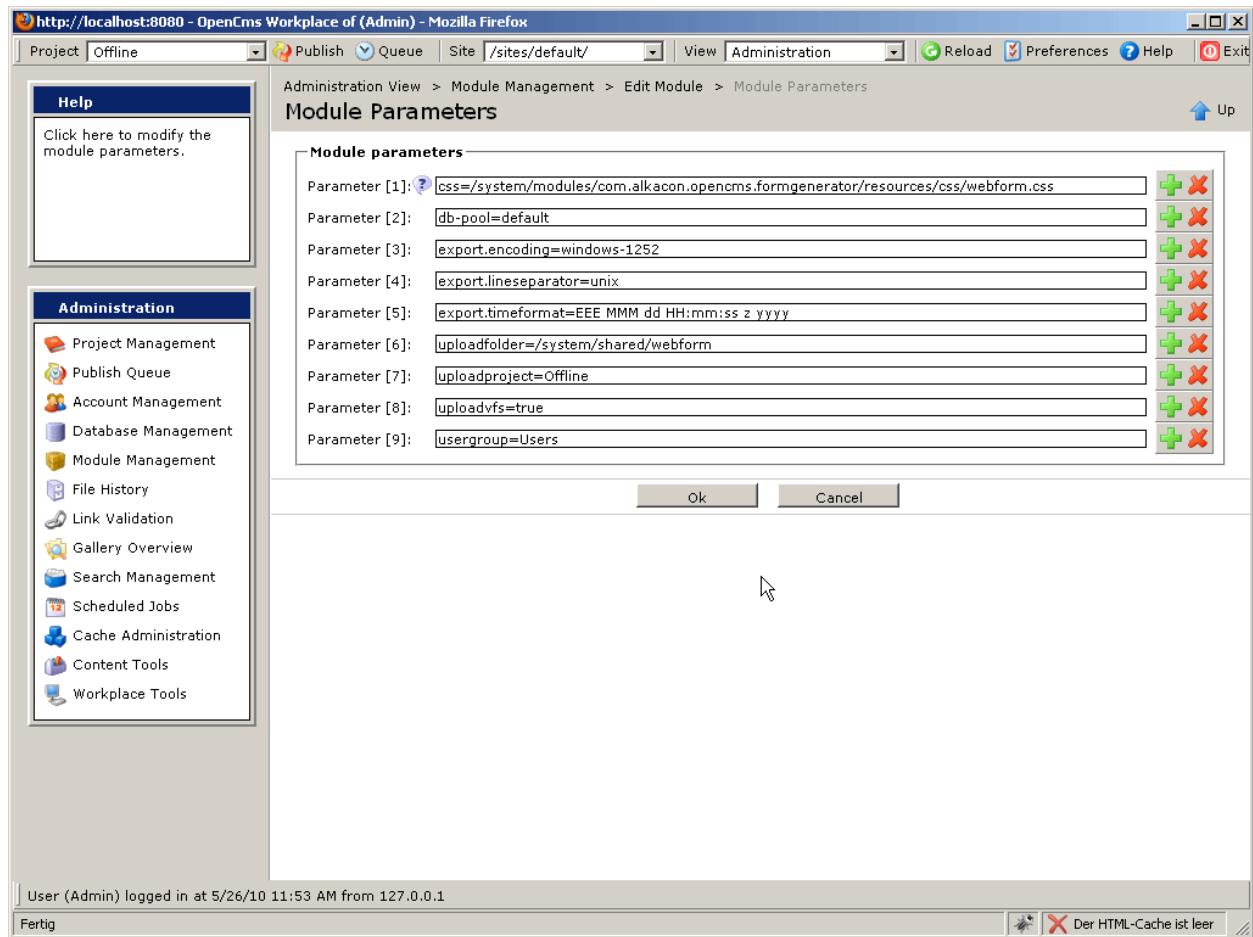
Figure 1: Editing Module Parameters

It is possible to use other property files for the module than the default files in com.alkacon.opencms.formgenerator/classes/com/alkacon/opencms/formgenerator/workplace. properties: you have to set the module parameter **message** to the bundle to the other property files. If the other files are named workplace2*.properties, you have to set the **message** parameter to "/com/alkacon/opencms/formgenerator/workplace2". It is also possible to use specific property files per form. Therefore see the description of the field "Property file" in chapter 5.2.

4.1.1.1 css

With this parameter, the CSS style sheet to use can be specified. If no value is defined, the default CSS file of the module **/system/modules/com.alkacon.opencms.formgenerator/resources/css/webform.css** is used and included.

Note: to generate valid XHTML output, set the value of this module parameter to "false". In this case, you have to include the CSS manually in your used template head section. See chapter 8 for details.

4.1.1.2 db-pool

The **db-pool** parameter value references a database pool configured in **<tomcat-home>/webapps/<webappname>/WEB-INF/config/opencms.properties**. The value "**default**" should be OK for most applications. If you need to collect your Webform data in a dedicated database you may configure another pool in the **opencms.properties** and refer to

it in this module parameter.

4.1.1.3 db-provider, index-tablespace

At the moment supported database providers are only MySQL and Oracle, being MySQL the default. If you are using Oracle as database provider you have to set following module parameters:

db-provider=oracle and **index-tablespace=<your tablespace>**, here the default is '**users**'. After changing these parameters you will have to restart your servlet container.

4.1.1.4 export.timeformat

This parameter allows defining a date and time format for the "Creation date" column in the exported CSV file. The syntax is explained here:
http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html.

Example: for an Excel – compatible format like "03.04.2009 11:06:59" one could configure "dd.MM.yyyy hh:mm:ss".

4.1.1.5 export.encoding

This parameter defines the encoding of the exported CSV file. E.g. for Excel 2003- compatibility choose "windows-1252". Else "utf-8" is recommended. If this parameter is omitted the OpenCms – configured default encoding is used (utf-8).

This setting is helpful if the exported file is directly opened with an external program that is hard wired to a special encoding.

4.1.1.6 export.lineseparator

A setting "windows" will cause "\r\n" in exported CSV files for line breaks. A setting "unix" or "excel" (Microsoft Excel displays a square for carriage return '\r') will cause '\n' in exported CSV files for line breaks.

4.1.1.7 font-prefix

This optional parameter allows the definition of character set prefixes which will be used for the generation of captcha images (see also 5.2.4). The prefixes should be separated by a pipe (|). If this parameter is not set the following default prefixes are used:

Arial|Courier|Monospaced|SansSerif|Serif

The filtering of the character sets can be disabled by setting the value to "false". In that case all available system character sets are used for the captcha generation.

Example: **font-prefix=Arial|Courier**

In this example only system character sets which start with "Arial" or "Courier" will be used for the generation of the captcha images.

Use this parameter to prevent the generation of "unreadable" symbols in the generated captcha images.

4.1.1.8 uploadfolder

The parameter **uploadfolder** is needed whenever you set up a Webform that has to store submissions in the database and contains a file upload field. These uploads are not stored in the database but in a folder that has to be set here. Please select a valid directory and check

the log file when setting up and testing such a form.

The files can either be stored in the real server file system (RFS, default setting) or in the OpenCms virtual file system (VFS).

To store files in the OpenCms VFS instead of the servers real file system, specify an existing unlocked and published folder here and also set the values of the module parameters `uploadproject` and `uploadvfs`.
Note: the absolute path to the folder has to be entered. For the RFS, this could be `c:\tmp`. The VFS folder should be below the `/system/` path, e.g. `/system/shared/webform`.

4.1.1.9 uploadproject

If uploading a file of an upload field to the OpenCms VFS, an existing offline project name has to be specified here. The folder defined in the parameter `uploadfolder` has to be part of this project.

4.1.1.10 uploadvfs

Set the value of this module parameter to "true" to enable the storage of uploaded files in the OpenCms virtual file system (VFS).

4.1.1.11 usergroup

This controls the visibility of the workplace tool on the top level of the workplace Administration. On the Database Management the tool is visible for Administrators only. On the top level it will be visible for every user of the group that is entered here.

By default every user of the group "Users" may view the workplace tool.

5 Module usage

After successful installation and configuration of the Webform module, it is ready to use. Webforms can be set up by creating a new resource of the type "`Alkacon Webform`".

5.1 Creating an Alkacon OpenCms Webform

To create a new Webform the "New" dialog in OpenCms has to be used:

1. Click "New" in the top bar of the OpenCms Workplace.

2. In the following dialog click "Structured content" and then "Continue".

3. Select "Alkacon Webform" and "Continue".

4. In the following dialog enter the name of the new file and set some properties if desired.

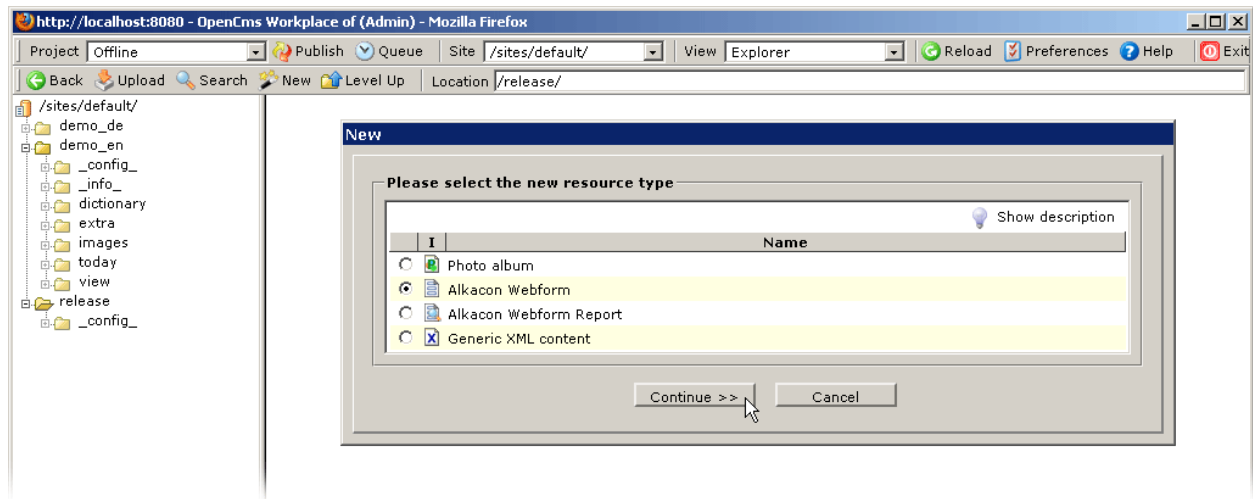→ A new file of type "alkacon-webform" has been created and is visible in the OpenCms Explorer.

Figure 2: The 2nd stage of the new dialog to create an Alkacon OpenCms Webform.

5.2 Editing an Alkacon OpenCms Webform

Webforms are edited by clicking their file symbol in the OpenCms explorer view with the right mouse button and choosing "Edit" from the context menu.

The XML content editor for Webforms appears. One can add as many labelled fields as desired to collect data. For a better overview, the Webform configuration options are grouped with tabs.
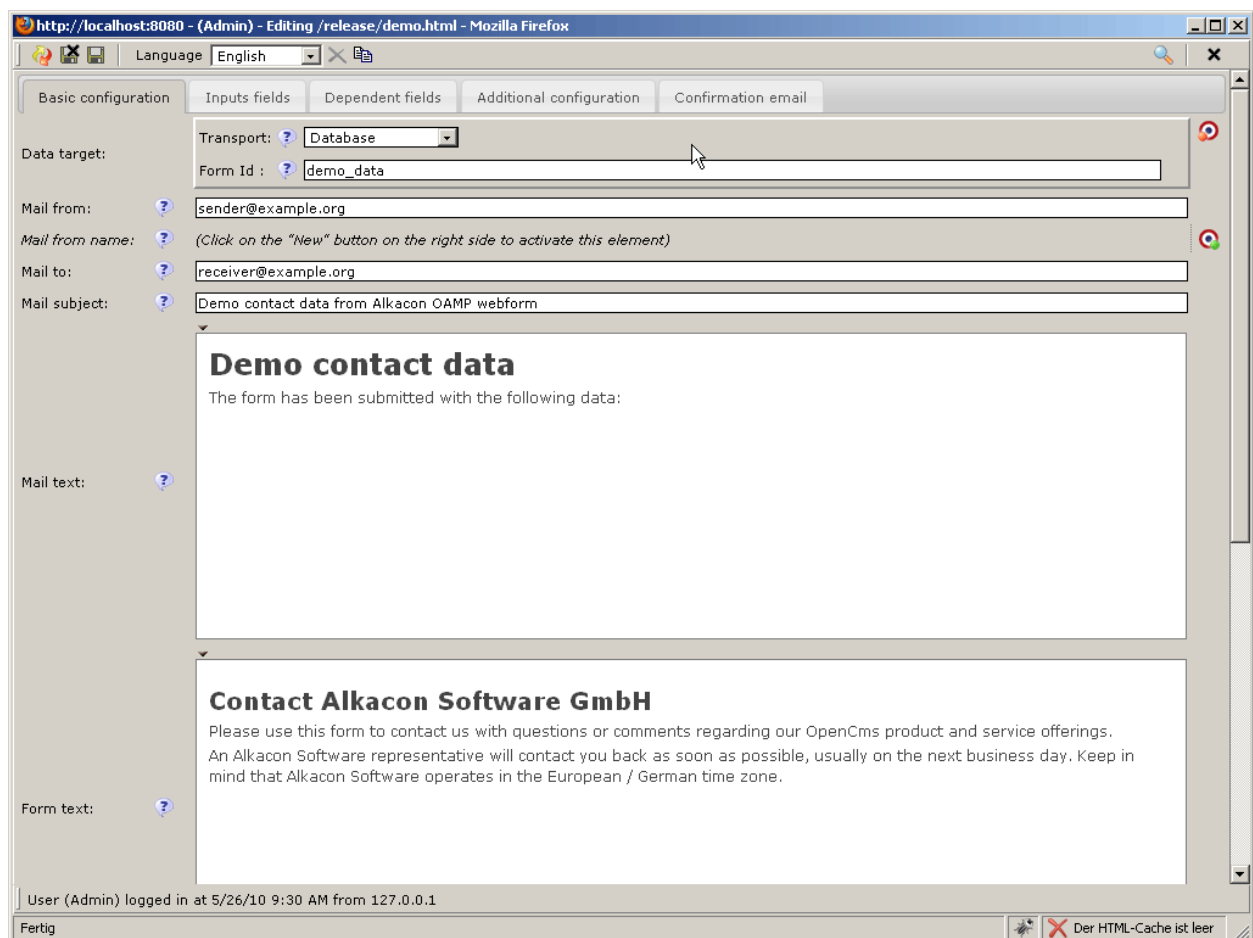

Figure 3: Alkacon OpenCms Webform editor.

5.2.1 Tab "Basic configuration"

The following fields are optional:

- o **Data target:** This nested box allows configuring how the data is processed. By default (if this box is not added with the plus button) the submitted data will be sent as an email to the configured "Mail to" address.

    - o **Transport**: Controls which mode of transportation is used. In case "Email & Database" or "Database" is used the 2nd field "Form Id" is required for database persistence.

    - o **Form Id**: In case submitted data is stored in the database, this ID will identify which form was submitted. Later on, if OpenCms workplace users want to download the data of a form, they see a "Download Data" link on the frontend page in an offline project which will produce an Excel CSV file with all data matching this form ID. With this solution it is possible to have several siblings of the same page in different language folders all sharing the same `Form Id` thus being exported all into a single CVS file.

There are **some mandatory** fields that have to be configured to make the resulting Webform work correctly:

- o **Mail from**: This has to be a valid email address that should be accepted from the mail server you use.[1]

- o **Mail from name**: The optional nice name that is displayed by email clients as sender of the email.

- o **Mail to**: This has to be a valid email address the collected data will be sent to.

- o **Mail subject**: The subject of the email if email transport is configured.

- o **Mail text**: This is some additional text that is sent in the optionally generated emails. This most often can be left out or some lines (e.g.: "New contact request made on http://www.alkacon.com!") are sufficient as many mails in this form will be received.

- o **Form text**: This is the text that will appear on the web page showing the email form.

- o **Form middle text**: This optional text will appear below the form fields, but above the submit button.

- o **Form footer text**: This optional text that will appear below the submit button of the email form.

- o **Confirmation Text**: This text is shown if the website visitor successfully submitted the email form.

- o **Target URI**: This most often should be blank. If configured, after a successful submission the user's web browser will be redirected to the entered page.

- o **Form expiration**: This optional nested content allows defining an expiration date for the form. After the expiration date, the configured expiration text will be shown instead of the form.

---

[1] The mail server that is used is configured in the OpenCms configuration (file: `opencms-system.xml`). If sending of emails fails please contact your OpenCms system administrator to check the configuration.

- o **Expiration date**: Specifies the date when the form is expired.

- o **Expiration text**: This text is shown instead of the email form after expiration.

5.2.2 Tab "Input fields"

The form input fields are configured in this tab. There has to be at least one input field for a valid email form.

- o **Form field**: This nested box configures a single input field of the form.

  - o **Type**: Specifies the kind of input field that is displayed on the Webform. Have a look at chapter 5.3 for a detailed overview of available field types.

  - o **Label**: Should be entered. It will be displayed to the left of the input field and also in the generated email to identify the value entered by the website visitor. If you are saving the data to a database it might be helpful to define a locale independent field name, this can be done by entering a pipe character ('**|**') after the locale dependent label, followed by the locale independent name. For instance, instead of "`Your Email Address:`", if you use "`Your Email Address:|email`" then the data will be stored in the database as "`email`" and not as "`Your Email Address:`". You can also use this special labels for macros in the following fields:

    - ▪ **Mail Subject**

    - ▪ **Mail Text**

    - ▪ **Confirmation Text**

    - ▪ **Check Page Text**

    - ▪ **Confirmation Mail Subject**

    - ▪ **Confirmation Mail Text**

  - o **Mandatory**: If the mandatory flag is selected empty fields will not be accepted.

  - o **Default value**: This field may be used to preset values for text fields or text area fields. **Especially for checkbox fields, radio button fields or select box fields the default value has to be used to specify the items available for selection**: A special syntax: "`i1:Item 1|i2:Item2|i3:Item3`" allows to specify each item separated by pipe symbol ("**|**"), where you may differ between the internal name for the selection that will be used in the email (here: i1,i2,i3,…) and the visible selection on the webpage (here: Item 1, Item 2, Item 3). You can just specify one value for each item: "`Spaghetti|Coca Cola|Cerveza`" if you don't require internal names. The sole purpose is to allow different localized versions that have translated visible items for each language but use the same value in the generated Email (in case a computer program reads those emails). Preselections (default items) may be marked with an asterisk like Mr in "`Mrs|*Mr`".

  - o **Parameters:** This field can be used to add more configuration parameters for an input field in the format "key=value|key=value...". It is currently used for two special configurations.

    If the data target "Database" is selected, the field can be defined as unique:

- unique=true

If a user tries to submit the form with an already existing field value in the database, a validation error will be shown.

It is also used by the Alkacon OAMP Survey Module with the following options:

- showAverage: If the average of numerical values shall be shown, this key is to set to true

- averageDigits: The number of internal decimal places for the average

- averageText: The text on the average bar

- orientation: Per default the average bar is longer all the more average of the numerical values. This key is set to "desc", when the average bar shall be longer all the less average of the numerical values.

- **Validation**: This field allows specifying a regular expression that has to match the entered value. If the user enters something that does not match such an expression, submission will fail with an error message.

- **Error Message**: Allows configuring a custom error message if validation fails.

5.2.3 Tab "Dependent fields"

In this tab it is possible to define sets of dependent fields that appear if the user selects a specific value in a form input field.
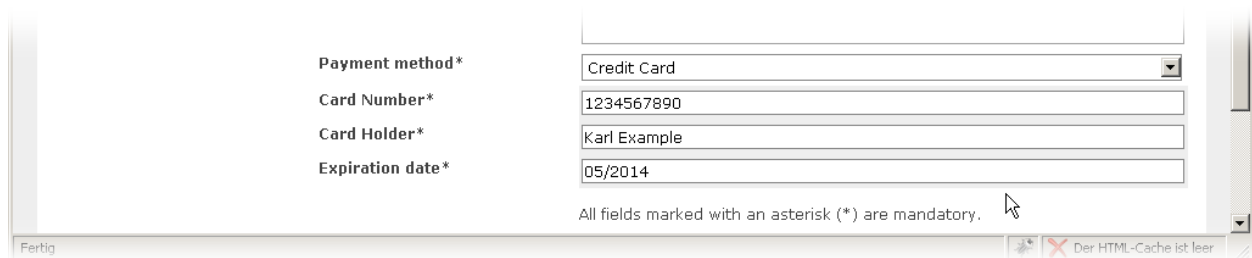
Figure 4: Dependent fields frontend view

A possible use case is the selection of a payment method. The user can select either "Credit Card" or "Bank Account". Depending on the selection, different dependent fields are shown.

Note: This feature is available for the field types "Radio button" and "Select box".

5.2.3.1 Configuration of dependent fields

First, a field of type Radio button" or "Select box" has to be defined. After that, the Webform has to be saved once.
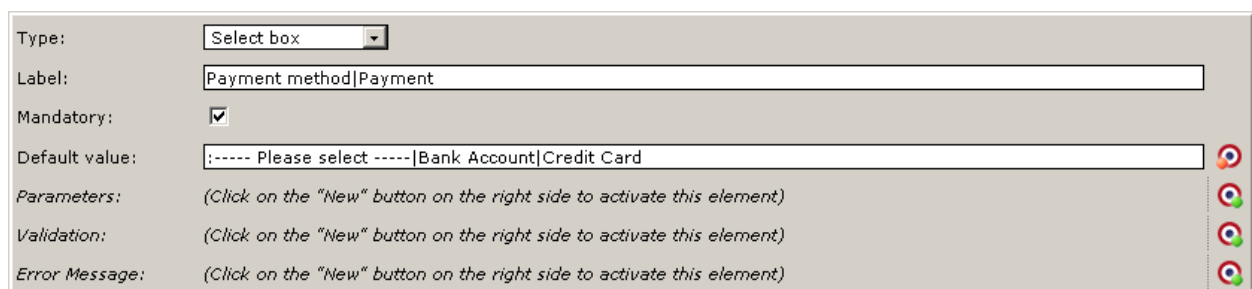
Figure 5: Main input field configuration

After that, the dependent field can be defined. For each dependent field, the following values

have to be set:

- o **Field name**: The field from which the fields should be dependent has to be selected. The label of the field is shown in the select box.

- o **Field value**: The necessary value of the field. The dependent fields are only shown if the main input field has the exact value that is entered here.

- o **Dependent field**: Configuration of one or more input fields that are shown. For a detailed description of the field configuration, have a look at section 5.2.2.

Figure 6: Editing dependent fields

Note: The mapping of the dependency is based on the main fields' database label. If this is changed after dependent fields were configured, the mapping of the dependent fields will be lost.

5.2.4   Tab "Additional configuration"

- o **Captcha field**: This is a powerful way to avoid robots / spiders [2] successfully send you emails by showing an image and a text field below where the user has to type in the same characters as seen in the image.

- o **Advanced configuration**: This allows some advanced configurations. These are:

  - o **Mail type**: The type of an email can be a text mail or a HTML formatted mail.

---

[2] Computer programs that want to send you ads / spam emails.

- o **Mail CC**: The mail CC recipient(s), use a semicolon to separate the recipients.
- o **Mail BCC**: The mail BCC recipient(s), use a semicolon to separate the recipients.
- o **Mail CSS**: Allows customizing the CSS that is used to format the emails sent by the Webform. If not activated, a default style sheet will be used.
- o **Enable form check page**: If enabled, an additional check page is shown before finally submitting the form data.
- o **Check page text**: The text displayed above the values on the check page.
- o **Enable mandatory marks**: If enabled, the mandatory marks and the mandatory notes are shown in the form.
- o **Enable reset button**: If enabled, the reset button will be shown on the form page.
- o **Form attributes**: Optional node attributes around the form, e.g. to define a CSS class name with class="myformclass".
- o **Field attributes**: Optional node attributes for each input field, e.g. to define a CSS class attribute with class="myinputclass".
- o **Dynamic fields class**: The class name for the dynamic fields value resolver.
- o **Style**: A style sheet class suffix specific for this form. A setting "orange" would result in style sheet class attributes like "webform_label_orange" in the generated form HTML.
- o **Property file**: With this setting a property file per form can be used. The property files are to create analogue to the com.alkcon.opencms.formgenerator/classes/com/alkacon/opencms/formgenerator/workplace*.properties files. They property files per form can be done in any module. The field has to be set with the bundle name, for example /com/alkacon/opencms/formgenerator/workplace2, if the property files are named workplace2*.properties.
- o **Action class**: The class name for the action which is executed after the form was saved in the database/the mail was sent. This class has to implement the interface **com.alkacon.opencms.formgenerator.I_CmsWebformActionHandler**.
- o **Refresh session**: To avoid that the visitors' session expires during filling out the form, an automatic refresh can be configured. The entered value is the interval in seconds which is used to refresh the session.
- o **Intermediate text**: These texts can be defined to appear below the selected input fields of the form. The following fields have to be defined to create an intermediate text:
  - o **Field name**: The label of the input field for which the text is shown.
  - o **Text column**: This controls in which column the text is shown. The left column means the column for the input field labels; the right column holds the fields. When selecting "Both", the text is shown in both columns.
  - o **Text**: The text entered here is shown below the field.

5.2.5 Tab "Confirmation email"

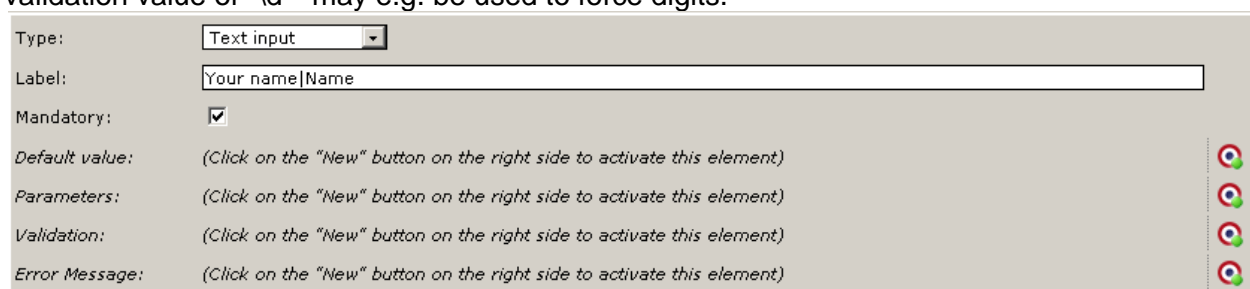In this tab, a confirmation email that will be sent to the person that submitted the Webform can be configured.

- o **Enable confirmation mail**: Enables sending a confirmation email to the user.

- o **Selectable confirmation**: The submission of the confirmation mail is selectable by the user by automatically adding a checkbox to the input form.

- o **Label confirmation field**: The label text for the automatically generated confirmation mail checkbox. If not provided, a default text will be used.

- o **Email Input field**: Select the label of the form input field (Form field [x]) that holds the email address the confirmation email should be sent to. This input field has to be of type "Email field".

- o **Confirmation mail from**: The optional confirmation mail sender address. If this field is empty, then the mail address in the field "Mail from" is used.

- o **Confirmation mail from name**: The optional nice name that is displayed by email clients as sender of the confirmation mail.

- o **Confirmation mail subject**: The subject of the confirmation email.

- o **Confirmation mail text**: The text of the confirmation email, the macros `%(date)` for the current date and `%(formdata)` as placeholder for the submitted form values can be used.

5.3 Webform field types

Many different types of fields for Webforms are available. They differ in functionality and in the resulting UI controls shown on the frontend of the Webform.

5.3.1 Text input

A common single line input field. This is the best choice for short data to collect like "City". A validation value of "\d*" may e.g. be used to force digits.

Figure 7: Definition of a text input field with database label

5.3.2 Text area

A text area offers more lines for more textual input. Most often it is used as "message" field.

5.3.3 Checkbox

This type will show up as a group of checkboxes on the frontend webpage. Each `value:displayname` pair like "`value1:Lorem Ipsum`" in the default value configuration will result in a single checkbox.

Figure 8: Definition of a checkbox group

5.3.4   Confirmation link

This field displays a checkbox and a link at the right side of it. It is very special. It may be used to let the website visitors confirm that they have read the "privacy statements" or "accept the license" where the link will lead to a more detailed explanation.

The link has to be site – relative like "/de/index.html". The link text is specified first, a separation colon follows and then the link has to be written in the "Default value" box: "Linktext:/de/privacy/index.html".

5.3.5   Radio buttons

Will show a group of selectable items, where only one item may be selected. Default value can be formatted as seen in 5.3.3.

Figure 9: Definition of radio buttons

Note: To show the radio buttons in a single row, enter the macro `%(row)` before the first item. The value should look like this: "%(row)Mrs|*Mr".

5.3.6   Select box

This type displays a select box with several options. This widget has the same function as radio buttons: select a single item out of a limited amount of items. But it will use less space if many items have to be offered. Most often it is good to use radio buttons for only two or three alternatives and select boxes for more items. Default value can be formatted as seen in 5.3.3.

Figure 10: Definition of a select box with 6 entries

5.3.7   Hidden Field

Hidden fields may be used to let a certain value be in the submission but not be visible or editable by the website visitors. This value (entered in "Default value") will be transmitted in the email and / or into the database. This might be used to differ between several forms that have all other fields equal. One could see from which form the submission was made even if all other submitted value names are the same.

Some spiders might crawl the web to find forms to fill out in order to enter their spam messages to forums or guest books and fill out even the hidden fields. As humans cannot access these fields a change of value of the hidden field is a proof that a program has submitted the form.

5.3.8   File Upload

File upload fields offer to select a file on the local hard drive which will be uploaded to the OpenCms server. A speciality is the possibility to use the validation field for configuration of the maximum upload size in the form "`<xkb`" where "`<`" and "`kb`" are fixed and `x` is an integer value for the amount of kilo bytes to allow for uploading.

Figure 11: Definition of a file upload field with maximum upload size limit

In case of email transport the upload will be attached to the email. If the submissions are stored to the database the uploaded temporary files will be copied according to the module parameter "uploadfolder" (and "uploadproject", "uploadvfs"). Their full path will be stored in the database.

5.3.9   Email field

Email fields are a comfortable way of validating the input as a valid email address. They are nothing more than a Text input with an internal additional regular expression.

5.3.10 Empty field

Empty fields may be used to have additional space between form rows. Also their "Default value" will appear in the column together with the other input field: This can be used to show input field group – headlines. No data is produced / submitted by empty fields.

5.3.11 Dynamic field

This field type works in conjunction with the advanced configuration option "Dynamic fields class", here you have to enter a class implementing the **com.alkacon.opencms.formgenerator.I_CmsDynamicFieldResolver** interface for generating the values for fields of this type.

5.3.12 Table field

This field type allows to create fields composed of several subfields in a "n times m matrix".

To define the columns and the rows you use the "Default value" field, by giving a comma separated list of column names followed by a 'pipe' symbol and a comma separated list of row
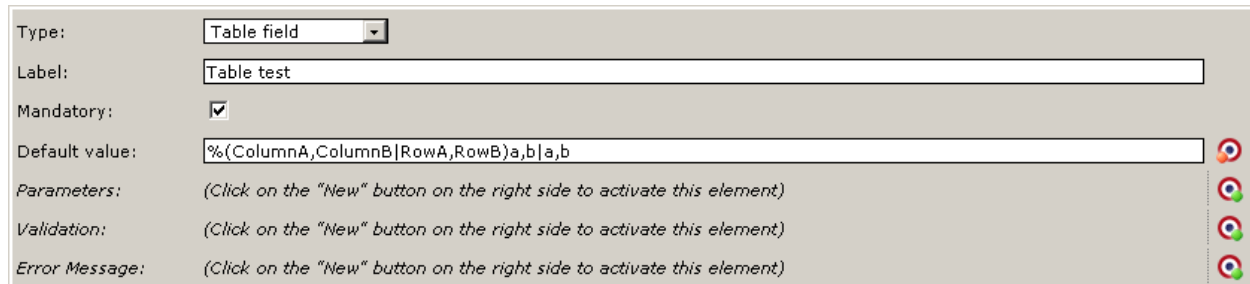
names, like:

```
colA,colB|rowA,rowB
```

In that case the same names will be used for displaying and in the database. If you want different names you can use the macro notation '%(' and ')' for the display names, like:

```
%(ColumnA,ColumnB|RowA,RowB)a,b|a,b
```
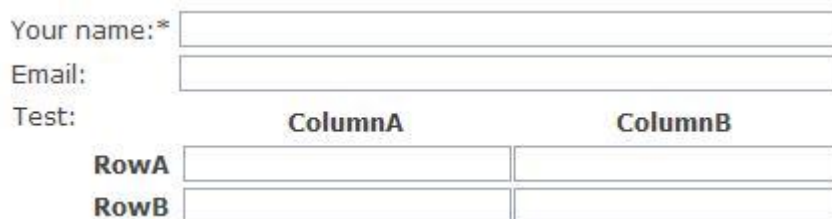
Additionally you have to know that the values are stored in the database in the form **colName_rowName**. And you can use also macros in the same form.

Type:	Table field
Label:	Table test
Mandatory:	☑
Default value:	%(ColumnA,ColumnB|RowA,RowB)a,b|a,b
Parameters:	(Click on the "New" button on the right side to activate this element)
Validation:	(Click on the "New" button on the right side to activate this element)
Error Message:	(Click on the "New" button on the right side to activate this element)

Figure 12: Definition of a table field with two columns and two rows

The resulting table field is displayed in the frontend like this:

	ColumnA	ColumnB
Your name:*		
Email:		
Test:		
RowA		
RowB		

Figure 13: Frontend view of a table input field

5.3.13 Password field

Password fields may be used instead of text input fields to hide the field contents on the form input and confirmation page. However, the value will be sent in a readable format in the emails.

5.3.14 New page

This special field type allows the creation of more than one form input page. All fields after this field type are shown on a new page. It is possible to browse between the different pages with a "previous" and a "next" button.

5.3.15 Display field

This field type works in conjunction with the advanced configuration option "Dynamic fields class", here you have to enter a class implementing the **com.alkacon.opencms.formgenerator.I_CmsDynamicFieldResolver** interface for generating the values of these type of fields. In contrary to the dynamic field the display field is initialized once at the call of the form. The returned value is displayed in an enabled text field.

5.3.16 Hidden display

This field type works in conjunction with the advanced configuration option "Dynamic fields class", here you have to enter a class implementing the

`com.alkacon.opencms.formgenerator.I_CmsDynamicFieldResolver`
interface for generating the values of these type of fields. The value is only used in a hidden field.

6 The frontend view

Once created an email Webform should seamlessly integrate with your web page.

Figure 14: Demo frontend view of a Webform

When looking at a web form in the offline project of OpenCms[3] a "Download data" button is

---

[3] You view pages in the offline mode when you are logged in to the OpenCms Workplace and have the project slider set to „offline".

offered that will lead to a page where it is possible to export a Microsoft Excel CSV file with the exported data of the submissions of this form. It is also possible to export the data only in a selected time range.
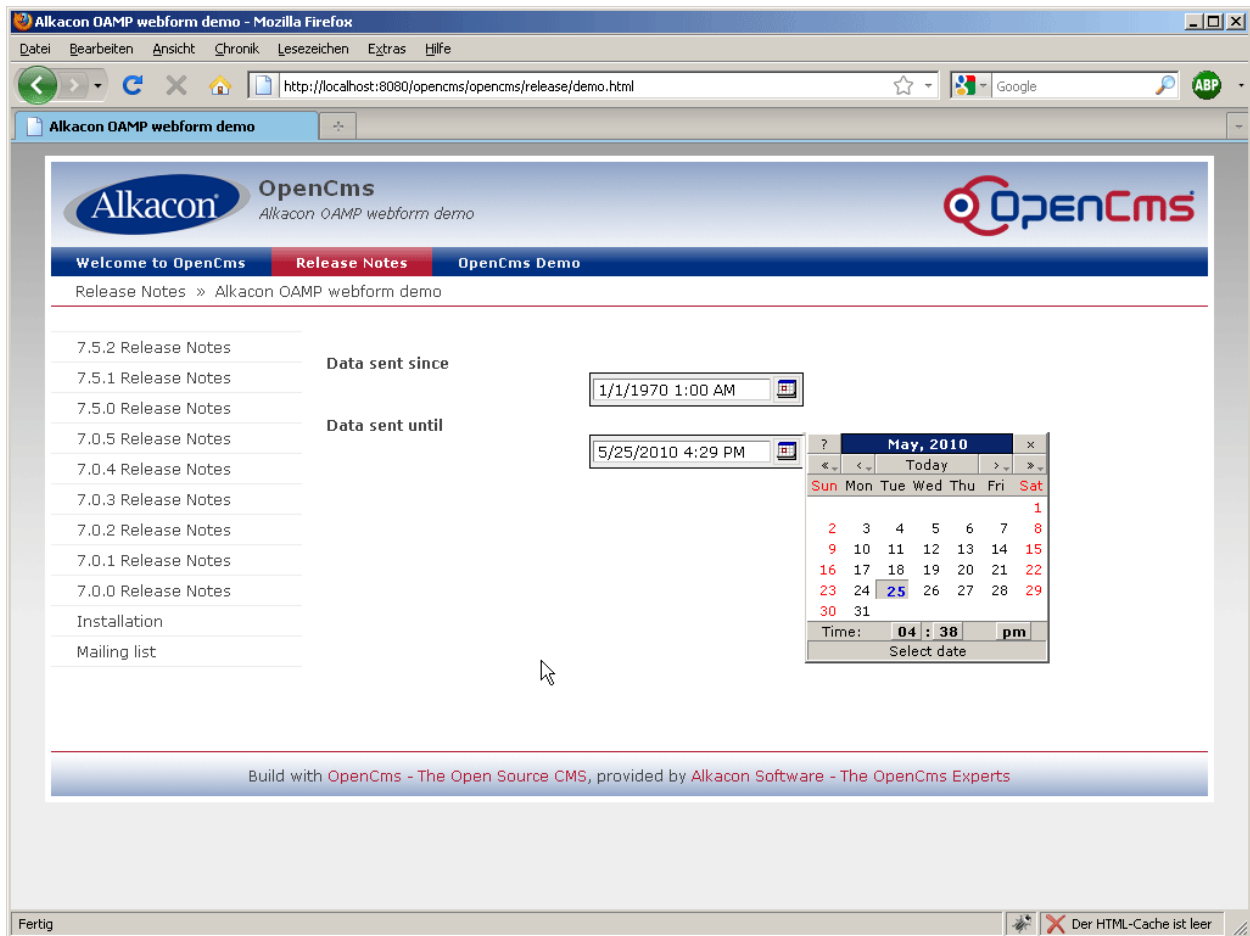

Figure 15: The data download page.

6.1 Customization of the frontend HTML

In case the layout is not OK you can configure the module parameter "`message`" of the OpenCms module `com.alkacon.opencms.formgenerator` to point at a properties file that contains the HTML snippets used for showing the webform in variations. The default file with these HTML snippets is located in the VFS:
`/system/modules/com.alkacon.opencms.formgenerator/classes/com/alkacon/opencms/formgenerator/workplace.properties`. If a properties file with these HTML snippets and other localizations has been changed it is required to publish it and re-initialize the workplace in the OpenCms Administration.

6.2 Customization of the frontend CSS

The HTML generated for Webforms contains CSS class attributes that may be attached to style sheet definitions in the CSS file that is linked by the template that is used.

A demo CSS file is located at /system/modules/com.alkacon.opencms.formgenerator/resources/css/webform.css. To begin writing your own style sheet definitions you may copy the content of that file into your CSS file

that is linked to your custom template. You can also use the module parameter "css" to use a customized style sheet instead of the default one.

Different Webforms may use different style sheet definitions. This is done by setting a value in the Webform configuration field "Style". This value is a style sheet class suffix.

Figure 16: Custom style sheet class suffix.

If e.g. the value "orange" is entered there the HTML will contain additional class attribute values like: "webform_field_orange". So in your CSS file you are free to define specific CSS properties for webform_field_<Style>, webform_label_<Style>, etc.

7   The report frontend view

It is possible to generate a report frontend view for Webforms that store their submitted data in the database.
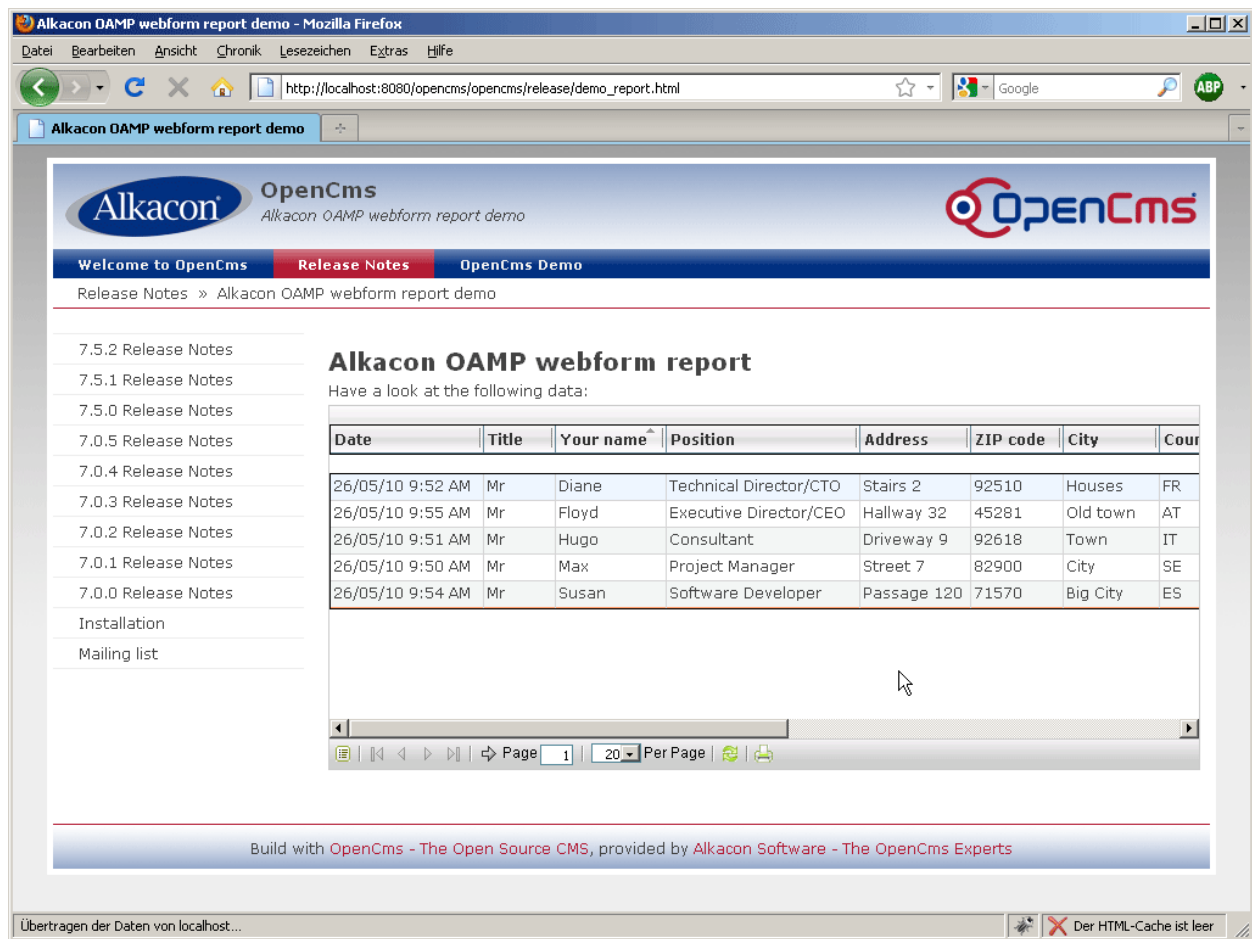
Figure 17: Report frontend view

7.1 Creating a report frontend view

To create a new Webform report, the "New" dialog in OpenCms has to be used:

1. Click "New" in the top bar of the OpenCms Workplace.

2. In the following dialog click "Structured content" and then "Continue".

3. Select "Alkacon Webform Report" and "Continue".

4. In the following dialog enter the name of the new file and set some properties if desired.

→ A new file of type "alkacon-webformreport" has been created and is visible in the OpenCms Explorer.

7.2 Editing a report frontend view

The report has to be configured using the OpenCms XML content editor. The fields to configure are grouped using tabs.

Figure 18: Editor view of a report output page

7.2.1   Tab "Report settings"

On this tab the necessary information to create a report output has to be configured. The following fields have to be entered:

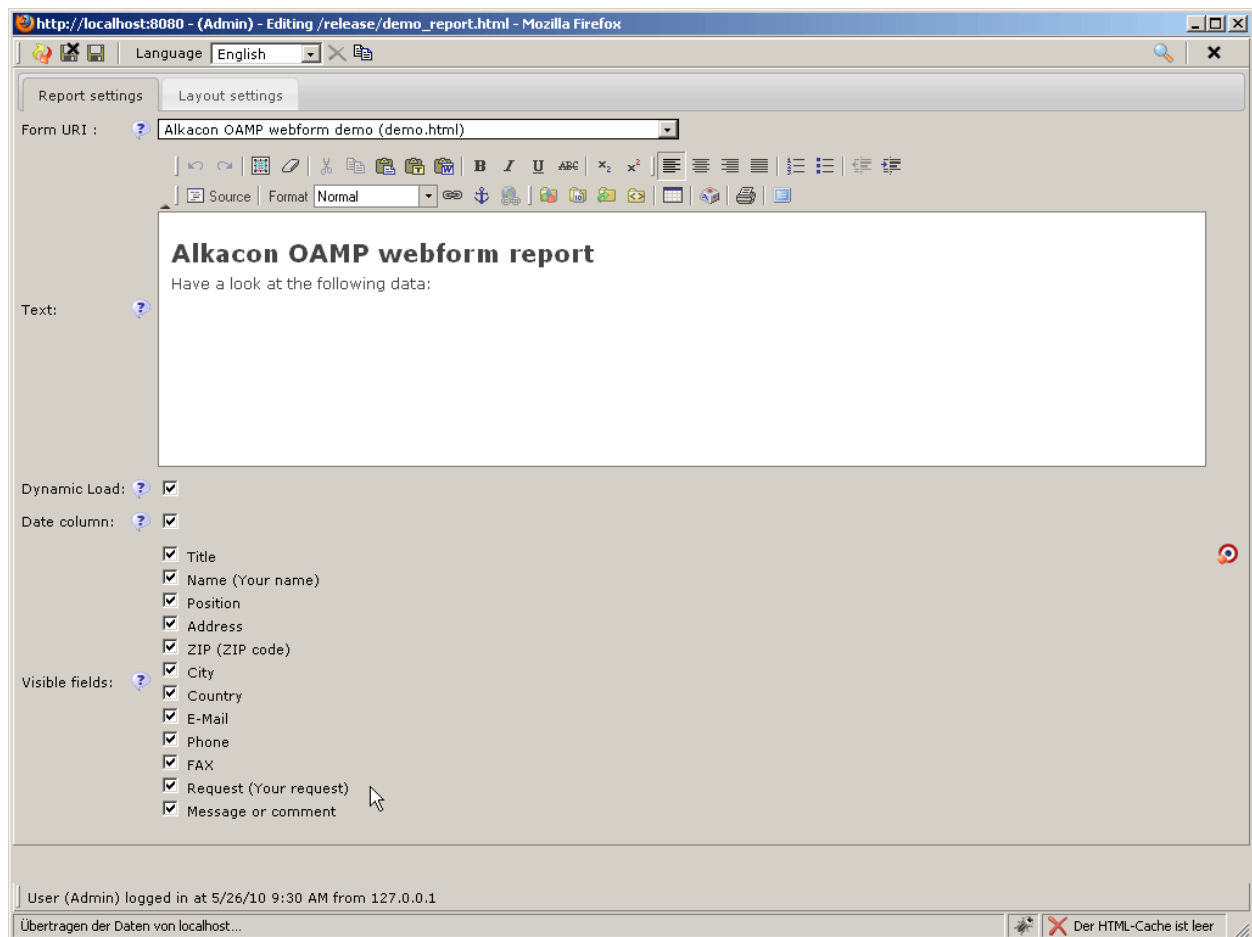- o **Form URI**: URI of the form from which the report should be shown. Only files of the type "alkacon-webform" are selectable.
  Note: Save the report file every time after selecting a different URI.

- o **Text**: The text entered here is shown above the report output.

- o **Dynamic Load**: If checked, the report data is dynamically loaded in the background to improve the browser performance.

- o **Date column**: Determines if the submission date column is shown in the report.

- o **Visible fields**: Selection of the visible fields.
  Note: dependent fields are always shown if the main field is selected.

7.2.2   Tab "Layout settings"

The settings on this tab configure the report layout.

- o **Show menu**: Enables the menu button for the report grid. Using the menu, the user can hide columns or switch to another skin of the grid.

- o **Show labels**: Show the field labels on the report page instead of the database labels.

- o **Grid height**: Sets the height of the report grid.

o **Grid skin**: The start skin of the grid. There are four different skins available: "Default" (green), "Mac" (grey), "Vista" (blue) and "Pink" (!).

o **Column width**: Sets the width of a single column of the report, the user may adjust this after loading the report.

8   Generating XHTML compliant frontend output

If the Webform should generate XHTML compliant output, the used CSS style sheets have to be included manually by the template that is used on the website.

Therefore, the Alkacon OAMP Webform Module has to be configured by setting the module parameter **css** to "false".

This prevents the automatic output of the links to the CSS files in the **<body>** part of the generated page. The template developer should do the following to format the HTML output correctly:

8.1 CSS for the Webform

The default CSS file used by the Webform is /system/modules/com.alkacon.opencms.formgenerator/resources/css/webform.css. A link to this file has to be placed manually in the template head JSP file:

```
<link rel="stylesheet" type="text/css"
href="<cms:link>/system/modules/com.alkacon.opencms.formgenerator/resources/css/webfor
m.css</cms:link>" />
```

A customized CSS file can be used instead of the default CSS to format the form view according to the template.

8.2 CSS for the report

The report output links to more than one CSS file because of the four different grid skins. The module contains a JSP that links to all the necessary style sheets. This JSP has to be included in the template head JSP file to make the report work:

```
<cms:include file="/system/modules/com.alkacon.opencms.formgenerator/elements/report_css.jsp" />
```

9   Administration view

To access the data stored by Webforms in the database, an own Administration icon "OAMP Webform" can be found in "Database Management" of the Administration view. There it is only accessible for users having the role "Database Manager".
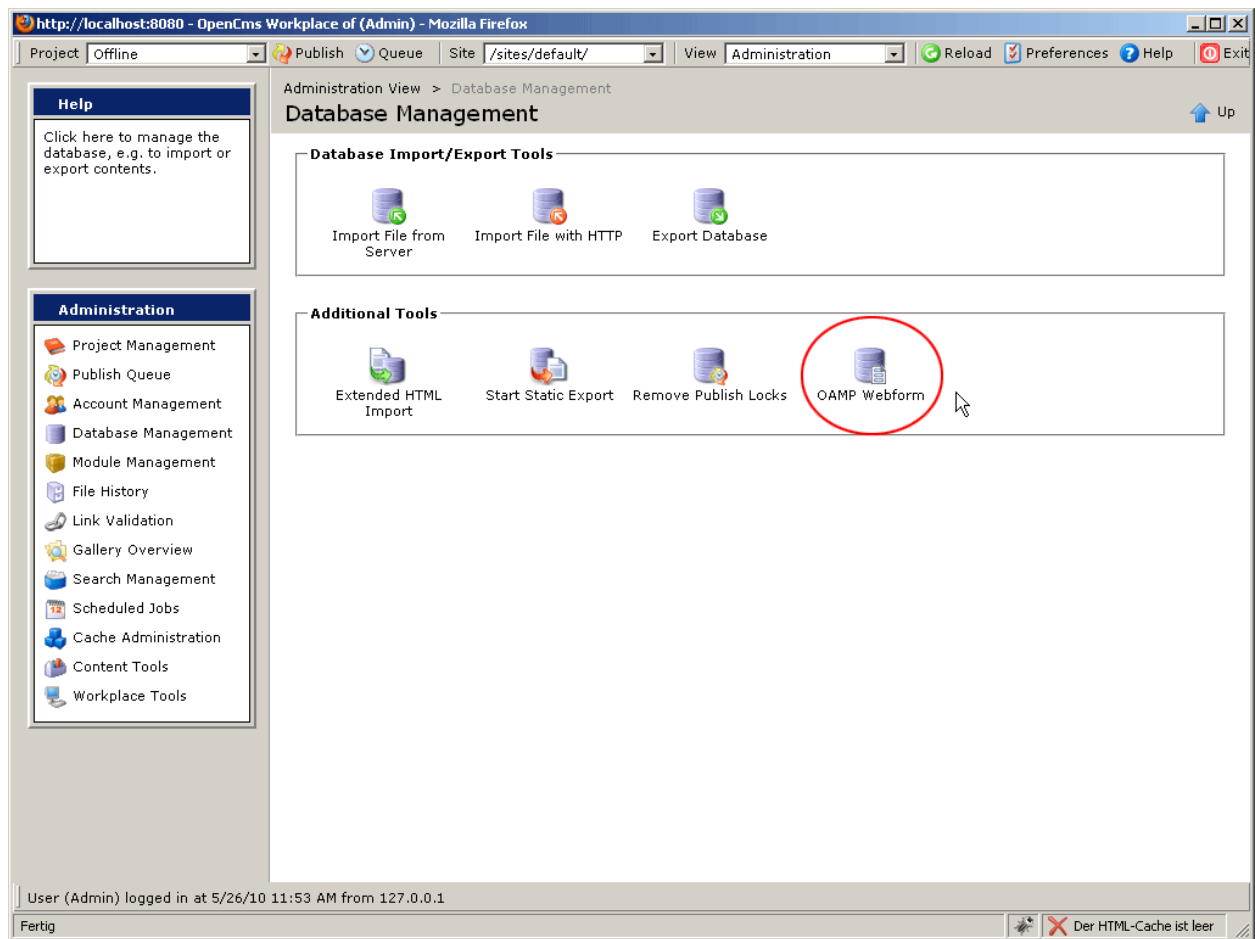
Figure 19: Database Management view with OAMP Webform

On the top level of the Administration it only shows up if the value of the module parameter "usergroup" defines a group the user currently logged in is a member of.

The first view (Figure 20) shows all existing OpenCms Webforms, which have the database transport activated. Only OpenCms Webforms which contain at least one submitted data entry are shown. The following fields are shown in the form overview:

- **View**: Shows the detail view of the selected OpenCms Webform.

- **Delete**: Deletes all entries of the OpenCms Webform. This option is only active for users having the role "Database Manager".

- **Name**: The **Form Id** of the OpenCms Webform, which is configured in the Data Target.

- **Count**: The number of submitted data entries of the OpenCms Webform with the configured Form Id.
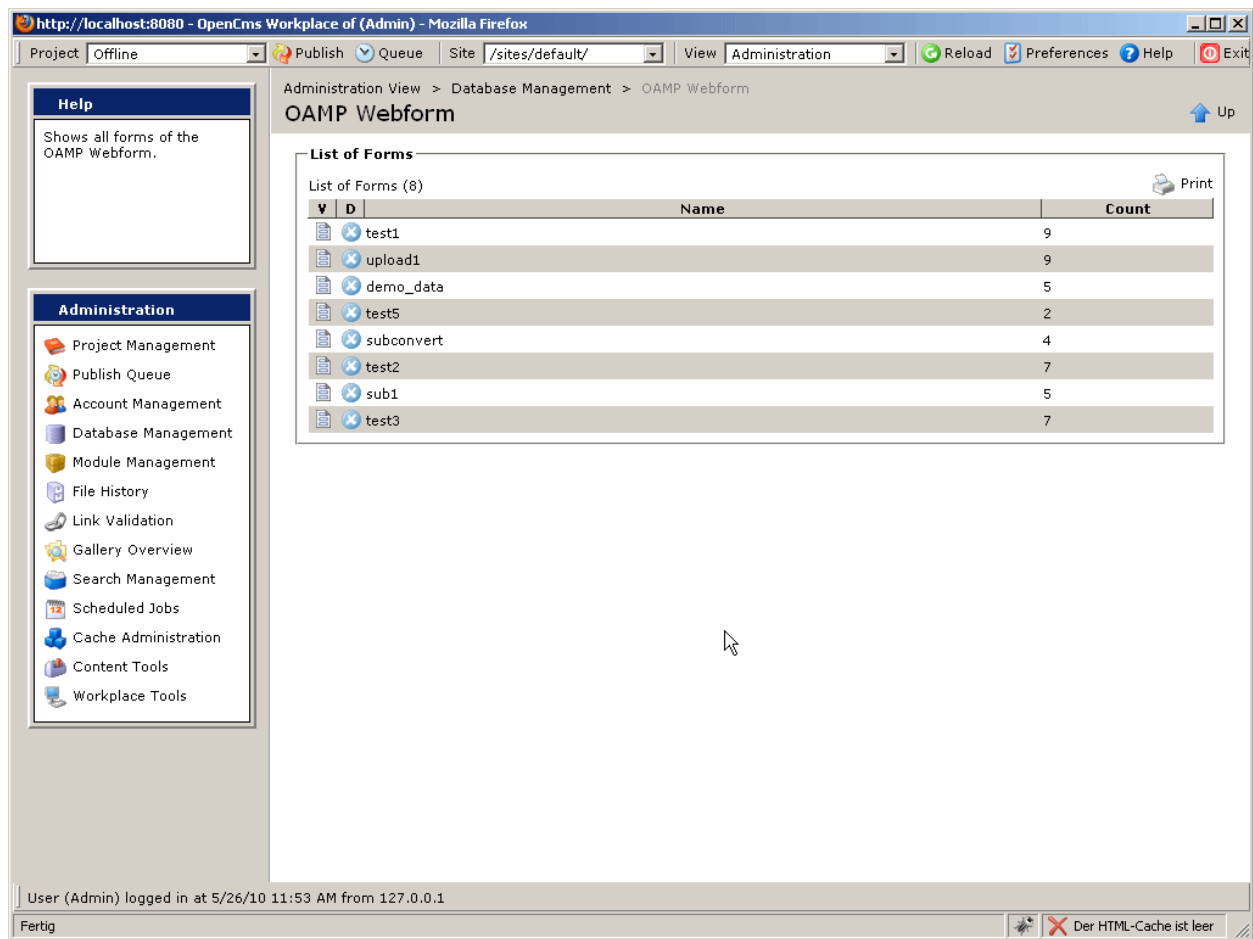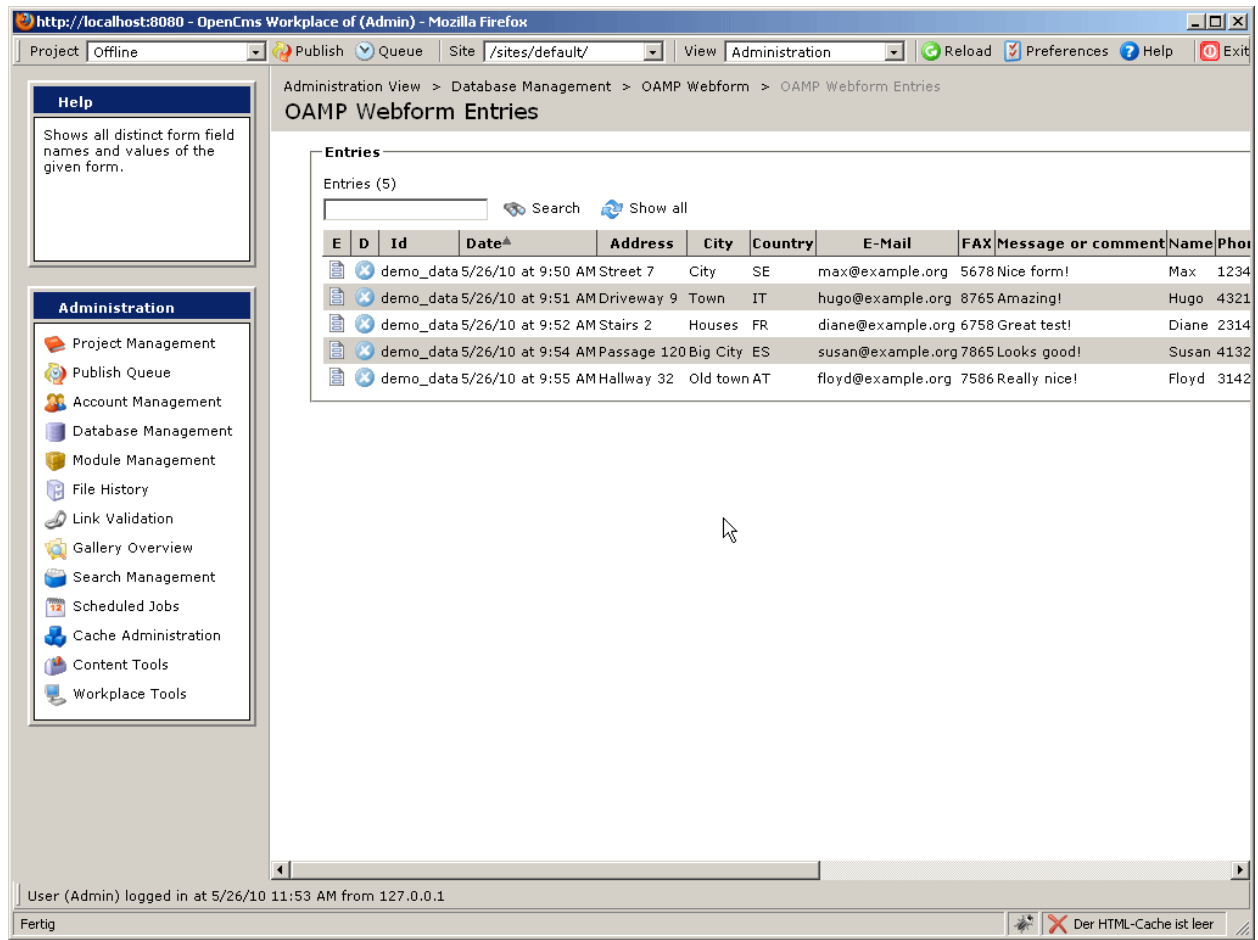
Figure 20: List of Forms in the Administration View

The detail view of a selected form contains static and dynamic columns. The dynamic columns are the configured fields in the selected form. The static columns are:

- **Edit**: Allows you to edit an entry (see Figure 22). This is only possible for users having the role "Database Manager" or users having the "write" permission on the related Webform file.

- **Delete**: Allows you to delete an entry. This is only possible for users having the role "Database Manager" or users having the "write" permission on the related Webform file.

- **Id**: The Form Id of the submitted data.

- **Date**: The date when the data was submitted.

- **Path**: The absolute VFS-path to the OpenCms Webform which was used to submit the data.

The dynamic field columns are shown between the **Date** and **Path** columns.

Figure 21: Entries of the form with the Form Id "demo_data"

9.1 Edit submitted data

The edit action in the detail view of a selected form allows you to edit the dynamically created columns.

This is only possible for users having the role "Database Manager" or users having the "write" permission on the related Webform file.

A text area-widget is used to edit when the value of a field contains a large text or line breaks.

By clicking "Ok" the modified values are stored back to the database.

Figure 22: Edit a submitted data

10 Using the module API

All classes used to generate and configure the webforms are part of the packages
com.alkacon.opencms.formgenerator,
com.alkacon.opencms.formgenerator.database,
com.alkacon.opencms.formgenerator.database.export and
com.alkacon.opencms.formgenerator.database.dialog.

10.1 com.alkacon.opencms.formgenerator

The package **com.alkacon.opencms.formgenerator** contains the implementation of the webform XML content data access along with the logic to display the webform and process the submission. The following classes are used:

- **I_CmsDynamicFieldResolver**: Interface for the dynamic field value generation.

- **I_CmsWebformActionHandler**: Interface to execute any action after the webform was sent.

- **I_CmsField**: Defines the methods required for form fields.

- **A_CmsField**: The abstract base class of field implementations.

- **CmsCaptchaEngine**: A JCaptcha implementation that allows configuring the captcha

deformation of images with the XML contents in the VFS folder
/system/workplace/admin/captcha/.

- **CmsCaptchaField**: A field implementation for generating captcha images and collection of the response phrase along with validation.

- **CmsCaptchaService**: A JCaptcha implementation to set up the captcha service that uses the **CmsCaptchaEngine**.

- **CmsCaptchaServiceCache**: A cache that avoids reading the captcha engine deformation configuration XML contents from database per request if possible.

- **CmsCaptchaSettings**: Bean for accessing captcha setting XML contents along with routines for reading from the XML contents and from requests.

- **CmsCheckboxField**: Form field implementation for checkboxes.

- **CmsDisplayField**: Represents a display field. This field is to fill dynamically using a class implementing the interface **I_CmsDynamicFieldResolver**.

- **CmsDynamicField**: Dynamic field implementation.

- **CmsEmailField**: Text field implementation with integrated email format validation.

- **CmsEmptyField**: Form field implementation for empty fields.

- **CmsFieldFactory**: A factory to create form field instances for a given type name.

- **CmsFieldItem**: Represents a single item of field implementations with items like **CmsCheckboxField**, **CmsRadioButtonField** and **CmsSelectionField**.

- **CmsFieldText**: Represents an additional text for an input field, with information in which column to show the text.

- **CmsFieldValue**: Represents a single input field value of a submitted form.

- **CmsFileUploadField**: Form field implementation for file upload fields.

- **CmsForm**: Bean for accessing webform XML contents along with routines for reading from the XML contents and from requests.

- **CmsFormContentUtil**: Utility class for accessing form content elements.

- **CmsFormHandler**: Contains the logic to process the webform (**CmsForm**): which page to show (confirmation, initial,…), where to send the data, error and validation processing,…

- **CmsFormReport**: Provides the methods to generate the form report output page.

- **CmsFormReportColumn**: Represents a form report column with all necessary data.

- **CmsHiddenDisplayField**: Represents a hidden display field. This field is to fill dynamically using a class implementing the interface **I_CmsDynamicFieldResolver**.

- **CmsHiddenField**: Form field implementation for hidden fields.

- **CmsHtmlToTextConverter**: Removes HTML tags and replaces them by line breaks or blanks. Used to remove tags if generating text emails.

- **CmsMaptcha**: Extends the text captcha for generating math challenges

- **CmsMaptchaEngine**: A captcha engine using a Maptcha factory to create mathematical captchas

- **CmsMaptchaFactory**: A factory for mathematical operations to display as maptcha on the form.

- **CmsMaptchaService**: Provides the facility to create and cache the maptchas.

- **CmsPagingField**: Represents a new page field.

- **CmsPasswordField**: Represents a password field.

- **CmsPrivacyField**: Form field implementation for privacy fields: a check box with a link to a configurable target.

- **CmsRadioButtonField**: Form field implementation for radio buttons.

- **CmsReportCheckFieldsWidget**: Provides a widget to check the fields to show on the form report page, for use on a widget dialog.

- **CmsSelectFieldWidget**: Special select widget that generates options from the configured form input fields.

- **CmsSelectionField**: Form field implementation for select boxes.

- **CmsSelectWidgetXmlcontentType**: A highly configurable select widget used in the XML content editor for Webforms to offer the different captcha preset XML contents found in the VFS. Also used in the report to select the form from which to display the report.

- **CmsTableField**: Represents a table with input fields.

- **CmsTextareaField**: Form field implementation for text areas.

- **CmsTextField**: Form field implementation for text fields.

- **CmsWebformDefaultActionHandler**: default action class which is executed after the webform was sent.

- **Messages**: convenience class to access the localized messages of the webform package.

10.2 com.alkacon.opencms.formgenerator.database

This package contains the database access layer for storing and reading submissions of webforms. The following classes are used:

- **CmsFileUtil**: Utility class for RFS file access with support for logging and localized exceptions.

- **CmsFormDataAccess**: The database access layer.

- **CmsFormDatabaseFilter**: Powerful filter to configure database queries.

- **CmsFormDatabaseModuleAction**: Module action that ensures that the required tables for webform persistence are created if they do not exist in OpenCms startup process.

- **CmsFormDataBean**: Bean that stores the data of a single submission of a webform.

- **Messages**: convenience class to access the localized messages of the webform database subpackage.

10.3 com.alkacon.opencms.formgenerator.database.export

This package contains a plain forward implementation for exporting webform submissions within a time – range to Microsoft Excel CSV files. The following classes are used:

- **CmsCvsExportBean**: Implementation of the database export that acts as a façade towards the database access layer

- **Messages**: convenience class to access the localized messages of the webform database export package.

10.4 com.alkacon.opencms.formgenerator.dialog

The package **com.alkacon.opencms.formgenerator.dialog** contains the dialogs to show the configured webforms in the administration view. The following classes are used:

- **CmsFormDataEditBean**: Bean that stores the data of the fields of a submitted data.

- **CmsFormDataListDialog**: Implementation of the detail view of a selected webform.

- **CmsFormDeleteAllEntriesAction**: List action to delete all entries of a form. This special handler is needed because only users with the role "Database Manager" should be able to perform this action.

- **CmsFormDeleteCheckedEntriesAction**: List multi action to delete checked form entries. This special handler is needed because only users with the role "Database Manager" or with write permission on the form VFS file should be able to perform this action.

- **CmsFormDeleteSingleEntryAction**: List action to delete a single form entry. This special handler is needed because only users with the role "Database Manager" or with write permission on the form VFS file should be able to perform this action.

- **CmsFormEditDialog**: Implementation of the detail view of a selected submitted data in the edit mode.

- **CmsFormFileWidget**: Provides a widget to view files uploaded by a web form, for use on a widget dialog.

- **CmsFormgeneratorToolHandler**:

- **CmsFormListDialog**: Implementation of all available webforms, which have the database transport activate.

- **CmsFormRfsFileDownloadDialog**: Generates a download for an RFS file uploaded by the web form.

- **Messages**: convenience class to access the localized messages of the webform database subpackage.