MMBase Editwizard Documentation

Kars Veling
Q42

Henk Hangyi
MMatch

Pierre van Rooden
Publieke Omroep

Table of Contents

Introduction

The editwizards allow designers to easily create wizard-like web-entry forms for content entry and maintance.

The wizards are not intended to edit all possible relations and fields of an object; they are more suitable for special content tasks, such as editing a news item or adding a user to the system and placing images and organisational information along with it. A wizard with more than 3 steps is generally over the top.

In this document you will learn how to install the editwizards on your MMBase release, and how to create your own wizards. First the basics are explained, and how the most simple wizards should be set up. Then, you'll see how more complex wizards can be created.

Installation

If you install and run an 1.6 or higher release of MMBase, the editwizards will be included.

To let the editwizards run properly, the /mmapps/editwizard directory must be in place (default installed). This directory contains the JSP pages, the wizards, images, XSL's and CSS stylesheets and client-side Javascript code - in shirt, all the fiels needed to run the wizards.

For all practival purposes, you only need to remember the location of tey fiels wizard.jsp (which you call to edit a specific object or create a new one), and list.jsp (which you use to list objects which you can then select for edit).

You are free to move the /mmapps/editwizard directory somewhere else. If you do, you need to change the calls in the examples below so they point to the right directory.

Requirements and Assumptions

The editwizards can be used by the following browsers:

- Internet Explorer 5.0 and higher on Windows platforms
- Internet Explorer 5.1 and higher on Apple Macintosh platforms
- Netscape 6.1 and higher on any platform
- Mozilla 0.9.6 and higher on any platform

You'll need a running instance of MMBase (as from version 1.6, the editwizards are included).

Assumptions

The examples below assume the right builders and relations are available for the demo's. If not, make sure the right builders are active. You can do this by deploying the 'MyNews' application (By the admin pages or by setting the autodeploy status of the MyNews.xml to true).

The editwizards rely heavily on XML and XSL. Therefore, this guide assumes you know something about xml nodes and attributes and how to make valid xml documents.

You also need a decent version of an xml parser. MMBase comes with xalan and xerces, which should suffice. If you are an administrator, note that the xalan/xerces jars that ship with orion are older versions. You will need to replace the orion jars with the ones that ship with the MMBase distribution.

Are the editwizards up and running?

To make sure the wizards are up and running, you can try the following:

1. Make sure your MMBase and webserver is up and running
2. Open a browser
3. The examples can be found in the 'mmexamples' subdirectory that is installed with the MMBase distribution. Type in the url to the editwizard examples, eg.: http://localhost:4242/mmexamples/editwizard/
4. A simple html file should show with a few online demo's
5. Click on the link to "A very simple example"
6. Click on the demo "Images"
7. A login screen should popup. Login as you would with the MMEditors or the MMBase admin.
8. If you logged in correctly, a list of images is shown (this can be an empty list).
9. Let's try to upload a new image. Click on the right-bottom plus-icon to upload a new image.
10. A wizard-form is shown. Enter a title of the image and its description. The 'title' and 'description' are required fields. Their prompts stay red until you enter something. Click on 'Upload new image'
11. Browse to an image, select it and click on 'upload'. The page with the image information will refresh, showing the newly uploaded image filename and size. (in Internet Explorer, the image itself can be previewed. In Netscape or Mozilla preview is not available)
12. By clicking on "Save" you store the image in MMBase. You return to the image list. The image you just uploaded is added to this list.

Getting Started

In this section we will make a simple wizard to learn the basics about the wizards. Let's try to build up a simple wizard.

Step 1. Deciding what to do

First, you need to decide what your wizard should do. In our first example, we just want to let the user edit the plain fields of a very simple object. We use the builder: "people" to make our first wizard (this builder is present if you installed the MyNews application).

Users will be able to add, delete and change people-objects (persons in real-life) using this wizard. If you just want to see how it's done, see the working example in the editwizard installation.

We also need to decide where to put our wizards; for the purpose of the current example, we suggest to place them in the directory 'data' inside the mmapps/editwizard directory. This will enable us to directly call the wizards and use the examples (we will later see how you can place your wizard files somewhere else).

Let's say we make a new directory, named "practice" in the "data" directory. So, the path from the web-root looks as follows: mmapps/editwizard/data/practice

Step 2. Create a basic wizard

Create a text file in the subdir you decided, with the extension ".xml". In our example, it should be named: /practice/simple.xml . Open the newly created simple.xml in a text-editor and copy or type the following code into simple.xml. Save the simple.xml file.

```
<?xml version="1.0"?>
<wizard-schema id="some_id">
 <!-- give the wizard a title -->
 <title>Practice</title>

 <!-- define an action to create a people object -->
 <action type="create">
  <object type="people" />
 </action>

 <!-- define an action to delete a people object -->
 <action type="delete" />

 <!-- define an form to edit a people object -->
 <form-schema id="step1">
  <!-- form title -->
  <title>People Wizard</title>

  <!-- edit field 'firstname' -->
  <field name="firstname">
   <prompt>First name</prompt>
  </field>

  <!-- edit field 'lastname' -->
  <field name="lastname">
   <prompt>Last name</prompt>
  </field>

 </form-schema>
</wizard-schema>
```

Step 3. Get it working

Check if your newly created wizard works.

Open your webbrowser and type in the url to the editwizards list.jsp page, as follows (replace with the correct domain name if your site does not run at localhost:4242) :

```
http://localhost:4242/mmapps/editwizard/list.jsp?wizard=practice/simple&nodepath=people
```

The list.jsp page is one of two pages you can call to start the wizards. List.jsp reads the xml you provide as the 'wizard' parameter, and creates a list of items to pick from - either to edit or to delete. It may also provide a button with which you can add new people.

You should see a login screen first. Login as you would with the MMEditors or the MMBase admin. After you logged in, you're screen may look like this:

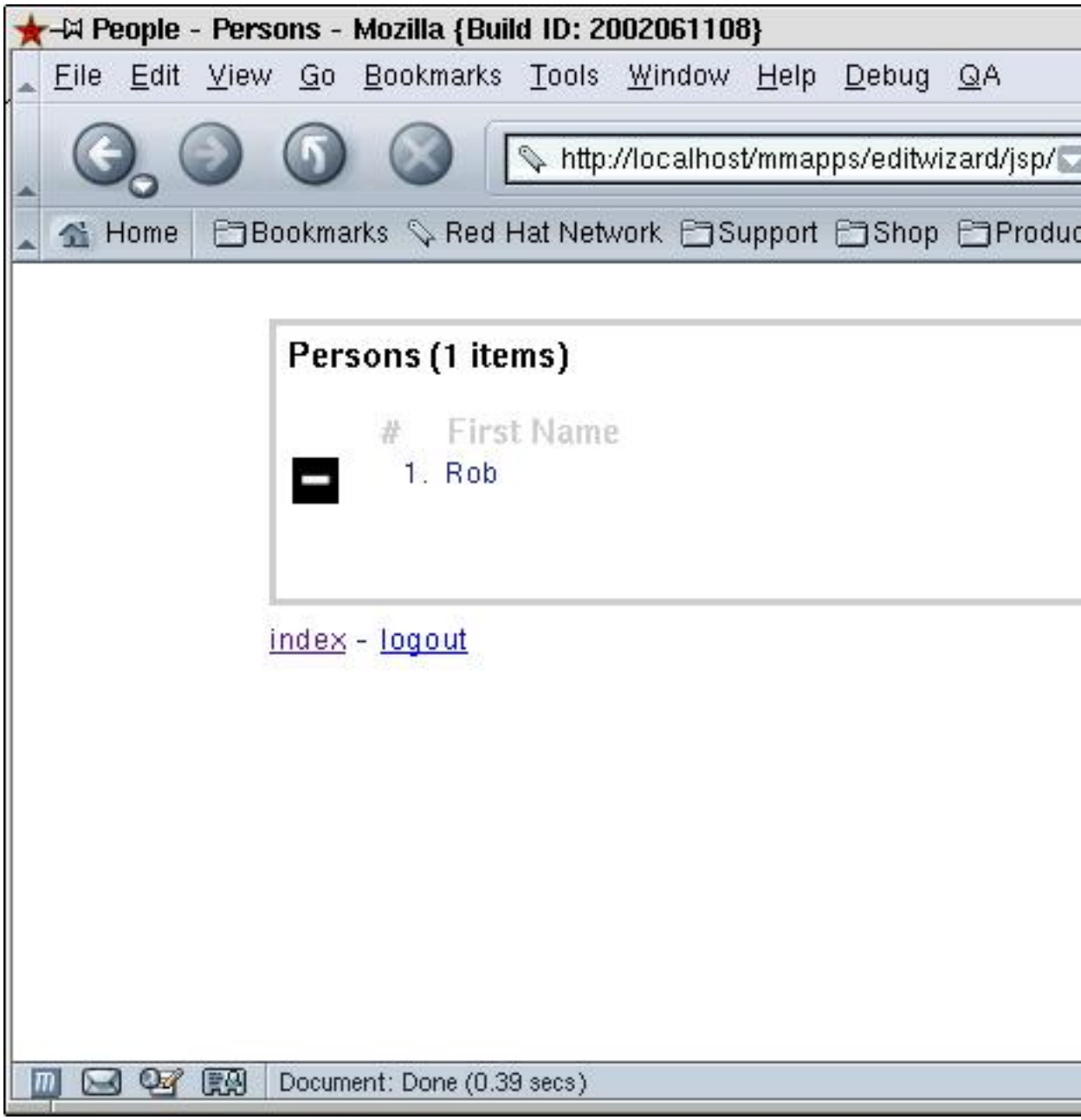


Figure 1. List of persons

There is one person ('Rob') entered in the list example above (this may be different in your case). It has a delete button (the minus-sign) with which you can remove the person. Clicking on the name 'Rob' will allow you to edit the person name. The 'plus' sign in the bottom-right screen allows you to add new people.

Let's create a new person. Click on the plus-icon. If you filled in the editwizard correctly, you should see the following page:

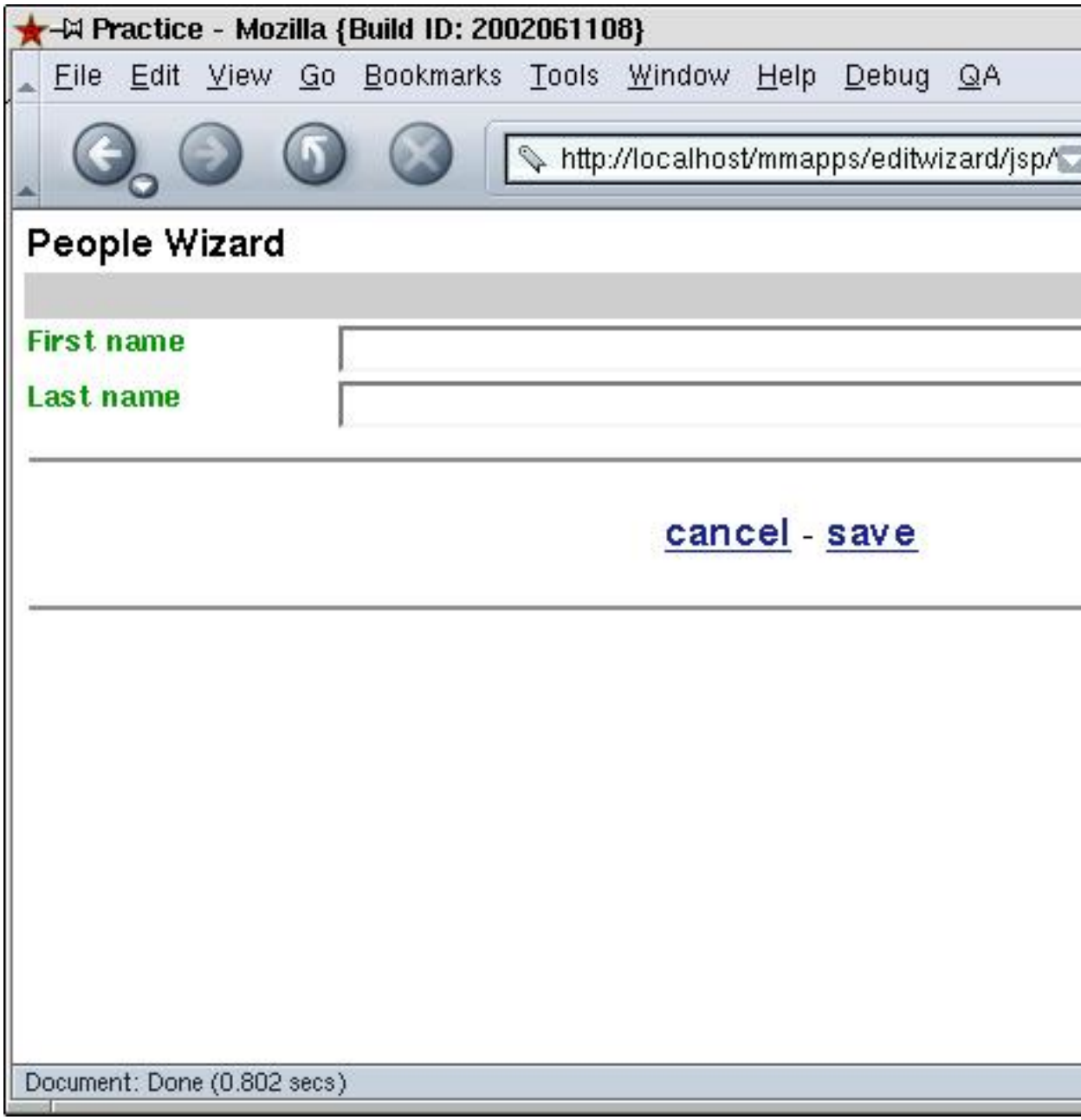


Figure 2. Creating a person.

Here you can see your first wizard in action! Two fields are visible (as defined in the simple.xml file). Fill in the two fields and press save. You are returned to the list, which shoul now contain your newly added person.

You can image you can change the fieldnames, titles and object-types in the wizard so that you can edit different objects in the system. But wizards can do a lot more. We'll get to that in the following section.

Wizard Basics

You can already make and run a simple wizard. In this section, we go deeper into the features of the wizards. We will take the simple wizard from the practice example in the former section, and expand this simple example with new elements.

Create action

The create action defines whether, and more importantly, what should be created in MMBase if the user clicks on the create-button in the list view. A create action is placed inside the /wizard-schema node. You can create multiple objects inside each other (and create relations between them), and place default values in some fields. Example:

```
<action type="create" >
 <!-- create an object of type 'people' -->
 <object type="people" >
  <!-- initialize the firstname field to 'NOBODY'  -->
  <field name="firstname">NOBODY</field>

  <!-- add a relation to a newly created, empty news object  -->
  <relation role="related">
   <object type="news" />
  </relation>
 </object>
</action>
```

This action creates 2 objects and 1 relation. It first creates a new 'people' object, and fills the field "firstname" with the value "NOBODY". It then creates a new 'news' object, and relation with role 'related' between the two new objects.

> **Note:** The relation between people and news need be defined before MMBase will create it for you. As noted previously, the MyNews application defines all these relations for you once installed.

If you do not define a create action, the wizard does not allow you to create anything, and the list page will not show a create button.

Load action

Generally, the wizard needs to know what objects and relations to <load> before the wizard-pages are displayed. By default, it loads the MMBase object that was passed to it, and all its fields. But often we will want to edit more than one object at once. In that case, we need to also load these other objects. You can use the load-action to define the load behaviour of the wizards, forcing it to load related objects so you can edit or display them along with the main object. Example:

```
<action type="load" >
 <!-- for the object passed, only load the firstname and lastname fields -
->
 <field name="firstname"/>
 <field name="lastname"/>

 <!-- load all relations from this object to 'news' objects with role 'pos-
rel' -->
 <relation role="posrel" destinationtype="news" >
```

```
  <object>
   <!-- only load the titel field of these news objects -->
   <field name="title"/>
  </object>
 </relation>
</action>
```

This load action loads only the firstname and lastname fields of the object passed (a person). It also loads all the news objects (but only the title field) that are related to this object.

Delete action

Place a delete action in the wizard if you want to allow the users to delete objects. Try to remove the <action type="delete" /> from the practice wizard and reload the list to see what the impact is.

Note: The delete buttons will disappear.

Note: When you delete an object in the editwizards, all relations with other objects will also be deleted. It is not possible to automatically delete the objects related to, just the relations themselves.

Field format types

You need to specify each field you like to edit in one of the form-schema's in a wizard. Every field has a format type that determines how the field can be edited. Fields can be edited using textboxes, text areas, dropdown lists, or using an upload button, all depending on the fields' format type.

By default, the wizard uses the format types (or guitype) as defined in the builders in MMBase. You can override these types with your own if you prefer another format. In the reference section you will find what field types you can use and what options they have.

Validation

An important element of the editwizards is the ability to do basic validation on the fields. You can add attributes to specify limits on what a field can contain. Eg.:

```
<field name="firstname"> ...
```

can be extended with validation fields. Because the "firstname" field is a text-field, you can apply text-validation rules, such as a minimum or maximum text length. Eg.:

```
<field name="firstname" dtminlength="1" dtmaxlength="20" > ...
```

The field is now only valid if it contain at least 1 and at most 20 characters. As long as the field is invalid, it will be marked red in the wizard, and you cannot save the object

Note: The exact validation rules available are dependent on the field's datatype. The datatype is automatically determined by MMBase. Examples of datatypes are string, integer, and datetime. You can override the datatype to force certain constraints, but it is advised you stick to the types MMBase advises, unless you know what you are doing.

Using lists

A quite important feature of the editwizard are the "lists". With lists, you can show, edit, delete and create objects related to the object you are working on.

For this example you'll need the people and the news objects in MMBase.

Create a new wizard

In the practice directory, create a new file named people.xml. Enter the following xml:

```
<wizard-schema id="people">
 <title>People</title>

 <!-- Action to create a new people object -->
 <action type="create">
  <object type="people">
   <field name="firstname">Enter firstname HERE</field>
  </object>
 </action>

 <!-- Action to delete a people object -->
 <action type="delete" />

 <!-- Action to load a people object, including all posrel relations to news  -
->
 <action type="load">
  <relation destinationtype="news" role="posrel" />
 </action>

 <form-schema id="step1">
  <title>People</title>
  <!-- Show people object -->
  <field name="firstname" >
   <prompt>Firstname</prompt>
  </field>
  <field name="lastname">
   <prompt>Lastname</prompt>
  </field>

  <!-- List all news items related with role posrel
     This list can have a minumum of 0 and a amximum of inifinite ("*") items.
     This is also the default for a list
  -->
  <list role="posrel" destination="news" minoccurs="0" maxoccurs="*">
   <title>Related newsitems</title>
   <!-- Show the 'item' in a list (the object) -->
   <item>
    <field name="title" ftype="data">
     <prompt>Newsitem</prompt>
     <description>Here you can see the name of the related newsitem</description>
    </field>
   </item>

   <!-- defines a search command.
```

```
      This creates a search box that allows you to search for objects
      using a specified ndoepath.
      In this case, the searchs ercahes for news items, using the fields
      'title' and 'subtitle' to show the results of the search in the list.
   -->
   <command name="search" nodepath="news" fields="title,subtitle" age="-
1">
    <prompt>Search newsitems</prompt>
    <!-- Search filters determine what you can search on (in this case, on title
       or both title and subtitle)
       If you do not specify a search filter, the default is to search on the
       'title' field.
    -->
    <!-- Search on title -->
    <search-filter>
     <name>subtitle contains</name>
     <search-fields>subtitle</search-fields>
    </search-filter>
    <!-- Search on title and subtitle
       not the notation in the search-fields for more than one field
    -->
    <search-filter>
     <name>title or subtitle contains</name>
     <search-fields>title|subtitle</search-fields>
    </search-filter>
   </command>
   <!-- Action to create a relation, used by teh search command to create
      the relation once it is found and chosen.
   -->
   <action type="create">
    <relation destinationtype="news" role="related" />
   </action>
  </list>
 </form-schema>
</wizard-schema>
```

Test the wizard

Try the new wizard.

1. Open a browser and go to the wizards. Eg.: http://localhost:4242/mmapps/editwizard/jsp/list.jsp?fields=firstname&wizard=practice/people
2. Login if needed
3. Now you should see a list with all people-objects available in the system. In our example, one user exists (Rob).
4. Let's create a new one. Click on the star-icon in the bottom-right of the window.
5. Now, you should see a wizard-form with your new people-object. By default, the firstname is filled with "Enter firstname HERE".

   Note: You can see that text in the entered wizard above.

6. Fill in your values and press Save. You should see the list again with your new people-object added.
7. To the list, edit the new people object, by clicking on its name.

8. Search for a newsitem to relate it to, by click on the search-icon in the square: "Related newsitems".
9. A popup should be visible now, with all newsitems available at this time. The MyNews application includes a few example items. If none are showing, you can add a new newsitem with the MMBase basic editors (/mmeditors/jsp/).
10. Click on a newsitem of your choice and press Ok. The wizard-form will reload and you should see the newly related newsitem in the square.
11. Press Save. The people object is edited; it is related to the newsitem of your choice. Open the people object again to verify.

Reference

This reference describes the syntax of the wizards and how to access them using a browser.

General - wizard definition schemas

The editwizards need a xml file with a definition of the wizard to operate. By default, those are found in the subdirectory /editwizard/data/, but it is possible to place your definitions, as well as the wizard stylesheets (whichd efine layout) elsewhere.

When the wizards search for a requested file (a wizard xsl stylesheet or a wizard xml schema), they look for the referenced file in the following order:

The referer page directory

> You call the editwizard scripts (list.jsp and wizxard.jsp) from a page located in another directory than the editwizard home directory. If you do, the wizards keep a reference to the calling page, the 'referer'. When a xml or xsl file is requested, the wizard first checks whether the file can be found in the directory of the referer page.
>
> Example: if the file requested is data/my_wizard.xml, and the referencing page is /myeditors/index.jsp, the system first checks for the file /myeditors/data/my_wizard.xml.

The editwizard home directory

> The editwizard home is the directory that contains the ditwizard basic stylesheets, data library, and jsp pages. It is one directory lower than the location of the list.jsp and wizard.jsp files. In the standard distribution, this location is /mmapps/editwizard/ (the list.jsp and wizard.jsp are located in /mmapps/editwizard/jsp/), but you can place the wizards wherever you like.
>
> Example: if the file requested is data/my_wizard.xml, and the editwizard home is in mmapps/ediwtizard/, the system checks for the file /mmapps/editwizard/data/my_wizard.xml.

list.jsp

To run a wizard, you'll need to call either list.jsp or wizard.jsp. These scripts are located in the jsp subdirectory of the editwizard home directory. In the standard distribution, this is mmapps/editwizard/jsp/.

List.jsp starts with running a search query on MMBase to create a list of items to search from. It shows a list of all found nodes and allows you to select one of these items to edit, to remove an item, or to add a new item. The actual options given depend on the security settings (if you are allowed to do this by MMBase), and the possibilities offered by the wizard. Note that the list.jsp does NOT run an editwizard itself, it is just a 'starter'. It eventually calls wizard.jsp (see below) to actually edit or create an item.

You need to specify a number of parameters to tell the list.jsp what wizard should be used, and what nodes and fields should be shown in the list.

The following parameters MUST be specified (list.jsp does not work if you do not specify them):

wizard

The wizard to use. This is a relative reference to the xml schema file. Do not specify the ".xml" prefix.

```
wizard=data/simple.xml
```

nodepath

The nodepath defines the object types to list. This can be one object type, or a comma-separated list of types, which are treated as a relation chain, like in path attribute of the taglib's <mm:list> tag. The last objecttype listed is the object that is used as the objecttype to edit in the wizard.

```
nodepath=people or nodepath=people,news
```

fields

The fields to show in the list. If you query only one nodetype (see nodepath), you can use simple field names. Otherwise, you need to preface the fieldnames with the name of the objecttype they belong to. If the first field listed is a number field, teh objecttype of that field is used as the objecttype to edit in the wizard (instead of the last object type in the nodepath list). This parameter is similar to the fields attribute of the taglib's <mm:list> tag.

```
fields=firstname,lastname or fields=news.title,people.firstname,people.lastname
```

The following parameters MAY be specified:

orderby

A list of fieldnames used to order the returning list. This parameter is similar to the orderby attribute of the taglib's <mm:list> tag.

```
orderby=lastname or  orderby=people.lastname,news.title
```

directions

A list of one or more UP/DOWN keywords. Each keyword determines whether the order of a matching field specified in the orderby parameter is ascending (UP) or descending (DOWN). This parameter is similar to the directions attribute of the taglib's <mm:list> tag.

```
directions=UP or  directions=UP,UP
```

constraints

A SQL-like constraint, used to filter the list. If you use fieldnames in a constraint, surround it with brackets ([]). This allows MMAbse to recognize the fields and optimize the query. This parameter is similar to the constraints attribute of the taglib's <mm:list> tag.

```
constraints=[news.title] like '%Brasil%'
```

distinct

If true, double entries in a list are returned only once.

```
distinct=true
```

pagelength

The maximum number of entries shown on one list page. Default value is 50.

```
pagelength=50
```

maxpagecount

The maximum number of pages available to browse through. Default value is 10.

```
maxpagecount=10
```

start

The number of the item in the list where to start listing. This value is used to skip to a certain page when brosing a large list. Default is 0.

```
age=31
```

age

The maximum age (in days) of the objects to list. Default is no age restriction.

```
age=31
```

searchfields

The field(s) on which to make a search, using the value specified in the search parameter and the type of comparison specified in the search type.

You can specify more than one field (separated by commas). if you do, a search matches if one or more of the specified fields matches.

```
searchfields=news.title,news.subtitle
```

search

The term or value to search for. This is only valid if you also specify the searchfields parameter.

```
search=Brasil
```

searchtype

Determines what type of search is used to filter on the searchfields, using the specified search value. This is only valid if you also specify the searchfields parameter.

Allowed values are 'string', 'like', 'equalto', 'greaterthan', 'lessthan', 'notgreaterthan', and 'notlessthan'. use 'string' to compare the search value to the field using an exact match. Use 'like' to make a pattern search using the specified searchvalue. Use the other values to compare teh feild to numeric values or dates. I.e. using 'greaterthan' selects all fields whose (numeric) value is greater than the specified (numeric) value.

```
searchtype=like
```

origin

A node alias or number, which will be passed to the underlying editwizard. the value can then be referenced in the editwizard schema using the {$origin} variable. This can be used to create relations to nodes whose 'origin' is determined outside the wizard (i.e. in a complex search or determined through user preferences).

```
origin=my_start_node
```

title

An optional title to use by the list. In general, you should specify this in the editwizard schema.

```
title=News of The Week
```

Example 1. How to use list.jsp to list objects

```
list.jsp?wizard=practice/people&nodepath=people&fields=firstname,lastname
list.jsp?wizard=practice/simple&startnodes=1&nodepath=people,news&fields=news.title,pec
```

wizard.jsp

You can use wizard.jsp to directly start a wizard. Like list.jsp, you will need to supply the correct parameters in order to let the wizard run correctly. If you call the wizard.jsp page, a wizard will be loaded and started. The proper html will be rendered and shown in the browser. The syntax of wizard.jsp to create a new object looks like this:

```
wizard.jsp?wizard=practice/simple&objectnumber=new
```

To create a wizard and use an existing mmbase object as source:

```
wizard.jsp?wizard=practice/simple&objectnumber=[objectnumber]
```

Example 2. How to use wizard.jsp to start an editwizard

```
wizard.jsp?wizard=practice/simple&objectnumber=184
wizard.jsp?wizard=practice/simple&objectnumber=new
```

Syntax reference

wizard

Syntax

<wizard-schema />

Usage

This is the root-node of a wizard. Always use a wizard-node to start a wizard.

Needed attributes

None

Possible attributes

id (String) The identifier of the wizard. Not used by the wizards at this time.

Needed childnodes

<title>(String)</title> Title of the wizard

<form-schema /> See also: form-schema.

Possible childnodes

<steps /> (see also: steps)

<action type="load" /> (see also: action)

<action type="create" /> (see also: action)

<action type="delete" /> (see also: action)

<lists /> (see also: lists)

Possible parents

None

Example

```
<wizard-schema>
 <title>Wizard-Title</title>
 <form-schema id="step1">
  <field name="firstname">
   <prompt>Your Firstname</prompt>
  </field>
 </form-schema>
</wizard-schema>
```

action type="load"

Syntax

<action type="load" />

Usage

Use this action to define what the wizard should load when initialising. Mainly, if this node is supplied specific relations will be loaded, specific fields will be loaded or not-loaded. (Note: this syntax should maybe be adjusted in the future.)

Needed attributes

type="load"

Possible attributes

None

Needed childnodes

None

Possible childnodes

<relation destination="[buildername]" role="[rolename]" /> Note: this relation node can contain complex load actions: eg.: relations-in-relations. (Not documented here -for now. See the example below how it works.)

<field name="[fieldname]" />

Possible parents

<wizard-schema />

Example

```
<action type="load">
 <relation destinationtype="people" role="related">
  <object>
   <relation destinationtype="urls" />
  </object>
 </relation>
 <field name="title" />
 <field name="subtitle" />
</action>
```

action type="create"

Syntax

<wizard-schema />

Usage

<action type="create" /> Use this action to define what objects and relations should be created if a user presses the create button. (Or -for that matter- uses the wizard.jsp?objectnumber=new)

Needed attributes

type="create"

Possible attributes

See needed attributes

Needed childnodes

<object type="[buildername]" />

or <relation role="[rolename]" (destinationtype="[buildername]") (destination="[objectnumber]" />

Possible childnodes

See needed childnodes

Possible parents

<wizard-schema />

Example

```
<action type="create">
 <object type="news">
  <relation destination="13234" role="related"/>
  <relation role="posrel">
   <field name="pos">42</field>
   <object type="images">
    <field name="title">new image</field>
   </object>
  </relation>
 </object>
</action>
```

action type="insert"

Syntax

<insert />

Usage

<action type="insert" /> Use this action to insert this <list /> in the editwizard with objectnumber is "new "

Needed attributes

type="insert"

Possible attributes

See needed attributes

Needed childnodes

None

Possible childnodes

None

Possible parents

<list />

Example

```
<list role="posrel" destination="images" minoccurs="0" maxoccurs="*" or-
    derby="field[@name='pos']" ordertype="number">
 <title>Images</title>
 <item displaymode="section" displaytype="image">
  <title>Overview</title>
  <field fdatapath="." ftype="startwizard" objectnumber="{object/@number}" wiz-
    ardname="images">
   <prompt>Change this image</prompt>
  </field>
  <field name="title" ftype="data">
   <prompt>Title</prompt>
```

```
  </field>
 </item>
 <command name="insert"/>
 <action type="create">
  <relation role="posrel">
   <object type="images" >
    <field name="pos">2</field>
   </object>
  </relation>
 </action>
</list>
```

action type="delete"

Syntax

<action type="delete" />

Usage

Use this action to define that users are allowed to delete objects of this kind. If a 'delete-action' is defined in the wizard, the list.jsp will show a delete button. If a delete is performed, only the given object is deleted, NOT the related objects.

Needed attributes

type="delete"

Possible attributes

See needed attributes

Needed childnodes

None

Possible childnodes

<prompt /> If a prompt-node exists, in the list.jsp there will be prompted for a confirmation with the given text.

<description /> If a description is supplied, the given text will be shown as a 'hint' on the delete-button.

Possible parents

<wizard-schema />

<list />

Example

```
<action type="delete">
 <prompt>Are you sure you intent to delete this news item?</prompt>
 <description>Click here to delete this news item</description>
</action>
```

steps

Syntax

<steps />

Usage

Use this node to define in what sequence the form-schema's should be placed. If not supplied, the normal order as found in the wizard's xml file will be used. Usually, you will not use this node.

Needed attributes

None

Possible attributes

None.

Needed childnodes

<step form-schema="[form-schema-id]' /> Place one or more <step /> nodes to define the sequence to be used.

Possible childnodes

See needed childnodes.

Possible parents

<wizard-schema />

Example

```
<steps>
 <step form-schema="basics" />
 <step form-schema="addmedia" />
 <step form-schema="incontext" />
</steps>
```

form-schema

Syntax

<form-schema />

Usage

A wizard will always be needing at least one form-schema. In a form-schema one form (or page) of a wizard is defined. If you are using multiple form-schema's, supply id's so that form-schema's can be identified.

Needed attributes

None

Possible attributes

id="[form-schema-id]"

Needed childnodes

<title /> Place the title of the form-schema here.

<field /> or <list /> (See also: field or list)

Possible childnodes

<description /> Here you can place your description of the form-schema.

Possible parents

<wizard-schema />

Example

```
<form-schema id="step1">
 <title>This is a simple Form</title>
 <field name="intro">
  <prompt>Enter the Intro-text</prompt>
 </field>
</form-schema>
```

object (inside action type="create")

Syntax

<object type="[buildername]" />

Usage

Inside a create-action, you can place an object-node to define what should be created within mmbase if the create-action is performed. With this node, you can define what object should be created, what values should be placed in what fields, and you can define possible new relations that should be created.

Needed attributes

type="[buildername]"

Possible attributes

See needed attributes.

Needed childnodes

None

Possible childnodes

<relation /> If a relation node is placed inside an objectnode, a relation is created relating the object defined by the <object /> node and some other object (eg.: another to-be created object or an existing one.)

<field /> In the field nodes inside the object nodes, the user can define the defaultvalues of the designated fields. Eg.: <field name="firstname">Enter your firstname</field>

Possible parents

<action type="create" />

Example

```
<object type="news">
 <relation destination="13234" role="related" />
 <relation role="posrel">
  <field name="pos">42</field>
  <object type="images">
   <field name="title">new image</field>
  </object>
 </relation>
</object>
```

relation (inside action type="create")

Syntax

<relation />

Usage

Usually, the relation is placed inside an object-node (inside a create-action), or it will be placed directly inside a create-action. With this node, the user can make relations between two newly created objects, or, create a relation between one newly created object and an already existing object.

Needed attributes

None

Possible attributes

destinationtype="[buildername]" This attribute is not used anymore. Forget it I'd say.

destination="[objectnumber or alias]" Use this attribute to point the relation to an already existing object in the mmbase cloud.

role="[relationname]" Use this attribute to define what kind of relation should be created. If omitted, the default InsRel relation is used. Best practice is to always define the role, eg.: "related" or "posrel" etc.

Needed childnodes

None

Possible childnodes

<object /> (see also: object)

<field /> Default values of fields of a relation itself can be set also. Eg.: <field name="pos">-1</field>

Possible parents

<action type="create" />

<object />

Example

(See also: the <action type="create" /> example and the <object /> example.

field (general)

Syntax

<field />

Usage

Field nodes define what form-elements will be placed in the wizard. This fieldnodes in the xml are an important part of the wizard's definition. For detailed information on what fieldtypes need what kind of settings, see the other fielddefinitions in the reference. Note: the field nodes that can be

placed inside a relation or object node have different syntax! (See also: <object type="create" />)

Needed attributes

name="[fieldname]" fdatapath, ftype, dttype, dtrequired, dtminlength, dtmaxlength are automatically retrieved from mmbase. Overriding is always possible, however.

or (advanced users:) fdatapath="[xpath]" and ftype="[fieldtype]"

Possible attributes

ftype="[fieldtype]" (line | text | enum | date | int)

dttype="[datatype]" (string | int | date | datetime | time | html)

dtminlength="[minlength]"

dtmaxlength="[maxlength]"

dtrequired="[true | false]"

rows="[rowcount]"

Needed childnodes

None

Possible childnodes

<prompt /> For every field, a prompt-text can be given. If defined, the prompt text will be visible in front of the field in the wizard.

<description /> For every field, a description can be given. If defined, the description will be shown "onmouseover". If the user move the mouse over the field, the description will be shown in a hint.

Possible parents

<form-schema />

<item />

Example

```
<form-schema id="step1">
 <title>Example form</title>
 <field name="title" dtminlength="1" ftype="line">
  <prompt>News Title</prompt>
  <description>You can enter the news-title here</description>
 </field>
</form-schema>
```

field ftype="line"

Syntax

<field ftype="line" />

Usage

A line-field will show a single-line inputfield in the wizard. Use them for simple text entry.

Needed attributes

None

Possible attributes

dttype="string"

dttype="int"

dtminlength="[minlength]"

dtmaxlength="[maxlength]"

dtrequired="[true|false]"

Needed childnodes

None

Possible childnodes

see field (general)

Possible parents

see field (general)

Example

```
<field name="title" dtminlength="1">
 <prompt>Title</prompt>
</field>
```

field ftype="int"

Syntax

<field ftype="int" />

Usage

Use this field for number editing

Needed attributes

None

Possible attributes

dtmin="[minvalue]"

dtmax="[maxvalue]"

dtrequired="[true|false]"

Needed childnodes

None

Possible childnodes

see field (general)

Possible parents

see field (general)

Example

```
<field name="score" dtmin="100" dtmax="5000" dtrequired="true"> <
 <prompt>Enter position</prompt>
 <description>Enter value between 100 and 5000 please.</description>
</field>
```

field ftype="date"

Syntax

<field ftype="date" />

Usage

Use this field to edit date, date-time, or time fields.

Needed attributes

None

Possible attributes

dtmin="[mindate]"

dtmax="[mindate]"

dttype="[date | datetime | time]"

Needed childnodes

None

Possible childnodes

see field (general)

Possible parents

see field (general)

Example

```
<field name="start" dttype="date">
 <prompt>Startdate</prompt>
</field>
```

field ftype="upload"

Syntax

<field ftype="upload" />

Usage

Use this field to process uploads. Note: Make sure that you use this field in the right context: Usually, this field will store it's binary-value in a mmbase field named 'handle'. See the upload example for more info.

Needed attributes

None

Possible attributes

None

Needed childnodes

see field (general)

Possible childnodes

see field (general)

Possible parents

see field (general)

Example

```
<wizard-schema>
 <title>Image Upload</title>
 <action type="create">
  <object type="images" />
 </action>
 <action type="load">
  <field name="title" />
  <field name="description" />
 </action>
 <form-schema id="step1">
  <title>Image upload</title>
  <field name="title">
   <prompt>Title</prompt>
  </field>
  <field name="description" ftype="text" rows="8">
   <prompt>Description</prompt>
  </field>
  <field name="handle" ftype="image" dttype="binary">
   <prompt>upload</prompt>
  </field>
 </form-schema>
</wizard-schema>
```

field ftype="startwizard"

Syntax

<field ftype="startwizard">

Usage

Use this field to start one wizard, from inside another wizard.

Needed attributes

objectnumber="{object/@number}"

wizardname="[name of the wizard]"

Possible attributes

inline="[true | false]" an inline startwizard will replace the current wizard to create the new object and come back when ready, a not-inline startwizard will pop-up a window to create the new object .

Needed childnodes

None.

Possible childnodes

None.

Possible parents

<item />

Example

```
<command name="startwizard" inline="false" wizardname="tasks/myurls" ob-
jectnumber="new"/> <command name="startwizard" inline="true" wizard-
name="tasks/myurls" objectnumber="new"/>
```

field ftype="data"

Syntax

<field ftype="data" />

Usage

Use this field to show values, but don't make it editable. In other words: use this to make a read-only field.

Needed attributes

None

Possible attributes

see field (general)

Needed childnodes

None

Possible childnodes

see field (general)

Possible parents

None

Example

```
<field name="firstname" ftype="data">
 <prompt>Your firstname is:</prompt>
</field>
```

field ftype="enum"

Syntax

<field ftype="enum" />

Usage

Use this field to make a "dropdown" inputfield. (In HTML-terms: a selectbox).

Needed attributes

None

Possible attributes

None

Needed childnodes

<optionlist /> Use this node to define the possible options for the field. (see also: optionlist)

Possible childnodes

see field (general)

Possible parents

see field (general)

Example

```
<field name="type" ftype="enum">
 <prompt>Articletype</prompt>
 <optionlist name="articletypes">
  <option id="1">News</option>
  <option id="2">Interview</option>
  <option id="3">Information</option>
 </optionlist>
</field>
```

list

Syntax

<list destination="[buildername]" role="[rolename]" />

Usage

Use this to show and edit relations and the related objects.

Needed attributes

destination="[buildername]"

role="[rolename]"

Possible attributes

minoccurs="[minimal required number of items in the list]"

maxoccurs="[maximum number of items in the list]"

Needed childnodes

<item /> (see also: item)

Possible childnodes

<action type="create" /> (see also: action)

<action type="insert" /> (see also: action)

<command name="search" /> (see also: command)

Possible parents

<form-schema />

<list />

Example

```
<list role="related" destination="images"> <command name="search" node-
path="images" fields="title" age="-1"> </command> <item> <field name="title" ftype=
tinationtype="images" /> </action> </list>
```

lists

Syntax

<lists />

Usage

Use this node to define optionlists in a wizard.

Needed attributes

None

Possible attributes

None.

Needed childnodes

<optionlist /> (see also:optionlist)

Possible childnodes

See needed childnodes

Possible parents

<wizard-schema />

Example

```
<lists>
 <optionlist name="priorities">
  <option id="1">Low</option>
  <option id="2">High</option>
 </optionlist>
</lists>
```

optionlist

Syntax

<optionlist />

Usage

Use this node to define an optionlist.

Needed attributes

None

Possible attributes

name="[optionlistname]" This optional attribute can be used if an optionlist is defined in a <lists /> node. To reference to the defined optionlist, you will need the optionlistname. Eg.: <optionlist select="earlier_defined_optionlist" />

Needed childnodes

<option id="[optionid]">[optionvalue_to_be_shown_in_wizard]</option>

Possible childnodes

See needed childnodes

Possible parents

<wizard-schema />

<field ftype="enum" />

Example

See the example for lists

command (name="search")

Syntax

<command name="search" />

Usage

Use this command inside a list node to define how a user can search for objects and add them to a list.

Attributes (general)

See for detailed information how to use nodepath, startnodes, fields, constraints, orderby the documentation of the MMBase taglib.

Needed attributes

name="search"

nodepath="[buildername_to_start_with]"

fields="[fieldnames of fields to show]"

Possible attributes

startnodes="[objectnumber]"

age="[default age to show in search-field]" use -1 to set it to "any age", use 1,7,31,365 for day, week, month, year ages.

constraints="[mmbase where part]"

Needed childnodes

None

Possible childnodes

<prompt /> Place the text to be shown in front of the searchfield here.

<search-filter /> Defines what extra searchfields should be shown and used in the query. See also: search-filter.

Possible parents

<list />

Example

```
<list destination="people" minoccurs="0" maxoccurs="*">
 <command name="search" nodepath="people" fields="firstname,lastname" or-
    derby="lastname" age="-1">
  <prompt>#Search command prompt</prompt>
  <search-filter>
   <name>firstname contains</name>
   <search-fields>firstname</search-fields>
  </search-filter>
  <search-filter>
   <name>lastname contains</name>
   <search-fields>lastname</search-fields>
  </search-filter>
 </command>
 <item>
  <field name="firstname" >
   <prompt>Field prompt</prompt>
   <description>Field Description</description>
  </field>
 </item>
 <action type="create">
  <relation destinationtype="people" role="related" />
 </action>
</list>
```

item

Syntax

<item />

Usage

Every list needs a way to present each occurrence in the list. These occurrences are named: "item". In an item you can place the same fields as inside a form-schema. So, usually one or more fields are placed in an item.

Needed attributes

None

Possible attributes

None

Needed childnodes

<field />

or <list />

Possible childnodes

<field />

or <list />

Possible parents

<list />

Example

```
See the example for list
```

search-filter

Syntax

<search-filter />

Usage

Use the searchfilter to allow the user to fire a free-text query from the wizard. See the command name="search" example.

Needed attributes

None

Possible attributes

None

Needed childnodes

<name />

<search-fields />

Possible childnodes

None

Possible parents

<command name="search" />

Example

```
<search-filter>
 <name>Naam bevat</name>
 <search-fields>name</search-fields>
</search-filter>
```

Configuration of the editwizards

The editwizards can be configured in a number of ways: where you store the xml configuration files, how the stylesheets and the templates of the editwizards and the language of the prompts.

Location of files

The editwizards xml files have to be in a subdirectory of the directory from where the list.jsp or wizard.jsp is referred (where the 'entrance page' is). The directory from which wizard.jsp and list.jsp are called is denoted the "referrer"-directory.

To make sure the "referrer" directory can always be determined, it is a good idea to supply the 'calling file' it to the editwizard-jsp's as an argument (taglib example):

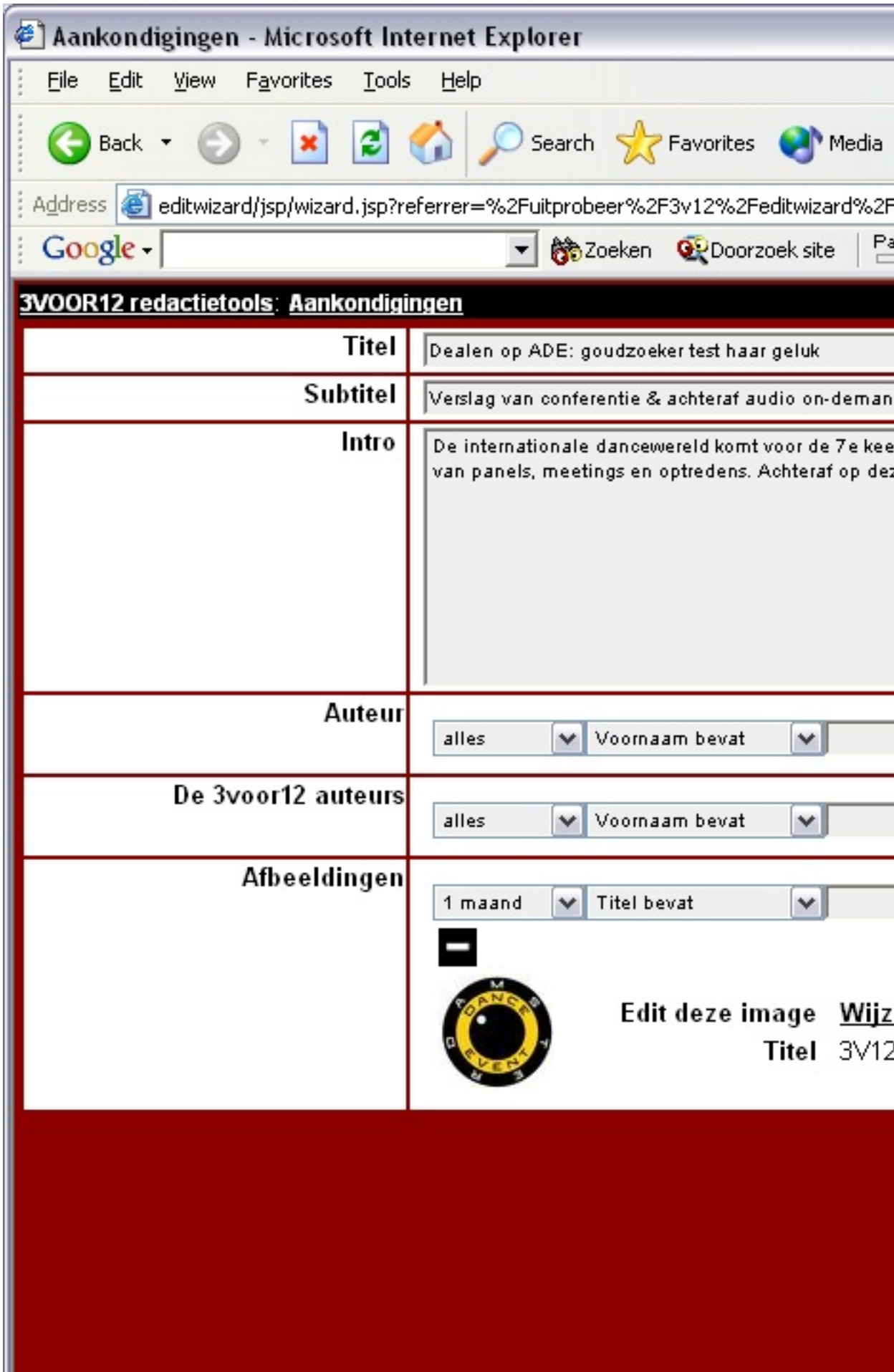
```
<mm:import id="referrer"><%=new java.io.File(request.getServletPath())%></mm:import>
<mm:import id="jsps">/mmapps/editwizard/jsp/</mm:import>
<mm:import id="language">nl/</mm:import>
 ..
 ..

<a href="<mm:url referids="referrer,language" page="${jsps}list.jsp">
      <mm:param name="wizard">tasks/people</mm:param>
      <mm:param name="nodepath">people</mm:param>
      <mm:param name="fields">number,firstname,lastname</mm:param>
      <mm:param name="orderby">number</mm:param>
      <mm:param name="directions">down</mm:param>
      </mm:url>">Person test</a>
```

Without this parameter the HTTP-request header 'Referer' will be consulted, but this header is for browsers only optional.

Manipulating the editwizard layout

The standard editwizard templates and stylesheets can be overridden to adjust them to your own needs and wishes. See the example below on what is possible.


Aankondigingen - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address editwizard/jsp/wizard.jsp?referrer=%2Fuitprobeer%2F3v12%2Feditwizard%2F
Google Zoeken Doorzoek site
3VOOR12 redactietools: Aankondigingen
Titel
Dealen op ADE: goudzoeker test haar geluk
Subtitel
Verslag van conferentie & achteraf audio on-deman
Intro
De internationale dancewereld komt voor de 7e kee
van panels, meetings en optredens. Achteraf op dez
Auteur
alles
Voornaam bevat
De 3voor12 auteurs
alles
Voornaam bevat
Afbeeldingen
1 maand
Titel bevat
Edit deze image Wijz
Titel 3V12

To override classes of the default stylesheet in "/mmapps/editwizard/style" place a file style.css in the "referrer"-directory and add any classes you wish to override to this file. This style sheet must be configured as 'extrastyle' to the XSL Transformations. You do that by placing a file xsl/base.xsl in the "referrer"-directory with the following content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  ver-
sion="1.0">
 <xsl:import href="ew:xsl/base.xsl" /> <!-- extend from standard  ed-
itwizard xslt -->

 <xsl:template name="extrastyle">
  <link rel="stylesheet" type="text/css" href="{$referrerdir}style.css" ></link>
  <!-- the $referrerdir variable can be used to to construct the right path to the 're-
ferrer' directory -->
 </xsl:template>

</xsl:stylesheet>
```

So, this means that you can override any xsl templates in /mmapps/editwizard/data/xsl. Create files like wizard.xsl, list.xsl and base.xsl and add the templates you wish to override in these files. Create a directory "xsl " in the "referrer"-directory and place your own *.xsl files in this directory.

Internationalisation

The default (english) prompts of the editwizards are defined in /mmapps/editwizard/data/xsl/prompts.xsl. If the language is not english then the 'prompts.xsl' is first search for in the directory /mmapps/editwizard/data/i18n/<language code>/.

The language is determined by the mmbaseroot.xml setting, but you can also specify it by supplying a "language" parameter to the editwizard jsp's, then your pages are independent of this setting.

Notes

1. http://www.mmbase.org/license