SAP NetWeaver
How-To Guide

How To Add Image Crop and Resize Capabilities in Web-based Applications

Applicable Releases:

SAP NetWeaver 04 >= SPS 21

SAP NetWeaver 7.0

IT Practice:

User Productivity Enablement

IT Scenario:

Enabling User Collaboration

Version 1.0

December 2008

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1. Scenario

In some customer scenarios users can upload images in Web-based applications e.g. profile pictures which will then be displayed e.g. in user profiles. In simple customer scenarios the users can just select any picture in any dimension and upload it in the application. In more complex customer scenarios though, the image might need to meet some prerequisites like the image dimension or the resolution. One way to deal with it is to modify the picture in an image editor like Adobe Photoshop and to upload the modified picture. Another way is to embed basic image edit functionalities to your Web-based application.

In the SAP NetWeaver Portal users can upload images e.g. user photos and store them e.g. as user specific properties which will then be displayed as the users' image. Therefore the SAP NetWeaver Portal provides a basic Web-based application for the photo upload. The user photo maintenance in the Collaboration area of the SAP NetWeaver Portal provides this upload functionality but the image has to meet the prerequisites like a dimension of 100 x 130 pixels or a resolution of 24 bit per pixel. If the image dimension is too large the uploaded will simply be rejected. In order to avoid this step in between this How to Guide will show you how to add basic image edit capabilities like cropping and resizing to your Portal application based on an enhanced version of the "User Photo Maintenance" in the SAP NetWeaver Portal.

In general most of the described steps can also be applied to add basic image edit capabilities to any Web-based Java application.

2. Background Information

2.1 JavaScript - Image Cropper UI library

The JavaScript Image Cropper UI is a JavaScript library which can be integrated in any Web-based application which requires image edit capabilities like cropping or resizing the image. The Image Cropper UI allows you to select any detail of an image displayed in a HTML-page, provides preview functionalities of the selected details and allows you to enlarge or scale down the selected image area.

For further details have a look at the JavaScript Image Cropper UI Homepage. If you want to have a look at the features of JavaScript Image Cropper have a look at the Demo page.

2.2 Java - image edit functionality

What the JavaScript library mainly does, is to provide the selected image details as coordinates of the original image e.g. a x- and y-coordinate (starting from (0,0) of original image) and width x height of the selected area.

In addition to that it is required to make use of Java to modify the image based on the information provided by the JavaScript library. As soon as the user has selected and submitted the desired image area the information about the coordinates is sent to a Java class (e.g. a Servlet or Portal Component) on the SAP NetWeaver Application Server Java which then resizes, crops and stores the modified image.

2.3 Implementation & usage in sample application

In the next steps it is explained how to integrate and bundle all the different parts together based on a sample application, which enables a Portal user to upload, edit and store photos for the Portal user profile picture. Sample application details:

- Portal Application
- UI Technology - HTML-Business for Java
- JSP pages (usage of htmlb taglib) - easy integration of JavaScript in JSP
- JavaScript Image Cropper UI - crop and resize image
- JSPDynPage - acting as Controller
- Image functionalities in Java SDK
- Simple Servlet - sending image as HTTP response to client

2.4 Sample application overview

Supported SAP NetWeaver versions:

- SAP NetWeaver 04 >= SPS 21

or

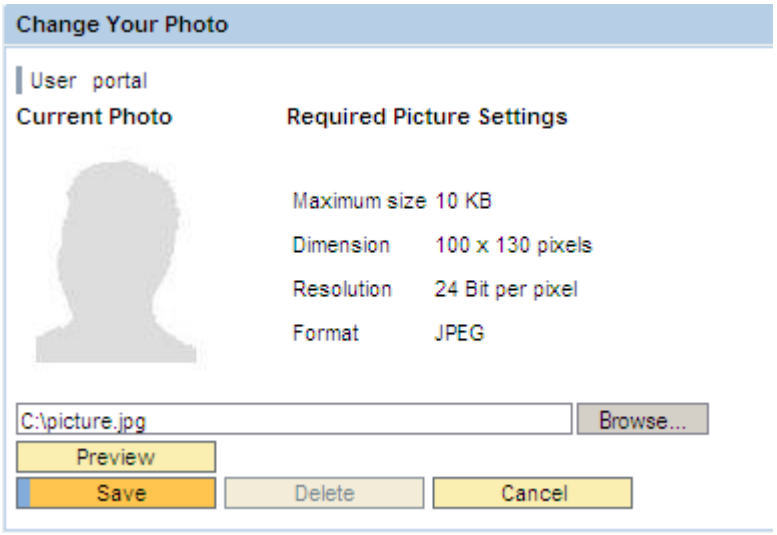
- SAP NetWeaver 7.0

2.4.1 Feature overview

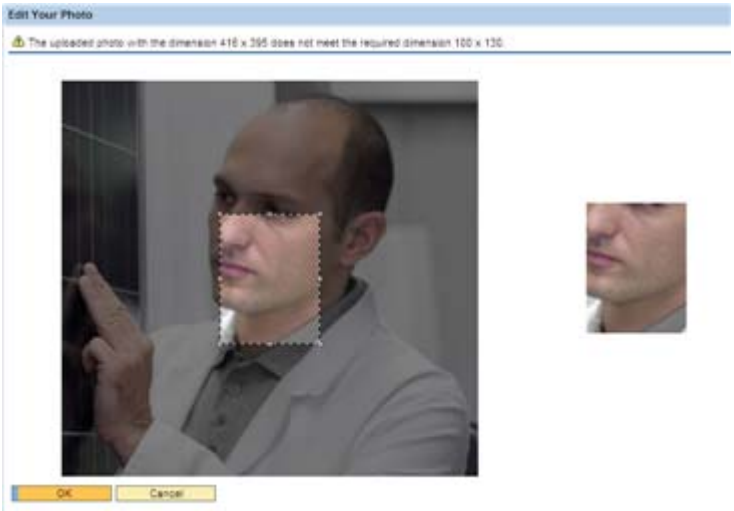
- Supported Format: JPEG
- Client-side image cropping and resizing
- Preview of selected image
- Server-side image cropping and resizing
- Scale down image to defined image edit area
- Configuration of image parameters in Portal configuration
- Supported languages: German, English

2.4.2 Samples

- Start page (upload, delete, save picture)

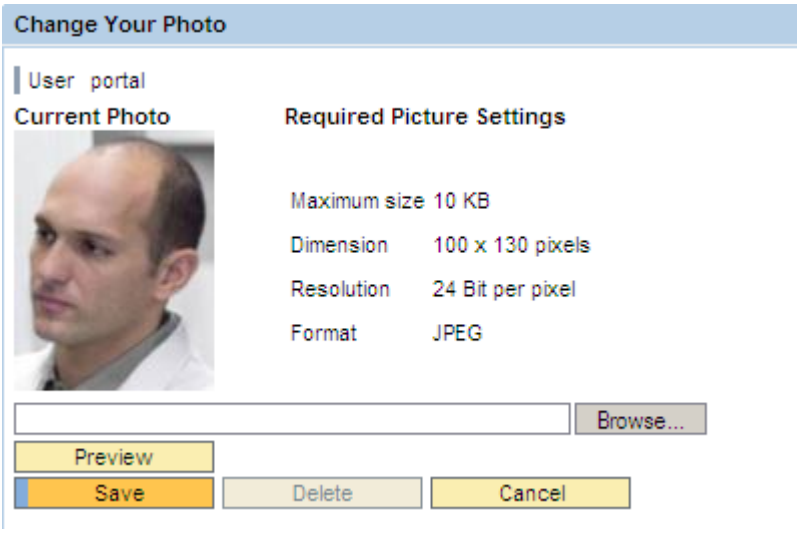


- Edit page (Rescaled image in order to fit to edit area and image preview)



- Edit page (rescaled image selection)

- Start page (modified image)



3. Prerequisites

3.1 Software Requirements

- SAP NetWeaver '04 Portal 6.0 >= SPS 21

or

- SAP NetWeaver 7.0 Portal 7.0

> **Note**
>
> If you want to develop your own application based on this How-To Guide you need a SAP NetWeaver Developer Studio as well.

3.2 Required Documentation / Knowledge

You should be familiar with the administration of SAP NetWeaver Portal and SAP NetWeaver Application Server Java to follow the instructions in this How-To Guide.

4. Obtain and deploy sample application

This section describes how to obtain and deploy the Image Upload.

4.1 Obtain Image Upload

> **Note**
>
> Please be sure to have the role of a system administrator to be enabled to deploy the PAR file.

Obtain the required Image Upload Application that is available as a ZIP-archive. Please use the "Download" Link provided in the abstract.

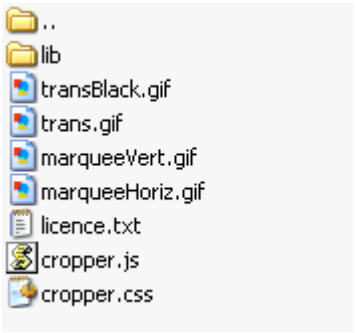
4.2 Deploy Image Upload

This section explains how to modify and deploy the PAR file to your SAP NetWeaver Portal. If you want to create your own application or want to have a better understanding of how the sample application works have a look at the Step-by-Step procedure. Modifying PAR file of the sample application:

> **Note**
>
> You can either modify the application with the SAP NetWeaver Developer Studio or directly modify the PAR file.

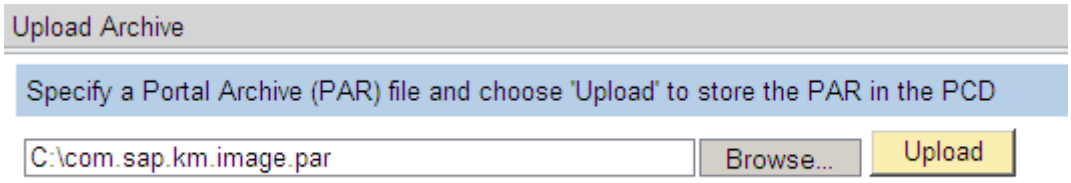
1. Download the PAR file
2. Download JS Cropper UI
3. Unzip the JS Cropper UI to e.g. **`c:\tmp\cropper`**
4. Open the PAR file with e.g. WinRAR
5. Navigate to the scripts folder
6. Navigate to cropper location e.g. **`c:\tmp\cropper`**
7. Change the cropper.css as described in the Step-by-Step procedure - step 3
8. Select following files / folders and add them to the scripts folder of your PAR file

9. The scripts folder of your PAR file should now contain following files:



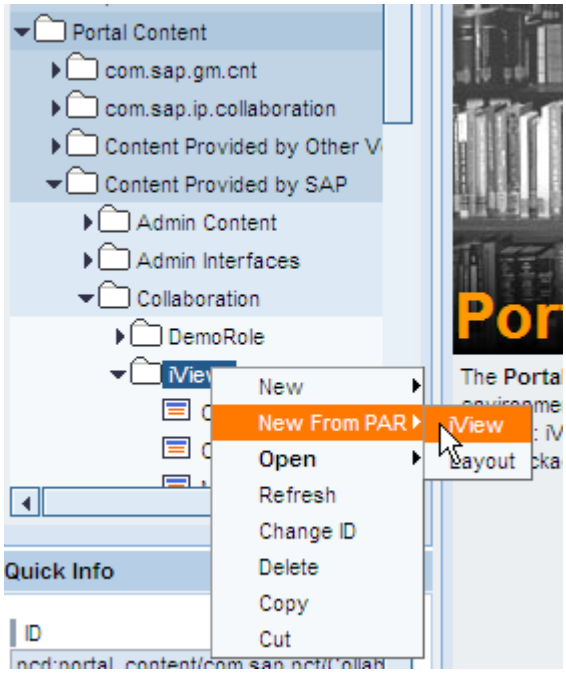
10. Deploy the PAR file to your SAP NetWeaver Portal

- Logon to your SAP NetWeaver Portal
- Navigate to System Administration -> Support
- Choose Portal Runtime from the Top Level Areas
- Start the Administration Console
- Select the PAR file and press the upload button

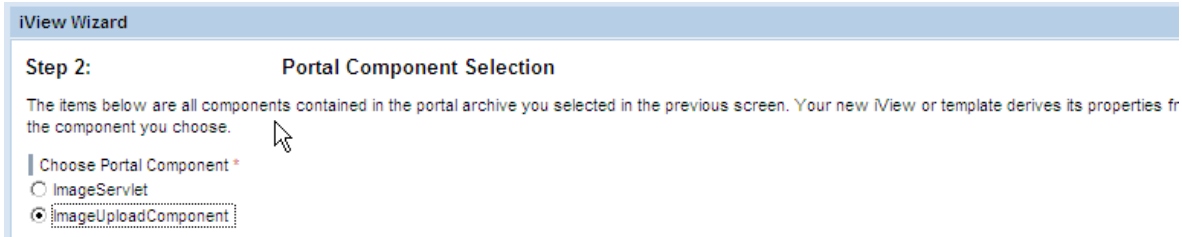


11. Create an iView for your PAR file

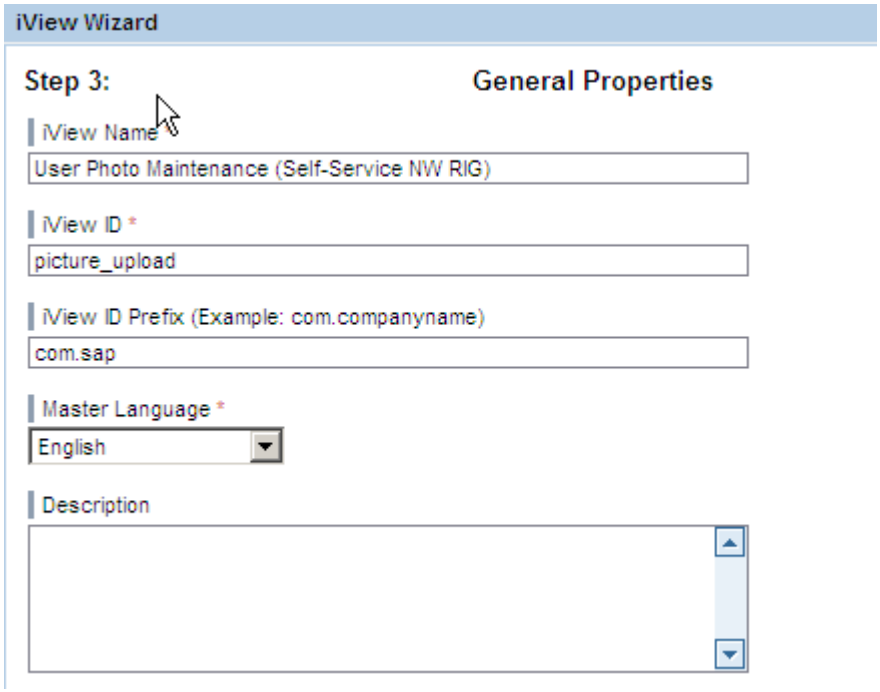
- Logon to your SAP NetWeaver Portal
- Navigate to Content Administration -> Portal Content
- Choose a location in the Portal Content
- Select Create iView from par file in the context menu



- Select the com.sap.km.image.par file from the list and press Next
- Select the ImageUploadComponent as Portal Component and press Next



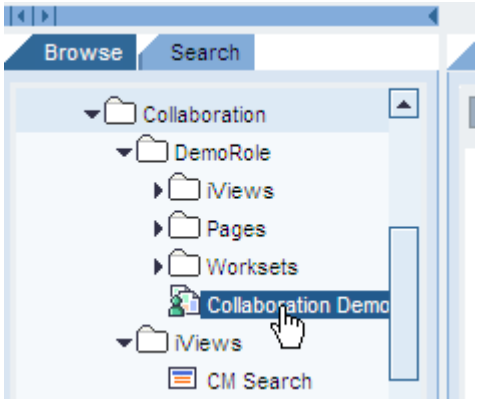
- Specify the General iView Properties and press Next



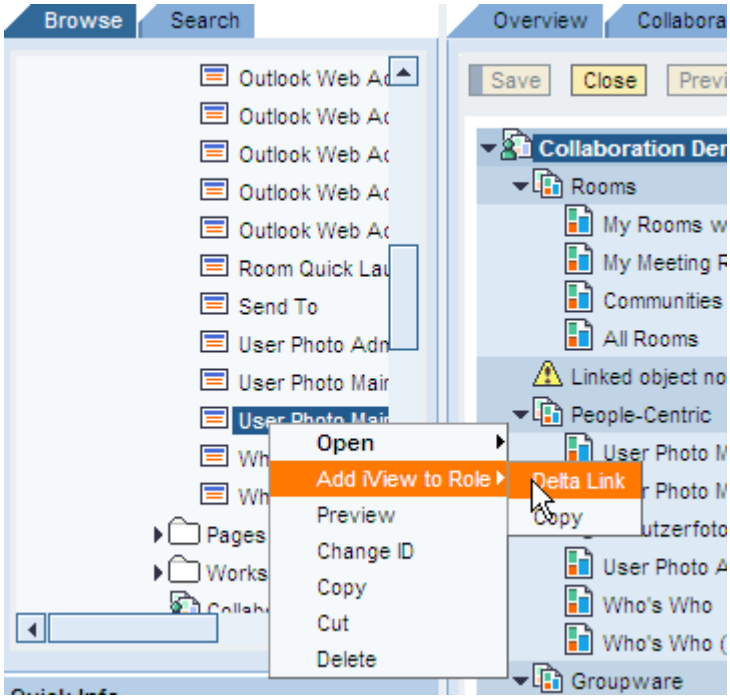
- Press Finish to complete the iView creation

12. Integrate your iView in the Portal Navigation

- Select the role in which you want to add your iView by double-clicking on the node



- Select your iView and select Add iView to role as Delta Link from the context menu



- Move the link to the desired place

5. Step-by-Step Procedure

The following part explains step-by-step how to create a Portal application with some basic edit capabilities. First of all a rough explanation guides you through the creation process of a Portal application project. In the next steps it is then mainly about how to add the edit capabilities to the application and how to integrate the JavaScript Image Cropper UI library.

As already mentioned above most of the steps have to be performed as well in any other Web-based project when you want to integrate basic image edit functionalities. The steps itself might vary a bit but the basic principles are the same.

Note

In general the principles for a Web-based Java application are more or less the same. You have a Controller which processes the user actions and a View which contains the user interface. The View consists of an image, some invisible input fields and some JavaScript code to add the edit capabilities. The Controller processes the user request and crops / resizes the image based on the selected image details.

5.1 Step 1: project creation / Required Java classes

- Create a Portal Application Project in your SAP NetWeaver Developer Studio
- Add an AbstractPortalComponent to your project
- Add a JSPDynPage to your project and enable the Java-Bean creation in the wizard (creates one Java class which extends the class JSPDynPage, one Java-bean class and one JSP-file)
- Add one attribute e.g. imageUrl of type String to your bean class
- Add another Java class to your project which extends the class HttpServlet
- Navigate to the portalapp.xml file
  - Remove the entry <property name="ComponentType" value="jspnative"/>
  - Create a component entry for your Servlet

```
<component name="ImageServlet">
    <component-config>
        <property name="ClassName"
        value="com.sap.km.image.servlet.ImageServlet"/>
        <property name="ComponentType" value="servlet"/>
    </component-config>
</component>
```

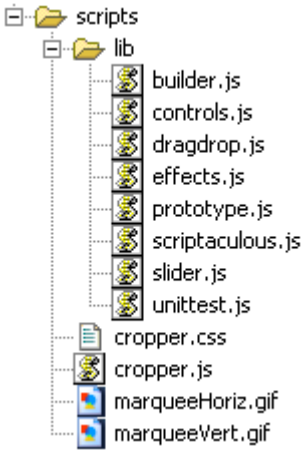
Tip

For further details about how to create a Portal application project refers to the Portal Development Kit which provides a detailed overview of how to create and develop a Portal application.

5.2 Step 2: integrate the JavaScript files to your portal project

This step describes how to integrate the JavaScript Cropper UI to your portal project.

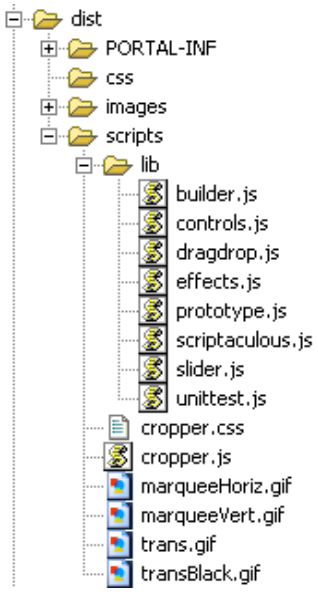
1. Download the JavaScript Cropper UI from here.
2. Unzip the file.
3. Copy the files to the path dist/scripts of your Portal Application project
   - Copy all files except the test folder and the cropper.uncompressed.js to your dist/scripts folder
   - Your script folder should now contain the files shown in the following picture:



5.3 Step 3: JS Cropper UI in nested <table> structure

The current JS Cropper UI version doesn't work properly when your image is enclosed in an html <table> tag (htmlb positions elements by tables). In order to make JavaScript Cropper UI working complete the following steps:

1. Copy the following two pictures to your dist/scripts folder (already included in sample application):
   - transBlack.gif
   - trans.gif
   - Your scripts folder should now look like the following picture:



2. Open the cropper.css and edit / remove the following parts as shown below:

```
.imgCrop_overlay {
  position: absolute;
  width: 100%;
  height: 100%;
- background-color: #000;
- opacity: 0.5;
- filter:alpha(opacity=50);
  background-image: url(transBlack.gif);
}
.imgCrop_clickArea {
  width: 100%;
  height: 100%;
- background-color: #FFF;
- opacity: 0.01;
- filter:alpha(opacity=01);
  background-image: url(trans.gif);
}
```

3. Save your changes.

> **Note**
>
> This modification is only required when you embed your image in a html table. It is not required when you don't use the <table> at all in your Web-Page.

5.4 Step 4: Integration of JavaScript in JSP page

The next step shows you how to integrate the JavaScript part in your JSP file. In order to get access to the controls we need some variables to store the IDs generated by htmlb. The helper class ApplicationUtil integrates the required JS files in the JSP file. The next step is to integrate the CSS as it needs to be embedded in your JSP page otherwise it wouldn't work properly. The image retrieves some data from the bean and displays the image corresponding to the data set in the bean. The invisible input fields store the coordinates given by the JavaScript which can then be sent to the server.

1. Open the JSP page created in step 1.
2. Include the required JavaScript files in your JSP (Use methods in ApplicationUtil class).

```
// include required javascript files
ApplicationUtil.includeJS(pageContext,"scripts/lib/prototype.js",componentRequest);
ApplicationUtil.includeJS(pageContext,"scripts/lib/scriptaculous.js?load=builder,dragdrop",componentRequest);
```

Note

The ApplicationUtil class is a helper class to easily integrate JavaScript files in your JSP.

3. Declare variables in your JSP for storing the IDs of the HTML controls.

```
// storing ids of different controls
String formId = "";
String coordx1 = "";
String coordy1 = "";
String width = "";
String height = "";
```

Important

Make sure that you declare these variables at the beginning of your JSP. When you don't use the HTMLB Tag-Library it is not required to store the IDs as you can directly access the elements by their names.

4. Add CSS to your JSP File after <hbj:page...> tag (ImageEdit.jsp - jsp file in par file):

```
...
<hbj:content
    id="myContext">
    <hbj:page
        title="PageTitle">
        <style type="text/css">
            label {
                clear: left;
                margin-left: 50px;
                float: left;
                width: 5em;
            }
            #testWrap {
                width: 500px;
                float: left;
                margin: 20px 0 0 50px;
            }

            #previewArea {
                margin: 20px; 0 0 20px;
                float: left;
            }
        </style>
    </hbj:page>
</hbj:content>
...
```

5. Retrieve the form Id generated by htmlb framework:

```
<hbj:form id="myForm">
<%
    // get the id of the form
    String formId = myContext.getCurrentFormId();
%>
```

6. Add image to your JSP page:

```
...
<div id="testWrap" style="height:<%=imageBean.getHeight()%>px;">
<img src="<%=imageBean.getImageUrl()%>"
  alt="<%=res.getString("image.edit.uploaded_image")%>"
  id="testImage"
  width="<%=imageBean.getWidth()%>"
  height="<%=imageBean.getHeight()%>">
</div>
<div id="previewArea"></div>
...
```

Note

The source of the image has to point to the URL of the Servlet. All information about the image is stored in a Java-Bean class. This information is retrieved after the image upload and set in the Java-Bean class.

7. Add invisible input fields in order to send the selected coordinates to your controller:

```
...
    <hbj:inputField id="x1" visible="false" type="string" jsObjectNeeded="true">
        <% coordx1 = myContext.getParamIdForComponent(x1); %>
    </hbj:inputField>
    <hbj:inputField id="y1" visible="false" type="string" jsObjectNeeded="true">
        <% coordy1 = myContext.getParamIdForComponent(y1); %>
    </hbj:inputField>
    <hbj:inputField id="imageWidth" visible="false" type="string" jsObjectNeeded="true">
        <% width = myContext.getParamIdForComponent(imageWidth); %>
    </hbj:inputField>
    <hbj:inputField id="imageHeight" visible="false" type="string" jsObjectNeeded="true">
        <% height = myContext.getParamIdForComponent(imageHeight); %>
    </hbj:inputField>
...
```

Tip

In the sample application it is not really important that the end user can see the selected coordinates of the image detail. Therefore the visibility of the fields is set to invisible.

8. Add a button to your JSP page in order to submit the form and send the data to your controller:

```
...
<hbj:button
    id="editFinishButton"
    onClick="onEditFinish"
    width="100"
    design="EMPHASIZED">
    <%
        editFinishButton.setText(res.getString("image.edit.image_edit_finish"));
        editFinishButton.setTooltip(res.getString("image.edit.image_edit_finish"));
    %>
 </hbj:button>
...
```

9. Add JavaScript scripting at the end of your JSP file (to retrieve and set the selected coordinates to invisible input fields):

```
<script language="JavaScript">
    function onEndCrop( coords, dimensions ) {
        <%=formId%>['<%=coordx1%>'].value = coords.x1;
        <%=formId%>['<%=coordy1%>'].value = coords.y1;
        <%=formId%>['<%=width%>'].value = dimensions.width;
        <%=formId%>['<%=height%>'].value = dimensions.height;
    }
    // example with a preview of crop results, must have minimumm dimensions
    Event.observe(
        window,
        'load',
        function() {
            new Cropper.ImgWithPreview(
                'testImage',
                {
                    minWidth: 100,
                    minHeight: 130,
                    ratioDim: { x: 100, y: 130 },
                    onEndCrop: onEndCrop,
                    previewWrap: 'previewArea'
                }
            )
        }
    );
</script>
```

Important

Make sure that the script is at the end of your JSP file as the IDs of the different fields have to be known first to make use of them in the script e.g. setting the current selection.

Note

The minimum width and height attribute specifies the minimum dimension the image detail can have. The ratioDim is the initial size of the image detail.

10. Save your changes.

5.5 Step 5: Retrieve the coordinates in your Controller and modify the image

1. Implement a method in your Controller for the onClick Event onEditFinish which reads the coordinates:

```
...
public void onEditFinish(Event event) {
    // set state to after edit as edit is finished
    state = AFTER_EDIT_STATE;

    // get the parameters from the ui
    InputField inputX1     = (InputField) this.getComponentByName("x1");
    InputField inputY1     = (InputField) this.getComponentByName("y1");
    InputField inputWidth  = (InputField) this.getComponentByName("imageWidth");
    InputField inputHeight = (InputField) this.getComponentByName("imageHeight");

     // get field values
    String x1     = inputX1.getValueAsDataType().getValueAsString();
    String y1     = inputY1.getValueAsDataType().getValueAsString();
    String width  = inputWidth.getValueAsDataType().getValueAsString();
    String height = inputHeight.getValueAsDataType().getValueAsString();
...
}
```

2. The ImageService in the par file provides some methods for modifiying the image. It is a sample implementation for editing the user profile picture in the UME. You can use this sample as a starting point for writing your own image editing class. Usage of class ImageService to edit picture:
    - saveImage() - stores picture in UME
    - deleteImage() - deletes picture in UME
    - checkImage() - checks whether it is a valid image or not
    - cropImage(int x, int y, int width, int height) - crops image by the given coordinates and dimension
    - scaleImage(int width, int height) - rescales the image by the given dimension

5.6 Step 6: Servlet for sending image data

As you have might seen the image in the edit page is set to an URL which points to a Servlet. This Servlet is intended to send the previously uploaded image to the JSP page. First of all the user

uploads a picture. If it is required to modify the picture it'll be stored in a session object. the <img src="
"> tag points to the Servlet URL. As soon as the page is rendered the Servlet is called, retrieves the image from the Session object and send it to the client.

1. Implement a Servlet which retrieves the picture from the Session object.
2. Send the picture as a mime stream to the client.

5.7 Step 7: Build and deploy the application

The last step is to deploy your application.

1. Build and deploy your application
2. Start the application within the SAP NetWeaver Developer Studio

6. Appendix

Appendix A – Links

JS Cropper UI

http://www.defusion.org.uk/code/javascript-image-cropper-ui-using-prototype-scriptaculous/

JS Cropper UI Demo page

http://www.defusion.org.uk/demos/060519/cropper.php

JS Cropper UI Download

http://www.defusion.org.uk/code/javascript-image-cropper-ui-using-prototype-scriptaculous/download-zip/

JS Cropper UI license

http://www.opensource.org/licenses/bsd-license.php

www.sdn.sap.com/irj/sdn/howtoguides