COMMON CONTROLS

Customizing the DefaultPainter

Adapting the appearance to your own corporate design

SCC Informationssysteme GmbH

Version 1.6 · As of: 15. Februar 2006

Impressum

PUBLISHED BY

SCC Informationssysteme GmbH
Drosselweg 13
64367 Mühltal
+49 6151 13 63 10
www.scc-gmbh.com

COPYRIGHT

Copyright © 2000–2026 SCC Informationssysteme GmbH. All rights reserved.

No part of this publication may be stored in a retrieval system, transmitted, or reproduced in any way without the prior agreement and written permission of SCC Informationssysteme GmbH.

TRADEMARKS

Java und JavaServer Pages sind Marken der Oracle Corporation in den USA und anderen Ländern. Microsoft und Microsoft Windows sind eingetragene Marken der Microsoft Corporation in den USA und anderen Ländern. Alle weiteren Produktnamen, Marken, Logos und Symbole können Marken oder eingetragene Marken ihrer jeweiligen Eigentümer sein.

Contents

1 Introduction

This document describes how the DefaultPainter that ships with the framework (DefPainterFactory) can be adapted to an alternative HTML style sheet, so that the control elements match your own corporate identity.

Beyond that, the appearance of the control elements can be changed to a limited degree by replacing a few graphics. DefaultPainter 2 (Def2PainterFactory) is an example: it drops the rounded shapes from the design of the control elements and lays out the header area of the form elements differently.

The adjustments described here call for no deeper knowledge of the framework architecture and its classes. That is only needed when the visual design of a control element has to be rebuilt from the ground up or when new functions are to be added; in such cases adapting the DefaultPainter is no longer enough and a new painter has to be developed — which is not the subject of this document.

Customising the DefaultPainter can proceed in the following steps:

- producing an HTML design based on the interface of the DefaultPainter;
- writing the derived PainterFactory class;
- creating a derived ResourceMap class that registers the graphics;
- generating the necessary cascading style sheets and the colour palette with the ResourceFactory tool;
- generating the buttons with the ResourceFactory tool.

2 Customising the DefaultPainter

2.1 Producing a colour scheme of your own

First the new design of the user interface is settled. Screenshots of the interface as the DefaultPainter produces it are a good starting point: by varying the colours, a new colour scheme for the application can be worked out step by step. The colour values arrived at then go into the style sheet of the new painter.

The ResourceFactory tool provides a style sheet generator that builds the style sheet files from templates. The colour values of the individual control elements are configured in the resource XML file of the ResourceFactory. Using properties keeps the number of separate colour values small: similar elements — headings, background colours and so on — can share one symbolic colour value and be changed together. Changes to the colour scheme can thus be made and tested quickly. The ResourceFactory tool is described in a document of its own.

XML

```xml
    <property name="col01"  value="#ffa510"/>
    ...
    <property name="col26"  value="#ffffff"/>

    <environment>

      <definitions code="error" name="Forms: Error form">
        <definitions code="color" name="Colortable">
          <color code="bg.header"       name="Background caption"   value="${col13}"/>
          <color code="bg.body"         name="Background body"      value="${col24}"/>
          <color code="border"          name="Border color"         value="${col13}"/>
          <color code="text"            name="Text color"           value="${col13}"/>
        </definitions>
      </definitions>

```

Excerpt from the resource XML file of the ResourceFactory

Besides the colours, individual graphical elements of the control elements — the corners of a list, for example — can be replaced. New graphics are first drawn by hand and then made known to the painter in a ResourceMap Java class. A ResourceMap defines all the graphics a painter needs for one particular design. Chapter 3 gives an overview of every graphic that can be replaced when adapting the DefaultPainter; how a ResourceMap is built is described in chapter 2.3.

2.2 Writing the PainterFactory class

Once the design of the application — its CSS files and images — is finished and the colour values and graphics are settled, the PainterFactory for the new painter is created. It is registered when the application starts and thereby determines the new appearance of the interface.

Its central task is to create the painters of the control elements, which in turn produce the actual HTML output.

JAVA

```java
import com.cc.framework.ui.painter.PainterFactory
```

```
import com.cc.framework.ui.painter.def.DefPainterFactory;
import com.cc.framework.ui.painter.html.HtmlPainterFactory;

public class MyFrontController extends ActionServlet {

    public void init() throws ServletException {
         super.init();
         // Register all Painter Factories with the preferred GUI-Design
         PainterFactory.registerApplicationPainter (
              getServletContext(), HtmlPainterFactory.instance());
         PainterFactory.registerApplicationPainter (
              getServletContext (), MyNewPainterFactory.instance());
    }
}

```

Code snippet 1: registering the new painter factory

A new PainterFactory is created by deriving from the existing DefPainterFactory class; alternatively the code fragment below can simply be copied. The passages highlighted in blue have to be adapted and are described afterwards.

Code fragment: MyNewPainterFactory

JAVA

```
package myPainterPackage;

import java.io.IOException;

import javax.servlet.jsp.JspWriter;

import com.cc.framework.common.Singleton;
import com.cc.framework.ui.painter.PainterFactory;
import com.cc.framework.ui.painter.ResourceMap;
import com.cc.framework.ui.painter.def.DefPainterFactory;

/**
 * Factory class for creating the MyNewPainterFactory.<br>
 * The MyNewPainterFactory renders the gui similarly to the DefPainter but
 * substitutes the stylesheet and some images.
*/
public final class MyNewPainterFactory extends DefPainterFactory implements Singleton {

    /**
     * Base directory used for resource by this Painterfactory
     */
    public static final String RESOURCE_DIR = "myDef/";

    /**
     * The single instance of this class
     */
    private static MyNewPainterFactory instance = new MyNewPainterFactory ();

    /**
     * Constructor
     */
    protected MyNewPainterFactory () {
        super();
    }

```

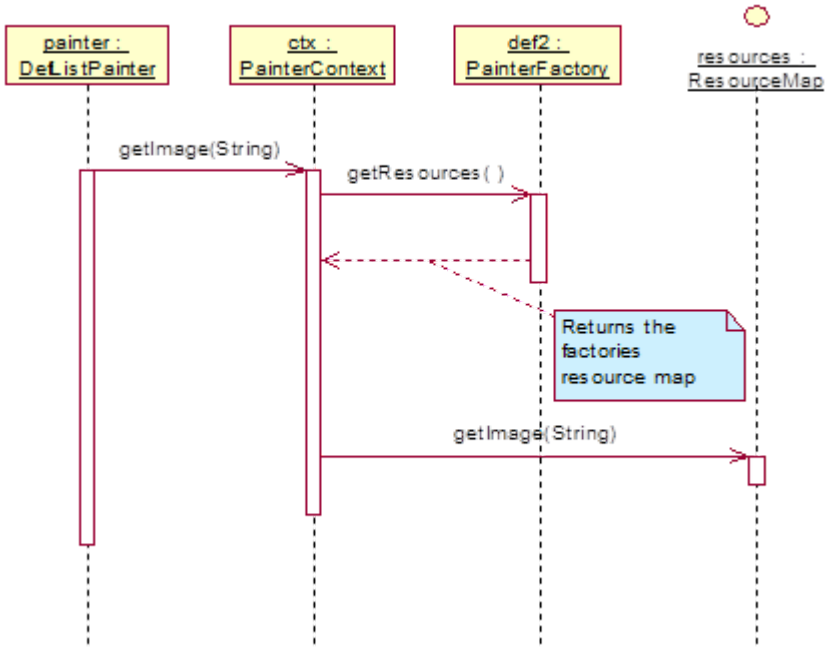
```
    /**
     * @see com.cc.framework.ui.painter.PainterFactory#createResourceMap()
     */
    protected ResourceMap createResourceMap() {
        return new MyNewPainterResourceMap();
    }

    /**
     * Returns the unique Id for this Painterfactory
     * @return  The unique Id for this Painterfactory which is "def"
     */
    public String getFactoryId() {
        return "MyDef";
    }

    /**
     * Returns the base directory used for resource by the Painterfactory
     * @return  The web resource directory
     */
    public String getResourceDir() {
        return RESOURCE_DIR;
    }

    /**
     * Returns the single instance of the class (singleton)
     * @return  The single instance  of this PainterFactory
     */
    public static PainterFactory instance() {
        return instance;
    }

    /**
     * @see com.cc.framework.ui.painter.PainterFactory#doCreateHeaderIncludes(JspWriter)
     */
    protected void doCreateHeaderIncludes(JspWriter writer) throws IOException {

        StringBuffer buf = new StringBuffer();

        buf
            .append("<!-- BEGIN Framework (MyDefPainter) -->")
            .append("<link rel='stylesheet' href='")
            .append(RESOURCE_DIR)
            .append("style/default.css' charset='ISO-8859-1' type='text/css'>");

        buf
            .append("<script language='JavaScript' src='")
            .append(RESOURCE_DIR)
            .append("jscript/functions.js'></script>")

            .append("<script language='JavaScript' src='")
            .append(RESOURCE_DIR)
            .append("jscript/controls.js'></script>")
            .append("<script language='JavaScript' src='")
            .append(RESOURCE_DIR).append("jscript/tabset.js'></script>")
            .append("<!-- END -->");

        // write the buffer
        writer.println();
        writer.println(buf);
        writer.println();
    }

}
```

To create a new PainterFactory class, the following parts of the code fragment have to be adapted:

Section	Description
RESOURCE_DIR	Sets the directory in which the new resources — style sheets and images — are found. For the DefaultPainter, for example, the directory “fw/def” is set.
createResourceMap()	The method has to return an instance of a ResourceMap that registers the new graphics. Creating the ResourceMap is described in chapter 2.3.
getFactoryId()	The method has to return a symbolic name for the painter factory. For the DefaultPainter it returns “def”; that abbreviation should be reserved for the Common Controls framework.
doCreateHeaderIncludes	This method serves to include the necessary style sheets and JavaScript blocks at the top of a JSP page. A JavaScript file needed on one single page only should not be added here. The <util:jsp> tag at the top of a JSP page writes the string generated here into the resulting HTML page (see the Common Controls Quickstart, <util:jsp directive="includes"/>).

2.3 Creating a ResourceMap

A ResourceMap defines all the graphics and colours a painter needs for one particular interface design. The UML sequence diagram below shows how a painter reaches an image through the ResourceMap.



A ResourceMap is derived from the class com.cc.framework.ui.painter.def.DefResourceMap. The advantage is that only the new graphics have to be defined, not all of them.

The graphics are registered in the method doRegisterImages(). Since the existing DefaultPainter is being adapted, the symbolic names of the graphics are already fixed (see chapter 3); all that happens here is a mapping onto the file in the resource directory.

A single resource — an image — is registered with a call to registerImage(...).

registerImage(String resourceId, ImageModel model)	Registers an image for an existing internal resource id.
registerImage(int size, String resourceId, ImageModel model)	Registers an image for an existing internal resource id. The size argument distinguishes graphics that have to be provided in several sizes. This matters for the TreeControl and the TreeListControl, for example: the TreeControl uses images of 15x15 for folder and node symbols, whereas the TreeListControl needs them at 20x20. By stating the size, the painter reaches the right collection. The argument serves internal bookkeeping only.

When writing a new ResourceMap it is best to take the DefResourceMap of the DefaultPainter as a model and change the names and dimensions of the images as needed. The example below shows the ResourceMap of DefaultPainter 2.

JAVA

```java
package com.cc.framework.ui.painter.def2;

import com.cc.framework.ui.Color;
import com.cc.framework.ui.model.ImageModel;
import com.cc.framework.ui.model.imp.ImageModelImp;
import com.cc.framework.ui.painter.def.DefResourceMap;

/**
 * Resourcemap for the Def2ResourceMap.
 * Defines resources like images used by the Def2ResourceMap.
*/
public class Def2ResourceMap extends DefResourceMap {

   /**
    * Constructor
    */
   public Def2ResourceMap () {
       super();
   }

   /**
    * @see com.cc.framework.ui.painter.ResourceMapImp#doRegisterImages()
    */
   protected void doRegisterImages() {
       super.doRegisterImages();

       // define the resources to use by this painter
       registerImage(IMAGE_DOT_COLOR,        createImage("dots/dot_{0}.gif",  5, 5));
       registerImage(IMAGE_MAGNIFIER,        createImage("magnifier.gif",     20, 23));
       registerImage(IMAGE_HEADER_TOP,       createImage("headertop.gif",     9, 17));
       registerImage(IMAGE_HEADER_BOTTOM,    createImage("headerbottom.gif",  9, 17));

       // icons
       registerImage(ICON_ADD,               createImage("icons/add.gif",     16, 15));
       registerImage(ICON_EDIT,              createImage("icons/edit.gif",    16, 15));
```

```
        registerImage(ICON_DELETE,            createImage("icons/delete.gif",  16, 15));
        registerImage(ICON_SELECT,            createImage("icons/select.gif",  21, 16));


        // corner
        registerImage(CORNER_TABLE_LEFT,         createImage("corners/tl.gif", 10, 17));
        registerImage(CORNER_TABLE_RIGHT,        createImage("corners/tr.gif", 10, 17));
        registerImage(CORNER_FORM_LEFT,          createImage("corners/fl.gif", 17, 20));
        registerImage(CORNER_FORM_RIGHT,         createImage("corners/fr.gif", 80, 20));
        registerImage(CORNER_FORMSEARCH_LEFT,    createImage("corners/tl.gif", 15, 20));
        registerImage(CORNER_FORMSEARCH_RIGHT,   createImage("corners/fr.gif", 80, 20));

        // buttons listcontrol
        registerImage(BUTTON_NEXT_1,      createImage("buttons/btnNext1.gif",   15, 15));
        registerImage(BUTTON_NEXT_2,      createImage("buttons/btnNext2.gif",   15, 15));
        registerImage(BUTTON_FIRST_1,     createImage("buttons/btnFirst1.gif",  15, 15));
        registerImage(BUTTON_FIRST_2,     createImage("buttons/btnFirst2.gif",  15, 15));
        registerImage(BUTTON_LAST_1,      createImage("buttons/btnLast1.gif",   15, 15));
        registerImage(BUTTON_LAST_2,      createImage("buttons/btnLast2.gif",   15, 15));
        registerImage(BUTTON_PREVIOUS_1,  createImage("buttons/btnPrev1.gif",   15, 15));
        registerImage(BUTTON_PREVIOUS_2,  createImage("buttons/btnPrev2.gif",   15, 15));
        registerImage(BUTTON_CREATE_1,    createImage("buttons/btnCreate1.gif", 15, 15));
        registerImage(BUTTON_REFRESH_1,   createImage("buttons/btnRefresh1.gif" 15, 15));

        // Tabset
        registerImage(TABSET_BACKGROUND,      createImage("tab/tab.gif",         1, 19));
        registerImage(TABSET_TABSEL_L_COLOR,  createImage("tab/tabLSel_{0}.gif", 10, 19));
        registerImage(TABSET_TABSEL_R_COLOR,  createImage("tab/tabRSel_{0}.gif", 11, 19));
        registerImage(TABSET_TABSEL_BG_COLOR,createImage("tab/tabBgSel_{0}.gif", 1, 19));
        registerImage(TABSET_TABUNSEL_L,      createImage("tab/tabL.gif", 8, 19));
        registerImage(TABSET_TABUNSEL_R,      createImage("tab/tabR.gif", 10, 19));
        registerImage(TABSET_TABUNSEL_BG,     createImage("tab/tabBg.gif",      1, 19));
        registerImage(TABSET_TABDISABLED_L,   createImage("tab/tabDisL.gif",    8, 19));
        registerImage(TABSET_TABDISABLED_R,   createImage("tab/tabDisR.gif",    10, 19));
        registerImage(TABSET_TABDISABLED_BG,  createImage("tab/tabDisBg.gif",   1, 19));


        // 15 Pixel images
        registerImage(15, TREE_FOLDEROPEN,
                createImage("tree/15/folderOpen.gif",             15, 15));
        registerImage(15, TREE_FOLDERCLOSED,
                createImage("tree/15/folderClosed.gif",           15, 15));
        registerImage(15, TREE_ITEM,            createImage("tree/15/item.gif",  15, 15));
        registerImage(15, TREE_STRUCTURE,       createImage("tree/15/0.gif",     15, 15));
        registerImage(15, TREE_STRUCTURE_2,     createImage("tree/15/2.gif",     15, 15));
        registerImage(15, TREE_STRUCTURE_10,    createImage("tree/15/10.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_12,    createImage("tree/15/12.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_14,    createImage("tree/15/14.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_16,    createImage("tree/15/16.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_18,    createImage("tree/15/18.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_26,    createImage("tree/15/26.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_30,    createImage("tree/15/30.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_32,    createImage("tree/15/32.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_34,    createImage("tree/15/34.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_42,    createImage("tree/15/42.gif",    15, 15));
        registerImage(15, TREE_STRUCTURE_46,    createImage("tree/15/46.gif",    15, 15));

        // checkboxen
        registerImage(15, CHECKBOX_INVALID,
                createImage("check/15/cb.gif",     15, 15));
        registerImage(15, CHECKBOX_UNCHECKED,
                createImage("check/15/cb0.gif", 15, 15));
        registerImage(15, CHECKBOX_CHECKED,
                createImage("check/15/cb1.gif", 15, 15));
```

```
        registerImage(15, CHECKBOX_INDETERMINATE,
                createImage("check/15/cb2.gif", 15, 15));

    }

    /**
     * @see com.cc.framework.ui.painter.ResourceMapImp#doRegisterColors()
     */
    protected void doRegisterColors() {
        // We are using the Def2ColorPalette Object
        // so we don't have to register any colors explicitly
        // The Def2ColorPalette Object was created by the
        // Resourcefactory Tool
        setColorPalette(new Def2ColorPalette());
    }

    /**
     * Creates a image model
     *
     * @param src     The image path relative to the painter
     * @param width   The width  of the image
     * @param height  The height of the image
     * @return   Image      The image model
     */
    private ImageModel createImage(String src, int width, int height) {
        StringBuffer fullPath = new StringBuffer()
         .append(Def2PainterFactory.RESOURCE_DIR)
         .append("image/")
         .append(src);

        return new ImageModelImp(fullPath.toString(), width, height);
    }
}

```

Naming resources that a painter needs in several colours is a special case. The TabSetControl is an example: it may contain nested tab sets, and because the background colour changes, different images have to be produced for the selected tabs.

So that not every graphic has to be defined for every colour in the ResourceMap, the notation tabLSel_{0}.gif is used: it tells the painter to substitute the colour value configured in the control element for the expression {0}.

Note: the background colour of a tab set is given in the bgcolor attribute when the tab set is declared on the JSP page. If the attribute is omitted, the colour #EFEFEF is used by default.

Example: the outer tab set has the background colour #EFEFEF, the nested one #DADFE0. For the selected states the tab set painter therefore needs the following graphics:

CODE

```
                tabLSel_EFEFEF.gif, tabBgSel_EFEFEF.gif, tabRSel_EFEFEF.gif
                tabLSel_DADFE0.gif, tabBgSel_DADFE0.gif, tabRSel_DADFE0.gif
```

The images are nevertheless registered only as tabLSel_{0}.gif, tabBgSel_{0}.gif and tabRSel_{0}.gif.

Graphics are registered with the method registerImage(..), which assigns an ImageModel — an image resource — to a given resource id.

The ImageModel itself is created by calling createImage(String src, int width, int height). That method prefixes the root directory in which the graphics are kept; in the example above they are looked for under myDef/image/.

If the graphics are to be kept elsewhere, this is the place to change the directory path.

Note: where buttons use hover effects, be sure to register the image ending in xxx1.gif — the one representing the normal state of the button.

The colour palette registered in the method doRegisterColors() is likewise generated by the ResourceFactory tool. The resulting Java class is simply placed in the directory of the new painter factory. The new painter thus consists of the following classes.

CODE

```
com.myapp.ui.painter.MyColorPalette     (generiert von dem ResourceFactory Tool)
com.myapp.ui.painter.MyPainterFactory
com.myapp.ui.painter.MyResourceMap

```

Code snippet 2: the classes of the adapted painter at a glance

2.4 Generating style sheets

Generating the style sheets for an adapted DefaultPainter is supported by the ResourceFactory from version 1.1 onwards. The environment section contains style definitions that refer to a globally defined colour table; within those definitions the colours can be stated directly, or only the globally defined values are overridden.

Once the colour values have been entered, the style sheets are produced by the Ant task “build-res” and the resulting files are copied into the web resources of the application.

XML

```
<resource-factory version="1.1">

    <!--
    use color macros to reduce the number of
    different color values so it's more easy
    to change the entire stylesheet
    -->

    <property name="col00"  value="#000000"/>
    <property name="col01"  value="#0000ff"/>
    <property name="col02"  value="#005a6b"/>
    <property name="col03"  value="#80adba"/>
    <property name="col04"  value="#84adbd"/>
    <property name="col05"  value="#87b1ba"/>
    <property name="col06"  value="#8d9da1"/>
    <property name="col07"  value="#a5c4cb"/>
    <property name="col08"  value="#a7c2c7"/>
    <property name="col09"  value="#b4ced4"/>
    <property name="col10"  value="#bdbdbd"/>
    <property name="col11"  value="#c1d6db"/>
    <property name="col12"  value="#c4c8c9"/>
    <property name="col13"  value="#c7003c"/>
    <property name="col14"  value="#cecece"/>
    <property name="col15"  value="#dadfe0"/>
    <property name="col16"  value="#dce8eb"/>
```

```
    <property name="col17"  value="#edeff0"/>
    <property name="col18"  value="#efefef"/>
    <property name="col19"  value="#f3f4f5"/>
    <property name="col20"  value="#f57e17"/>
    <property name="col21"  value="#fae4c2"/>
    <property name="col22"  value="#fea217"/>
    <property name="col23"  value="#ffa510"/>
    <property name="col24"  value="#ffd3d3"/>
    <property name="col25"  value="#ffffe1"/>
    <property name="col26"  value="#ffffff"/>

    <environment>
        <definitions code="form" name="Forms: Formelements">
            <definitions code="color" name="Colortable">
                <color code="bg.field"
                       name="Background-Color Field"
                       value="${col18}"/>
                <color code="bg.header"
                       name="Background-Color Form-Caption "
                       value="${col02}"/>
                <color code="bg.label"
                       name="Background-Color Label"
                       value="${col15}"/>
                <color code="bg.section"
                       name="Background-Color Section"
                       value="${col08}"/>
                <color code="bg"
                       name="Hintergrundfarbe"
                       value="${col15}"/>
                <color code="border.item"
                       name="Rahmen einer Zeile"
                       value="${col10}"/>
                <color code="border.section"
                       name="Rahmen Gruppenüberschrift"
                       value="${col02}"/>
                <color code="border"
                       name="Fensterrahmen"
                       value="${col02}"/>
                <color code="text.caption"
                       name="Textfarbe Überschriftsbereich"
                       value="${col26}"/>
                <color code="text.detail"
                       name="Textfarbe Labelbereich"
                       value="${col26}"/>
                <color code="text.header"
                       name="Textfarbe Hauptüberschrift"
                       value="${col26}"/>
                <color code="text.section"
                       name="Textfarbe Detailüberschrift"
                       value="${col02}"/>
            </definitions>
        </definitions>

        <definitions code="error" name="Forms: Fehlerformular">
            <definitions code="color" name="Farbtabelle">
                <color code="bg.header"
                       name="Hintergrund Überschriftsbereich"
                       value="${col13}"/>
                <color code="bg.body"
                       name="Hintergrund Body Bereich"
                       value="${col24}"/>
                <color code="border"
                       name="Rahmenfarbe"
                       value="${col13}"/>
```

```
                    <color code="text"
                           name="Textfarbe"
                           value="${col13}"/>
                </definitions>
            </definitions>


            .......

        </environment>
```

Code snippet: configuration file of the ResourceFactory (excerpt)

3 Graphics of the DefaultPainter that can be customised

This chapter shows which graphics of a control element drawn by the DefaultPainter can be changed.

3.1 ListControl

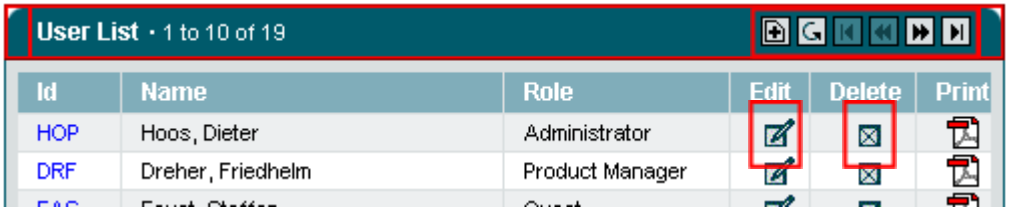
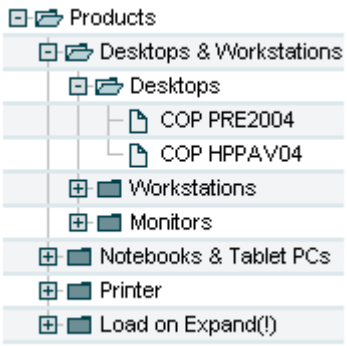


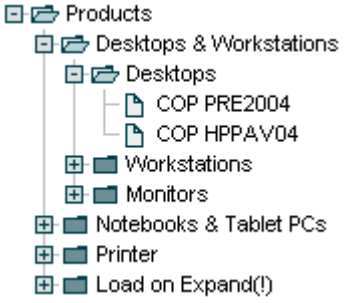
Figure 2: Replaceable graphics — ListControl

Image	ResourceId	Example	Width	Height
Left corner	CORNER_TABLE_LEFT	corners/l.gif	10	17
Right corner	CORNER_TABLE_RIGHT	corners/r.gif	10	17
Buttons in the table header				
New button	BUTTON_CREATE_1	buttons/btnCreate1.gif	15	15
Refresh button	BUTTON_REFRESH_1	btnRefresh1.gif	15	15
First button (active)	BUTTON_FIRST_1	buttons/btnFirst1.gif	15	15
First button (inactive)	BUTTON_FIRST_2	buttons/btnFirst2.gif	15	15
Previous button (active)	BUTTON_PREVIOUS_1	buttons/btnPrev1.gif	15	15
Previous button (inactive)	BUTTON_PREVIOUS_2	buttons/btnPrev2.gif	15	15
Next button (active)	BUTTON_NEXT_1	buttons/btnNext1.gif	15	15
Next button (inactive)	BUTTON_NEXT_2	buttons/btnNext2.gif	15	15
Last button (active)	BUTTON_LAST_1	buttons/btnLast1.gif	15	15
Last button (inactive)	BUTTON_LAST_2	buttons/btnLast2.gif	15	15

Image	ResourceId	Example	Width	Height
Icons of the edit, delete and check columns				
Edit icon	ICON_EDIT	icons/edit.gif	16	16
Delete icon	ICON_DELETE	icons/delete.gif	16	16
Check column (unchecked)	ICON_CHECK	icons/check.gif	16	16
Check column (checked)	ICON_CHECKED	icons/checked.gif	16	16
Icons for sorting the columns				
Sortable icon	BUTTON_SORTABLE	buttons/btnSortable1.gif	11	13
Ascending sort icon	BUTTON_SORTASC	buttons/btnSortUp1.gif	11	13
Descending sort icon	BUTTON_SORTDESC	buttons/btnSortDown1.gif	11	13

3.2 TreeControl






```
<style>
.tc .tlodd TD {
border-bottom: none;
}
/* even rows */
.tc .tleven TR {
font-family: Arial;
font-size: 8pt;
font-weight: normal;
background-color: white;
border: none;
}
.tc .tleven TD {
border-bottom: none;
}
</style>
```

In the TreeControl the graphics for the nodes and the connecting lines can be replaced.

Image	ResourceId	Example	Width	Height
	TREE_STRUCTURE	def/image/tree/15/0.gif	15	15
	TREE_STRUCTURE_10	def/image/tree/15/10.gif	15	15
	TREE_STRUCTURE_12	def/image/tree/15/12.gif	15	15
	TREE_STRUCTURE_14	def/image/tree/15/14.gif	15	15
	TREE_STRUCTURE_16	def/image/tree/15/16.gif	15	15

Image	ResourceId	Example	Width	Height
	TREE_STRUCTURE_18	def/image/tree/15/18.gif	15	15
	TREE_STRUCTURE_2	def/image/tree/15/2.gif	15	15
	TREE_STRUCTURE_26	def/image/tree/15/26.gif	15	15
	TREE_STRUCTURE_30	def/image/tree/15/30.gif	15	15
	TREE_STRUCTURE_32	def/image/tree/15/32.gif	15	15
	TREE_STRUCTURE_34	def/image/tree/15/34.gif	15	15
	TREE_STRUCTURE_42	def/image/tree/15/42.gif	15	15
	TREE_STRUCTURE_46	def/image/tree/15/46.gif	15	15
	TREE_FOLDERCLOSED	def/image/tree/15/folderClosed.gif	15	15
	TREE_FOLDEROPEN	def/image/tree/15/folderOpen.gif	15	15
	TREE_ITEM	def/image/tree/15/Item.gif	15	15

Note:

The graphics for the open and the closed folder, and the one for a leaf element, can also be changed on the JSP page by means of an image map.

Naming convention for the images of the tree structure:

The name of each image of the tree structure is a bit-coded decimal number.

Hex	Dec	Meaning
0x20	32	Plus sign in front of the item
0x10	16	Minus sign in front of the item
0x08	8	Connecting line to the north
0x04	4	Connecting line to the south
0x02	2	Connecting line to the east
0x01	1	Connecting line to the west

3.3 TreeListControl

Figure 3: Replaceable graphics — TreeListControl

Image	ResourceId	Example	Width	Height
Left corner	CORNER_TABLE_LEFT	corners/l.gif	10	17
Right corner	CORNER_TABLE_RIGHT	corners/r.gif	10	17
Buttons in the table header				
New button	BUTTON_CREATE_1	buttons/btnCreate1.gif	15	15
Refresh button	BUTTON_REFRESH_1	btnRefresh1.gif	15	15
Icons of the add, edit, delete and check columns				
Add icon	ICON_ADD	icons/add.gif	16	16
Edit icon	ICON_EDIT	icons/edit.gif	16	16
Delete icon	ICON_DELETE	icons/delete.gif	16	16
Check column (unchecked)	ICON_CHECK	icons/check.gif	16	16
Check column (checked)	ICON_CHECKED	icons/checked.gif	16	16
Icons for sorting the columns				
Sortable icon	BUTTON_SORTABLE	buttons/btnSortable1.gif	11	13
Ascending sort icon	BUTTON_SORTASC	buttons/btnSortUp1.gif	11	13
Descending sort icon	BUTTON_SORTDESC	buttons/btnSortDown1.gif	11	13

3.4 TabSet

Figure 4: Replaceable graphics — TabSetControl

Image	ResourceId	Example	Width	Height
Background	TABSET_BACKGROUND	tab/tab.gif	1	19
Selected tab, left corner	TABSET_TABSEL_L_COLOR	tab/tabLSel_{0}.gif	10	19
Selected tab, background	TABSET_TABSEL_BG_COLOR	tab/tabRSel_{0}.gif	1	19
Selected tab, right corner	TABSET_TABSEL_R_COLOR	tab/tabBgSel_{0}.gif	11	19
Unselected tab, left corner	TABSET_TABUNSEL_L	tab/tabL.gif	8	19
Unselected tab, background	TABSET_TABUNSEL_BG	tab/tabBg.gif	1	19
Unselected tab, right corner	TABSET_TABUNSEL_R	tab/tabR.gif	10	19
Disabled tab, left corner	TABSET_TABDISABLED_L	tab/tabDisL.gif	8	19
Disabled tab, background	TABSET_TABDISABLED_BG	tab/tabDisBg.gif	1	19
Disabled tab, right corner	TABSET_TABDISABLED_R	tab/tabDisR.gif	10	19

3.5 Tabbar

Figure 5: Replaceable graphics — TabBarControl

Image	ResourceId	Example	Width	Height
Background	TABBAR_BACKGROUND	tab/tab.gif	1	19
Selected tab, left corner	TABBAR_TABSEL_L_COLOR	tab/tabLSel_{0}.gif	10	19
Selected tab, background	TABBAR_TABSEL_L_COLOR	tab/tabRSel_{0}.gif	1	19
Selected tab, right corner	TABBAR_TABSEL_R_COLOR	tab/tabBgSel_{0}.gif	11	19
Unselected tab, left corner	TABBAR_TABUNSEL_L	tab/tabL.gif	8	19
Unselected tab, background	TABBAR_TABUNSEL_BG	tab/tabBg.gif	1	19
Unselected tab, right corner	TABBAR_TABUNSEL_R	tab/tabR.gif	10	19
Disabled tab, left corner	TABBAR_TABDISABLED_L	tab/tabDisL.gif	8	19
Disabled tab, background	TABBAR_TABDISABLED_BG	tab/tabDisBg.gif	1	19
Disabled tab, right corner	TABBAR_TABDISABLED_R	tab/tabDisR.gif	10	19

3.6 BreadCrumbs

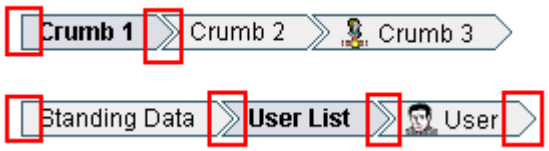


Figure 6: Replaceable graphics — BreadCrumbControl

Image	ResourceId	Example	Width	Height
	CRUMBS_UNSEL_BG	crumbs/u_bg.gif	1	18
	CRUMBS_UNSEL_NONE	crumbs/u_.gif	19	18
	CRUMBS_UNSEL_UNSEL	crumbs/uu.gif	27	18
	CRUMBS_UNSEL_SEL	crumbs/us.gif	27	18
	CRUMBS_SEL_BG	crumbs/s_bg.gif	1	18
	CRUMBS_SEL_NONE	crumbs/s_.gif	19	18
	CRUMBS_SEL_UNSEL	crumbs/su.gif	27	18
	CRUMBS_SEL_SEL	crumbs/ss.gif	27	18
	CRUMBS_NONE_SEL	crumbs/_s.gif	9	18
	CRUMBS_NONE_UNSEL	crumbs/_u.gif	9	18

3.7 Forms

3.7.1 Standard form

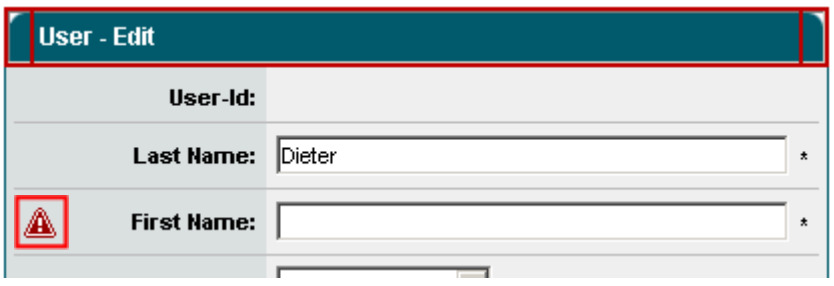


Figure 7: Replaceable graphics — form

Image	ResourceId	Example	Width	Height
Left corner	CORNER_FORM_LEFT	corners/l.gif	10	17
Right corner	CORNER_FORM_RIGHT	corners/r.gif	10	17
Error marker				
Error marker	IMAGE_ERROR_INPUT	errInput.gif	16	16

3.7.2 Search dialog

Figure 8: Replaceable graphics — search form

Image	ResourceId	Example	Width	Height
Left corner	CORNER_FORMSEARCH_LEFT	corners/l.gif	10	17
Magnifier image	IMAGE_MAGNIFIER	magnifier.gif	20	20
Right corner	CORNER_FORMSEARCH_RIGHT	corners/r.gif	10	17

3.7.3 Message dialog

Figure 9: Replaceable graphics — message dialogue

Image	ResourceId	Example	Width	Height
Left corner	CORNER_FORM_LEFT_COLOR	corners/l_{0}.gif	10	17
Info image	IMAGE_INFORMATION	info.gif	14	23
Right corner	CORNER_FORM_RIGHT_COLOR	corners/r_{0}.gif	10	17
Bullet				
Bullet	IMAGE_DOT_COLOR	dots/dot_{0}.gif	5	5

3.7.4 Error dialog

Figure 10: Replaceable graphics — error dialogue

Image	ResourceId	Example	Width	Height
Left corner	CORNER_FORM_LEFT_COLOR	corners/l_{0}.gif	10	17
Error image	IMAGE_ERROR	error.gif	14	23
Right corner	CORNER_FORM_RIGHT_COLOR	corners/r_{0}.gif	10	17
Bullet				
Bullet	IMAGE_DOT_COLOR	dots/dot_{0}.gif	5	5

3.8 HeadlineControl

The colours of the headline control element are adjusted in the style sheet.

4 Support

If you have questions or run into trouble, we are glad to help. Please use the service form on our website; we endeavour to answer enquiries promptly.