Common-Controls Individualizing the DefaultPainter

Version 1.5.0 - Stand: 30. Januar 2005

Common Controls

Published by:
SCC Informationssysteme GmbH
64367 Mühltal

Tel: +49 (0) 6151 / 13 6 31 0
Internet www.scc-gmbh.com

Product Site
http://www.common-controls.com

Table of content

1 Introduction 1

2 Individualizing the DefaultPainter 2
2.1 Creation of your own color design 2
2.2 Generation of the PainterFactory 2
2.3 Generating a ResourceMap 5
2.4 Generating StyleSheets 8

3 Modifiable images DefaultPainters 10
3.1 ListControl 10
3.2 TreeControl 11
3.3 TreeListControl 12
3.4 TabSet 13
3.5 Tabbar 14
3.6 BreadCrumbs 15
3.7 Forms 16
3.8 HeadLine Control 17

4 Support 18

1 Introduction

This document describes how the DefaultPainter that is part of the supplied kit can be customized to an alternative HTML Stylesheet. This facilitates a customization of the control elements to the (company's own) Coorperate Identity.

In addition, the appearance of the control elements can be changed to a small extent by replacement of a few graphics. One example of this is the DefaultPainter2 (Def2Painter), which does not use any roundings in the layout of the control elements and also formats the header area of the form elements differently.

The customizations described here do not require any in-depth knowledge of the Framework architecture and classes. This is only required when the visual design of a control element has to be fundamentally altered or new functions have to be integrated. In such cases, a customization of the DefaultPainter is not sufficient any more, and a new Painter has to be developed. However, that is not the objective of this document.

When individualizing the DefaultPainter, you need to follow the steps below:

- Creation of an HTML design on the basis of the interface of the DefaultPainter
- Generation of the PainterFactory class
- Generation of a ResourceMap (registration of graphics) class
- Generation of the necessary Cascading Stylesheets and ColorPalette with the ResourceFactory Tool from Version 1.1 onwards.
- Generation of buttons

2 Individualizing the DefaultPainter

2.1 Creation of your own color design

To start with, the new design of the user interface is defined. In doing so, you can fall back on the existing screenshots of the interface has generated by the DefaultPainter. Then, by varying the colors, a new color scheme can be worked out for the application step-by-step. The color values so obtained are incorporated in the Stylesheet of the new Painter.

The **ResourceFactory**, from version 1.1 onwards, provides a Stylesheet Generator which generates the Stylesheet files with the help of templates. For this purpose, the color values for the individual control elements are configured in the resources file of the PainterFactory. By using Properties, the number of individual color values here can be reduced and similar elements (titles, background colors etc.) can be populated with the same symbolic color value and modified. With the ResourceFactory, therefore, changes in the color scheme can be quickly implemented and tested.

```
<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>
```

Figure 1: Extract from resource file ResourceFactory

Apart from colors, individual graphics elements such as the corners of lists can also be replaced in the control elements. New graphics are declared to the Painter in a `ResourceMap`. A ResourceMap defines, for a Painter, all the necessary graphics that it needs for a particular design. In Chapter 3, there is an overview of all the graphics that can be replaced in the course of the customization of the DefaultPainter. The structure of the ResourceMap is described in Chapter 0.

2.2 Generation of the PainterFactory

After the design for the application has been determined and the color values and graphics are laid down, the **PainterFactory** for the new Painter is generated. The PainterFactory is subsequently registered at the time of starting of the application and thus determines the (new) design of the interface.

```
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());
    }
}
```

CodeSnippet 1: Registering the new Painterfactory

Individualizing the DefaultPainters

The generation of a new PainterFactory is done by derivation from the existing `DefPainterFactory` class. Alternatively, the following code fragment can be copied. The places highlighted in blue have to be individually customized and are described later.

Code Fragment - MyNewPainterFactory

```
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;
    }
```

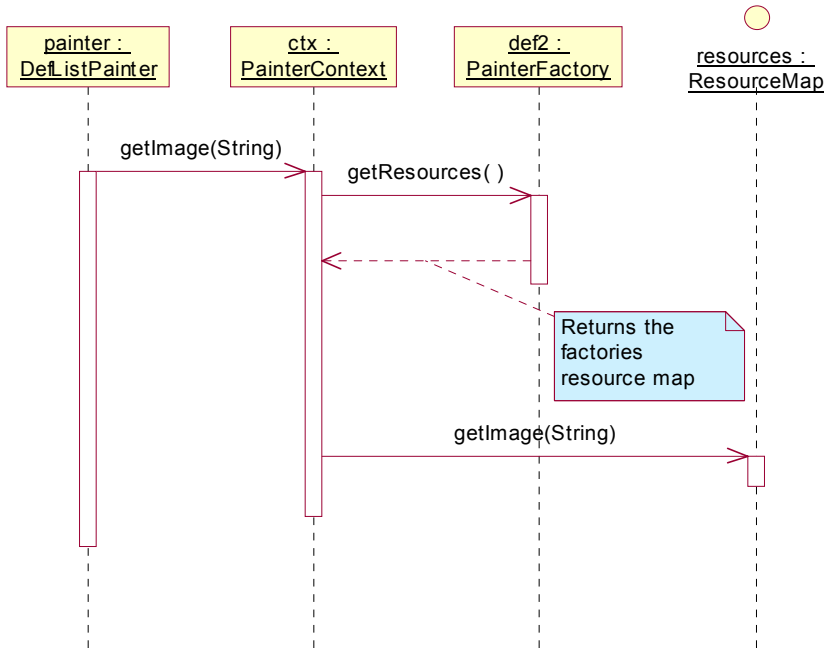
```
    /**
     * @see com.cc.framework.ui.painter.PainterFactory#doCreateHeaderIncludes(JspWriter)
     */
    protected void doCreateHeaderIncludes(JspWriter writer) throws IOException {

        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();
    }

}
```

For generating a new PainterFactory class, the following sections from the code fragment must be customized:

Section	Description
`RESOURCE_DIR`	Defines the directory in which the new resources like StyleSheets and images can be found. E.g. for the DefaultPainter, the directory “fw/def” is set.
`createResourceMap()`	The method must return an instance of a ResourceMap, which registers the new graphics. The generation of the ResourceMap is described in Chapter 0.
`getFactoryId()`	The method must return a symbolic name for the PainterFactory. The call to the method of the DefaultPainter returns, for example, “def”. The abbreviation “def” is to remain reserved for the Common Controls - Framework.
`doCreateHeaderIncludes`	The method serves to include the necessary Stylesheets and JavaScript blocks at the beginning of a JSP page. A JavaScript file that is only required on a single page should not be included here. The <util:jsp> tag at the start of a JSP page writes the string generated here into the generated HTML page (compare Common Controls - FAQs Quickstart `<util:jsp directive="includes"/>`).

2.3 Generating a ResourceMap

A ResourceMap defines, for a Painter, all the necessary graphics and colors that it needs for a particular design of the interface. The following UML sequence diagram shows how a Painter accesses an image via the ResourceMap.



A ResourceMap is derived from the class `com.cc.Framework.ui.painter.def.DefResourceMap` . This has the advantage that not all the graphics, but only the new ones have to be determined.

The registration of the graphics takes place in the method **`doRegisterImages()`**. Since the existing DefaultPainter is customized, the symbolic names of the graphics are already defined (see Chapter 3). Only a mapping to the file in the resource directory takes place here.

A single resource (image) is registered through the call **`registerImage(...)`** .

`registerImage(`
`  String resourceId,`
`  ImageModel model)`
	Registers an image for an existing internal ResourceId.

`registerImage(`
`  int size,`
`  String resourceId,`
`  ImageModel model)`
	Registers an image for an existing internal ResourceId.
	The `size` argument is used to distinguish graphics that have to be made available in different sizes. This is relevant, e.g., in for the Tree control and the TreeListControl. Whereas the TreeControl uses images in the size 15x15 for folder or node symbols, the TreeListControl requires the graphics in the size 20x20. With the specification of the size, the Painter can now access the corresponding collection. It is used here only for internal administration.

When creating a new `ResourceMap`, It is best to orient yourself according to the `DefResourceMap` of the DefaultPainter and change the names and dimensions of images as required in your own class. The following example shows the ResourceMap of the DefaultPainter2.

```
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);
    }
}
```

There is one special feature with regard to the nomenclature of resources that a Painter requires in different color values. An example of this is the TabSetControl, which can contain sub-ordinate Tabsets. Owing to the changing background color, different images have to be prepared for the selected tabs.

But in order that within the ResourceMap, all the graphics do not have to be defined for all the different color values, we fall back on the notation `tabLSel_{0}.gif` . This states that the Painter should use the color value that has been configured in the control element for the expression `{0}`.

Note: The background color of a TabSet is saved at the time of declaration of the TabSet in the JSP page in the `bgcolor`-attribute. If the attribute is not specified, then by default, the color value `#EFEFEF` is used.

Example: The first Tabset has the background color `#EFEFEF`. The nested Tabset, the color `#DADFE0`. The Tabset-Painter requires the following graphics for the selected states:
`tabLSel_EFEFEF.gif, tabBgSel_EFEFEF.gif, tabRSel_EFEFEF.gif`
`tabLSel_DADFE0.gif, tabBgSel_DADFE0.gif, tabRSel_DADFE0.gif`

Nonetheless however, the images are only registered as `tabLSel_{0}.gif`, `tabBgSel_{0}.gif` and `tabRSel_{0}.gif` .

The registration of graphics is done with the method `registerImage(..)`. This assigns an ImageModel (image resource) to a given ResourceId.

The ImageModel is generated by calling the method `createImage(String src, int width, int height)`. Within the method, the **root directory**, in which the**graphics** are saved, is enhanced. In the above example, the graphics are searched for in the path**`myDef/image/`**.

If the graphics are to be stored in another location, the directory path can be modified here.

Note: When graphics are being registered, in the case of **hover effects** for buttons, it must be noted that the image with the ending xxx**1**.gif, which represents the normal state of the button, is registered in each case.

The generation of the **`ColorPalette`**, which is registered in the method `doRegisterColors()`, is also done with the ResourceFactory Tool. After creation, the corresponding Java class is simply stored in the folder of the new PainterFactory. Thus, the new Painter consists of the following classes.

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

CodeSnippet 2: Klassen des angepassten Painters im Überblick

2.4 Generating StyleSheets

The generation of StyleSheets for customizations of the DefaultPainter is supported with the ResourceFactory from version 1.1 onwards. For this purpose, the Environment section contains a few Style Definitions that point to a globally defined color table. Within the Style Definitions, the colors can be specified directly, or only the globally defined color values are overwritten.

After the transfer of the color values, the StyleSheets can be generated with the Ant Task “build-Res“. The StyleSheet files are then transferred to the Web resources of the application.

```
<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 ResourceFactory (extract)

3 Modifiable images DefaultPainters

This chapter shows which graphics can be changed within a control element that is drawn with the DefaultPainter.

3.1 ListControl

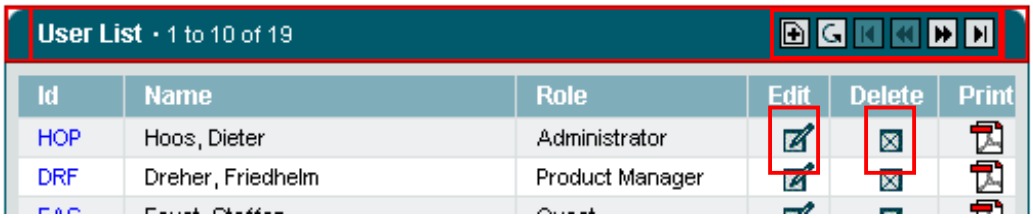
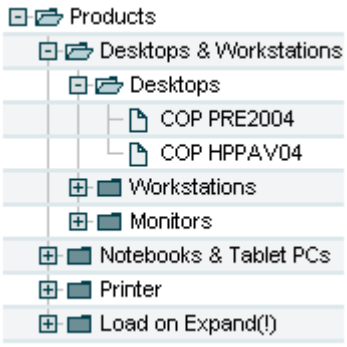


Figure 2: Modifiable graphics - ListControl

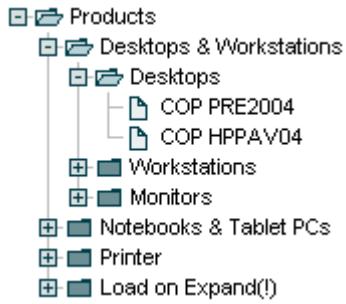
Image	ResourceId	Example	Width	Height
Left corner	CORNER_TABLE_LEFT	corners/l.gif	10	17
Right corner	CORNER_TABLE_RIGHT	corners/r.gif	10	17
Buttons in the Header of the table				
New Button	BUTTON_CREATE_1	buttons/btnCreate1.gif	15	15
Refresh Button	BUTTON_REFRESH_1	btnRefresh1.gif	15	15
First Button (aktiv)	BUTTON_FIRST_1	buttons/btnFirst1.gif	15	15
First Button (inaktiv)	BUTTON_FIRST_2	buttons/btnFirst2.gif	15	15
Previous Button (aktiv)	BUTTON_PREVIOUS_1	buttons/btnPrev1.gif	15	15
Previous Button (inaktiv)	BUTTON_PREVIOUS_2	buttons/btnPrev2.gif	15	15
Next Button (aktiv)	BUTTON_NEXT_1	buttons/btnNext1.gif	15	15
Next Button (inaktiv)	BUTTON_NEXT_2	buttons/btnNext2.gif	15	15
Last Button (aktiv)	BUTTON_LAST_1	buttons/btnLast1.gif	15	15
Last Button (inaktiv)	BUTTON_LAST_2	buttons/btnLast2.gif	15	15
Icons of the edit, delete und check column				
Edit Icon	ICON_EDIT	icons/edit.gif	16	16
Delete Icon	ICON_DELETE	icons/delete.gif	16	16
CheckColumn (unchecked)	ICON_CHECK	icons/check.gif	16	16
CheckColumn (checked)	ICON_CHECKED	icons/checked.gif	16	16

Icons to sort the columns				
Icon sortable	BUTTON_SORTABLE	buttons/btnSortable1.gif	11	13
Icon sortable ascending	BUTTON_SORTASC	buttons/btnSortUp1.gif	11	13
Icon sortable descending	BUTTON_SORTDESC	buttons/btnSortDown1.gif	11	13

3.2 TreeControl



Standard depiction



With a different StyleSheet as shown left.

```
<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>
```

Within the TreeControl, the graphics for the nodes and lines can be exchanged.

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
	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 closed folders as well as the graphics of an element can also be modified within the JSP page via an Imagemap.

Naming conventions for the images of the tree structure:

The name of an image of the restructure is a bit-coded decimal number.

Hex	Dez	Meaning
0x20	32	Plus sign before the item
0x10	16	Minus sign before the item
0x08	8	Connecting line towards the North
0x04	4	Connecting line towards the South
0x02	2	Connecting line towards the East
0x01	1	Connecting line towards the West

3.3 TreeListControl

Figure 3: Modifiable 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 header of the table				
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 column				
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
CheckColumn (unchecked)	ICON_CHECK	icons/check.gif	16	16
CheckColumn (checked)	ICON_CHECKED	icons/checked.gif	16	16
Icons to sort the columns				
Icon sortable	BUTTON_SORTABLE	buttons/btnSortable1.gif	11	13
Icon ascending	BUTTON_SORTASC	buttons/btnSortUp1.gif	11	13
Icon descending	BUTTON_SORTDESC	buttons/btnSortDown1.gif	11	13

3.4 TabSet

Figure 4: Modifiable graphics - TabSetControl

Image	ResourceId	Example	Width	Height
Selected Tab left corner	TABSET_TABSEL_L_COLOR	tab/tabLSel_{0}.gif	10	19
Selcted 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 cornner	TABSET_TABDISABLED_L	tab/tabDisL.gif	8	19
Disabled Tabe Background	TABSET_TABDISABLED_BG	tab/tabDisBg.gif	1	19
Disabled Tab right cornner	TABSET_TABDISABLED_R	tab/tabDisR.gif	10	19

3.5 Tabbar

Figure 5: Modifiable graphics – Tabbar Control

Image	ResourceId	Example	Width	Height
Selected Tab left corner	TABBAR_TABSEL_L_COLOR	tab/tabLSel_{0}.gif	10	19
Selcted Tab background	TABBAR_TABSEL_BG_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 cornner	TABBAR_TABDISABLED_L	tab/tabDisL.gif	8	19
Disabled Tabe Background	TABBAR_TABDISABLED_BG	tab/tabDisBg.gif	1	19
Disabled Tab right cornner	TABBAR_TABDISABLED_R	tab/tabDisR.gif	10	19

3.6 BreadCrumbs

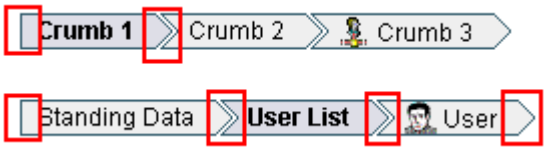


Figure 6: Modifiable graphics – BreadCrumb Control

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 Default Form

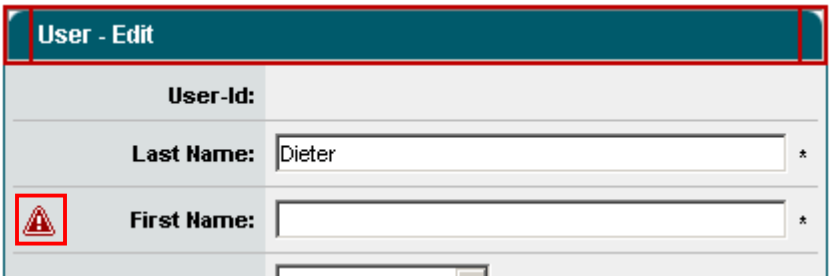


Figure 7: Modifiable graphics -form

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 icon				
Error sign	IMAGE_ERROR_INPUT	errInput.gif	16	16

3.7.2 Seach form

Figure 8: Modifiable graphics – Search form

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

3.7.3 Message dialog

Figure 9: Modifiable graphics – Message dialog

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

3.7.4 Error dialog

Figure 10: Modifiable graphics – Error dialog

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

3.8 HeadLine Control

The color matching for the Headline Control element is done via the customization of the colors within the StyleSheet.

4 Support

We would be happy to be of service if you have any questions or problems. Please use our Service Form on our homepage for your queries. We shall endeavor to answer your queries as quickly as possible.