SCC
INFORMATIONSSYSTEME

COMMON CONTROLS

Individualisierung des DefaultPainters

Das Erscheinungsbild an ein eigenes Corporate Design anpassen

SCC Informationssysteme GmbH

Version 1.6 · Stand: 15. Februar 2006

Impressum

HERAUSGEBER

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

Inhalt

1 Einleitung

Dieses Dokument beschreibt, wie sich der im Lieferumfang enthaltenen Default Painter (DefPainterFactory) an einen alternativen HTML Stylesheet anpassen lässt. Damit wird eine Anpassung der Kontrollelemente an das eigene Corporate Identity (CI) ermöglicht.

Zudem kann das Aussehen der Kontrollelemente durch den Austausch einiger Grafiken im begrenzten Umfang verändert werden. Ein Beispiel hierfür stellt der Default Painter 2 (Def2PainterFactory) dar, der keine Rundungen mehr im Design der Kontrollelemente verwendet und auch den Kopfbereich der Formularelemente anders gestaltet.

Für die hier beschriebenen Anpassungen bedarf es keiner vertiefenden Kenntnisse der Framework Architektur und Klassen. Dies wird nur dann erforderlich, wenn das visuelle Design eines Kontrollelementes grundlegend umgestellt werden muss oder neue Funktionen integriert werden sollen. In solchen Fällen reicht eine Anpassung des Default Painters nicht mehr aus und es muss ein neuer Painter entwickelt werden. Dies ist jedoch nicht Gegenstand dieses Dokumentes.

Bei der Individualisierung des DefaultPainters kann in folgenden Schritten vorgegangen werden:

- Erstellung eines HTML Designs auf Basis der Oberfläche des Default Painters
- Erstellung der abgeleiteten PainterFactory Klasse
- Erzeugung einer abgeleiteten ResourceMap (Registrierung von Grafiken) Klasse
- Generierung der notwendigen Cascading Stylesheets (CSS) und ColorPalette mit dem ResourceFactory Tool
- Generierung von Schaltflächen mit dem ResourceFactory Tool

2 Individualisierung des DefaultPainters

2.1 Erstellung eines eigenen Farb-Designs

Zu Begin wird das neue Design der Benutzeroberfläche festgelegt. Dabei kann man auf bestehende Bildschirmfotos der Oberfläche zurückgreifen, wie sie der Default Painter erzeugt. Durch die Variation der Farben, lässt sich dann Schritt für Schritt ein neues Farbschema für die Applikation erarbeiten. Die so gewonnenen Farbwerte fliesen in den Stylesheet des neuen Painters ein.

Das ResourceFactory Tool stellt einen Stylesheet Generator zur Verfügung, der die Stylesheet Dateien anhand von Templates erstellt. Die Farbwerte für die einzelnen Kontrollelemente werden dazu in der Ressourcen XML Datei der ResourceFactory konfiguriert. Durch die Verwendung von Properties lässt sich hier die Anzahl der einzelnen Farbwerte reduzieren und ähnliche Elemente (Überschriften, Hintergrundfarben etc.) können mit dem gleichen symbolischen Farbwert belegt und verändert werden, Mit der ResourcFactory können so Änderungen am Farbschema schnell umgesetzt und getestet werden. Das ResourceFactory Tool ist in einem eigenene Dokument beschrieben.

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>

```

Auszug Resource XML Datei ResourceFactory

Neben Farben, lassen sich in den Kontrollelementen auch einzelne grafische Elemente wie beispielsweise die Ecken von Listen austauschen. Neue Grafiken werden zunächst von Hand gezeichnet und dann dem Painter in einer ResourceMap Java Klasse bekannt gemacht. Eine ResourceMap definiert für einen Painter alle notwendigen Grafiken, die er für ein bestimmtes Design benötigt. In Kapitel 3 findet sich ein Überblick über alle Grafiken welche sich im Zuge der Anpassung des Default Painter austauschen lassen. Der Aufbau der ResourceMap wird in Kapitel 2.3 beschrieben.

2.2 Erstellung der PainterFactory Klasse

Nachdem das Design (CSS Dateien und Images) für die Anwendung fertiggestellt wurde und die Farbwerte und Grafiken feststehen, wird die PainterFactory für den neuen Painter erzeugt. Die PainterFactory wird später beim Start der Anwendung registriert und bestimmt damit das (neue) Design der Oberfläche.

Ihre zentrale Aufgabe ist die Erzeugung von Kontrollelement Paintern, welche den eigentlichen HTML Code für die Ausgabe erzeugen.

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());
    }
}

```

CodeSnippet 1: Registrierung der neuen Painterfactory

Die Erstellung einer neuen PainterFactory erfolgt durch Ableitung von der bestehenden DefPainterFactory Klasse. Alternativ lässt sich das nachfolgende Code Fragment kopieren. Die Blau hervorgehobenen Stellen müssen individuell angepasst werden und sind im Anschluss beschrieben.

Code Fragment - MyNewPainterFactory

JAVA

```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 () {
```

```
33          super();
34      }
35
36      /**
37       * @see com.cc.framework.ui.painter.PainterFactory#createResourceMap()
38       */
39      protected ResourceMap createResourceMap() {
40          return new MyNewPainterResourceMap();
41      }
42
43      /**
44       * Returns the unique Id for this Painterfactory
45       * @return  The unique Id for this Painterfactory which is "def"
46       */
47      public String getFactoryId() {
48          return "MyDef";
49      }
50
51      /**
52       * Returns the base directory used for resource by the Painterfactory
53       * @return  The web resource directory
54       */
55      public String getResourceDir() {
56          return RESOURCE_DIR;
57      }
58
59      /**
60       * Returns the single instance of the class (singleton)
61       * @return  The single instance  of this PainterFactory
62       */
63      public static PainterFactory instance() {
64          return instance;
65      }
66
67      /**
68       * @see com.cc.framework.ui.painter.PainterFactory#doCreateHeaderIncludes(JspWriter)
69       */
70      protected void doCreateHeaderIncludes(JspWriter writer) throws IOException {
71
72          StringBuffer buf = new StringBuffer();
73
74          buf
75              .append("<!-- BEGIN Framework (MyDefPainter) -->")
76              .append("<link rel='stylesheet' href='")
77              .append(RESOURCE_DIR)
78              .append("style/default.css' charset='ISO-8859-1' type='text/css'>");
79
80          buf
81              .append("<script language='JavaScript' src='")
82              .append(RESOURCE_DIR)
83              .append("jscript/functions.js'></script>")
84
85              .append("<script language='JavaScript' src='")
86              .append(RESOURCE_DIR)
87              .append("jscript/controls.js'></script>")
88              .append("<script language='JavaScript' src='")
89              .append(RESOURCE_DIR).append("jscript/tabset.js'></script>")
90              .append("<!-- END -->");
91
92          // write the buffer
93          writer.println();
94          writer.println(buf);
95          writer.println();
96      }
```

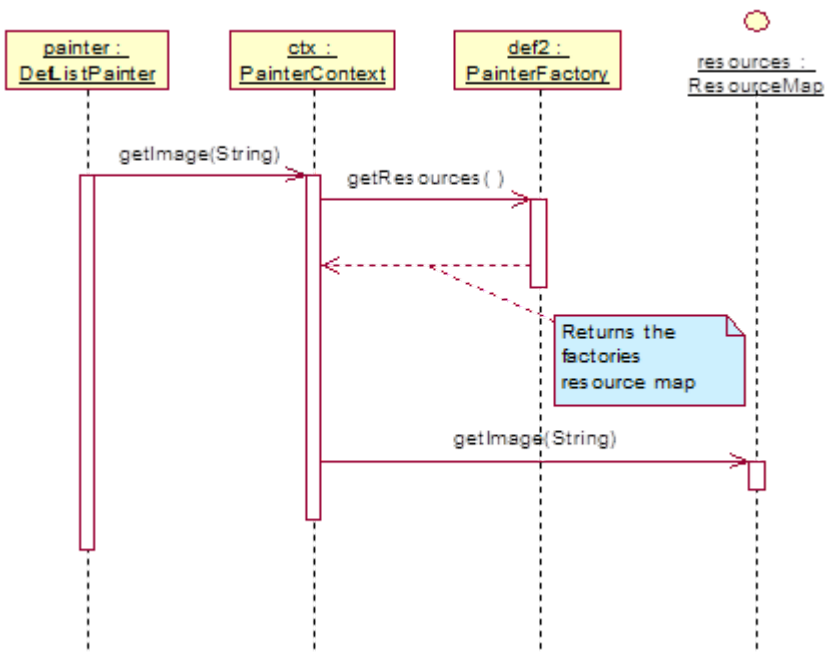
```
97
98  }
99
```

Zur Erstellung einer neuen PainterFactory Klasse müssen die folgenden Abschnitte aus dem Code Fragment angepasst werden:

Abschnitt	Beschreibung
RESOURCE_DIR	Legt das Verzeichnis fest, in welchem die neuen Ressourcen wie StyleSheets und Images gefunden werden. Für den DefaultPainter ist beispielsweise das Verzeichnis „fw/def“ gesetzt.
createResourceMap()	Die Methode muss eine Instanz einer ResourceMap zurückliefern, die die neuen Grafiken registriert. Die Erstellung der ResourceMap wird im Kapitel 2.3 beschrieben.
getFactoryId()	Die Methode muss einen symbolischen Namen für die PainterFactory zurückliefern. Der Aufruf der Methode des DefaultPainter liefert beispielsweise „def“. Das Kürzel „def“ sollte dem Common Controls Framework vorbehalten bleiben.
doCreateHeaderIncludes	Die Methode dient dazu, um am Anfang einer JSP Seite die notwendigen Stylesheets und JavaScript Blöcke zu inkludieren. Eine JavaScript Datei, die nur auf einer einzelnen Seite benötigt wird, sollte hier nicht aufgenommen werden. Das <util:jsp> Tag am Anfang einer JSP Seite schreibt den hier generierten String in die erzeugte HTML Seite (vgl. Common Controls Quickstart <util:jsp directive="includes"/>).

2.3 Erzeugung einer ResourceMap

Eine ResourceMap definiert für einen Painter alle notwendigen Grafiken und Farben, die er für ein bestimmtes Design der Oberfläche benötigt. Das folgende UML Sequenzdiagramm zeigt, wie ein Painter über die ResourceMap auf ein Image zugreift.



Eine ResourceMap wird von der Klasse com.cc.framework.ui.painter.def.DefResourceMap abgeleitet. Dies hat den Vorteil, dass nicht alle, sondern nur die neuen, Grafiken festgelegt werden müssen.

Die Registrierung der Grafiken findet in der Methode doRegisterImages() statt. Da der bestehende DefaultPainter angepasst wird, stehen die symbolischen Namen der Grafiken bereits fest (siehe Kapitel 3). Es findet hier nur noch ein Mapping auf die Datei im Ressourcenverzeichnis statt.

Eine einzelne Ressource (Image) wird mittels des Aufrufs registerImage(...) registriert.

registerImage(String resourceId, ImageModel model)	Registriert zu einer existierenden internen ResourceId ein Image.
registerImage(int size, String resourceId, ImageModel model)	Registriert zu einer existierenden internen ResourceId ein Image. Das size Argument dient zur Unterscheidung von Grafiken, die in unterschiedlichen Größen bereitgestellt werden müssen. Relevant ist dies z.B. bei dem Tree- und dem TreeListControl. Während das TreeControl Images in der Größe 15x15 für Ordner- oder Knotensymbole verwendet, benötigt das TreeListControl die Grafiken in der Größe 20x20. Durch die Angabe der Größe, kann der Painter nun auf die entsprechende Kollektion zugreifen. Sie dient hier als nur zur internen Verwaltung.

Bei der Anlage einer neuen ResourceMap orientiert man sich am günstigsten am der DefResourceMap des DefaultPainters und ändert in der eigenen Klasse die Namen und Größenabmessungen der Images wie benötigt ab. Das folgende Beispiel zeigt die ResourceMap des DefaultPainters2.

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);
    }
}
```

Eine Besonderheit ergibt sich bei der Benennung von Ressourcen, die ein Painter in verschiedenen Farbwerten benötigt. Ein Beispiel hierfür ist das TabSetControl, welches untergeordnete Tabsets enthalten kann. Wegen der wechselnden Hintergrundfarbe müssen für die selektierten Taben unterschiedliche Images angefertigt werden.

Damit innerhalb der ResourceMap aber nicht alle Grafiken für alle unterschiedlichen Farbwerte definiert werden müssen, wird auf die Notation tabLSel_{0}.gif zurückgegriffen. Diese besagt, dass der Painter für den Ausdruck {0} den jeweils im Kontrollelement konfigurierten Farbwert einsetzen soll.

Anmerkung: Die Hintergrundfarbe eines TabSets wird bei der Deklaration des TabSets in der JSP Seite im bgcolor-Attribute hinterlegt. Wird das Attribut nicht angegeben, wird per Default der Farbwert #EFEFEF

verwendet.

Beispiel: Das erste Tabset besitz die Hintergrundfarbe #EFEFEF. Das verschachtelte Tabset die Farbe #DADFE0. Der Tabset-Painter benötigt für die selektierten Zustände die folgenden Grafiken:

CODE

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

Gleichwohl werden die Images aber nur als tabLSel_{0}.gif, tabBgSel_{0}.gif und tabRSel_{0}.gif registriert.

Die Registrierung von Grafiken erfolgt mittels der Methode registerImage(..). Diese ordnet einer gegebenen ResourceId eine ImageModel (Image Ressource) zu.

Die Erstellung des ImageModels erfolgt über den Aufruf der Methode createImage(String src, int width, int height). Innerhalb der Methode wird das Root Verzeichnis, indem die Grafiken abgelegt sind, ergänzt. Im obigen Beispiel werden die Grafiken im Pfad myDef/image/ gesucht.

Wenn die Grafiken an einer anderen Stelle abgelegt werden sollen, kann auf den Verzeichnispfad an dieser Stelle Einfluss genommen werden.

Hinweis: Bei der Registrierung von Grafiken ist bei Hover Effekten für Schaltflächen darauf zu achten, dass jeweils das Image mit der Endung xxx1.gif registriert wird, welches den normalen Zustand der Schaltfläche repräsentiert.

Die Generierung der ColorPalette, welche in der Methode doRegisterColors() registriert wird, erfolgt ebenfalls über das ResourceFactory Tool. Die entsprechende Java Klasse wird nach der Erstellung einfach in das Verzeichnis der neuen PainterFactory abgelegt. Der neue Painter besteht damit aus den folgenden Klassen.

CODE

```
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 Generierung von StyleSheets

Die Generierung der StyleSheets für Anpassungen des DefaultPainters wird mit der ResourceFactory ab der Version 1.1 unterstützt. Der Environment-Abschnitt enthält hierzu einige Style Definitionen, die auf eine global definierte Farbtabelle verweisen. Innerhalb der Style Definitionen können die Farben direkt angegeben werden oder es werden nur die Global definierten Farbwerte überschrieben.

Nach der Übernahme der Farbwerte, lassen sich die StyleSheets über den Ant Task „build-res“ erzeugen. Die StyleSheet Dateien werden anschießend in die Web Ressourcen der Anwendung übernommen.

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 – Konfigurationsdatei ResourceFactory (Auszug)

3 Individualisierbare Grafiken DefaultPainters

Dieses Kapitel zeigt welche Grafiken innerhalb eines Kontrollelementes, das über den DefaultPainter gezeichnet wird, abgeändert werden können.

3.1 ListControl

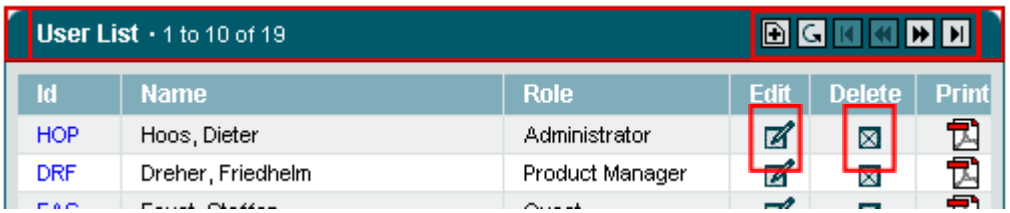
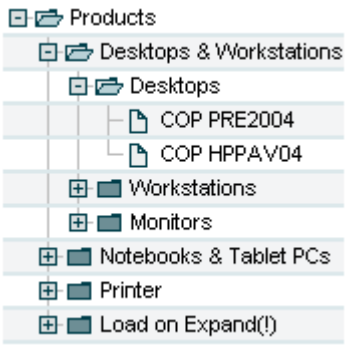


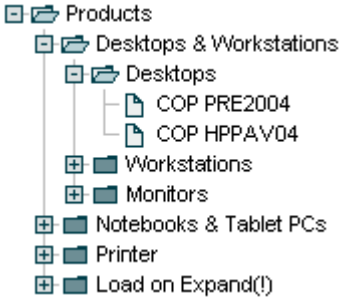
Abbildung 2: Änderbare Grafiken - ListControl

Image	ResourceId	Beispiel	Breite	Höhe
Linke Ecke	CORNER_TABLE_LEFT	corners/l.gif	10	17
Rechte Ecke	CORNER_TABLE_RIGHT	corners/r.gif	10	17
Buttons im Header der Tabelle				
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

Image	ResourceId	Beispiel	Breite	Höhe
Icons der Edit, Delete und Check Spalte				
Edit Icon	ICON_EDIT	icons/edit.gif	16	16
Delete Icon	ICON_DELETE	icons/delete.gif	16	16
CheckSpalte (unchecked)	ICON_CHECK	icons/check.gif	16	16
CheckSpalte (checked)	ICON_CHECKED	icons/checked.gif	16	16
Icons zur Sortierung der Spalten				
Icon Sortierbar	BUTTON_SORTABLE	buttons/btnSortable1.gif	11	13
Icon aufsteigende Sortierung	BUTTON_SORTASC	buttons/btnSortUp1.gif	11	13
Icon absteigende Sortierung	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>
```

Innerhalb des TreeControls können die Grafiken für die Knoten und Linien ausgetauscht werden.

Image	ResourceId	Beispiel	Breite	Höhe
	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	Beispiel	Breite	Höhe
	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

Anmerkung:

Die Grafiken für den offenen und geschlossenen Ordner sowie die Grafik ein Element lassen sich auch innerhalb der JSP Seite über einen ImageMap abändern.

Namenskonvention für die Images der Baumstruktur:

Bei dem Namen eines Bildes der Baumstruktur handelt es sich jeweils um eine bitkodierte Dezimalzahl

Hex	Dez	Bedeutung
0x20	32	Plus-Zeichen vor dem Item
0x10	16	Minus-Zeichen vor dem Item
0x08	8	Verbindungslinie nach Norden
0x04	4	Verbindungslinie nach Süden
0x02	2	Verbindungslinie nach Osten
0x01	1	Verbindungslinie nach Westen

3.3 TreeListControl

Abbildung 3: Änderbare Grafiken - TreeListControl

Image	ResourceId	Beispiel	Breite	Höhe
Linke Ecke	CORNER_TABLE_LEFT	corners/l.gif	10	17
Rechte Ecke	CORNER_TABLE_RIGHT	corners/r.gif	10	17
Buttons im Header der Tabelle				
New Button	BUTTON_CREATE_1	buttons/btnCreate1.gif	15	15
Refresh Button	BUTTON_REFRESH_1	btnRefresh1.gif	15	15
Icons der Add, Edit, Delete und Check Spalte				
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
CheckSpalte (unchecked)	ICON_CHECK	icons/check.gif	16	16
CheckSpalte (checked)	ICON_CHECKED	icons/checked.gif	16	16
Icons zur Sortierung der Spalten				
Icon Sortierbar	BUTTON_SORTABLE	buttons/btnSortable1.gif	11	13
Icon aufsteigende Sortierung	BUTTON_SORTASC	buttons/btnSortUp1.gif	11	13
Icon absteigende Sortierung	BUTTON_SORTDESC	buttons/btnSortDown1.gif	11	13

3.4 TabSet

Abbildung 4: Änderbare Grafiken - TabSetControl

Image	ResourceId	Beispiel	Breite	Höhe
Hintergrund	TABSET_BACKGROUND	tab/tab.gif	1	19
Selektierte Tabe linke Ecke	TABSET_TABSEL_L_COLOR	tab/tabLSel_{0}.gif	10	19
Selektierte Tabe Hintergrund	TABSET_TABSEL_BG_COLOR	tab/tabRSel_{0}.gif	1	19
Selektierte Tabe rechte Ecke	TABSET_TABSEL_R_COLOR	tab/tabBgSel_{0}.gif	11	19
Unselektierte Tabe linke Ecke	TABSET_TABUNSEL_L	tab/tabL.gif	8	19
Unselektierte Tabe Hintergrund	TABSET_TABUNSEL_BG	tab/tabBg.gif	1	19
Unselektierte Tabe rechte Ecke	TABSET_TABUNSEL_R	tab/tabR.gif	10	19
Deaktivierte Tabe linke Ecke	TABSET_TABDISABLED_L	tab/tabDisL.gif	8	19
Deaktivierte Tabe Hintergrund	TABSET_TABDISABLED_BG	tab/tabDisBg.gif	1	19
Deaktivierte Tabe rechte Ecke	TABSET_TABDISABLED_R	tab/tabDisR.gif	10	19

3.5 Tabbar

Abbildung 5: Änderbare Grafiken – Tabbar-Control

Image	ResourceId	Beispiel	Breite	Höhe
Hintergrund	TABBAR_BACKGROUND	tab/tab.gif	1	19
Selektierte Tabe linke Ecke	TABBAR_TABSEL_L_COLOR	tab/tabLSel_{0}.gif	10	19
Selektierte Tabe Hintergrund	TABBAR_TABSEL_L_COLOR	tab/tabRSel_{0}.gif	1	19
Selektierte Tabe rechte Ecke	TABBAR_TABSEL_R_COLOR	tab/tabBgSel_{0}.gif	11	19
Unselektierte Tabe linke Ecke	TABBAR_TABUNSEL_L	tab/tabL.gif	8	19
Unselektierte Tabe Hintergrund	TABBAR_TABUNSEL_BG	tab/tabBg.gif	1	19
Unselektierte Tabe rechte Ecke	TABBAR_TABUNSEL_R	tab/tabR.gif	10	19
Deaktivierte Tabe linke Ecke	TABBAR_TABDISABLED_L	tab/tabDisL.gif	8	19
Deaktivierte Tabe Hintergrund	TABBAR_TABDISABLED_BG	tab/tabDisBg.gif	1	19
Deaktivierte Tabe rechte Ecke	TABBAR_TABDISABLED_R	tab/tabDisR.gif	10	19

3.6 BreadCrumbs

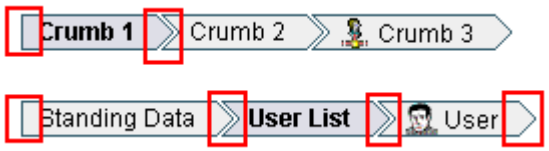


Abbildung 6: Änderbare Grafiken – BreadCrumb Control

Image	ResourceId	Beispiel	Breite	Höhe
	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 Formulare

3.7.1 Standard Formular

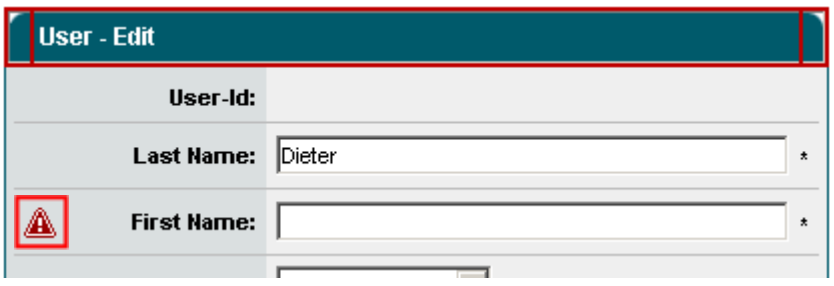


Abbildung 7: Änderbare Grafiken - Formular

Image	ResourceId	Beispiel	Breite	Höhe
Linke Ecke	CORNER_FORM_LEFT	corners/l.gif	10	17
Rechte Ecke	CORNER_FORM_RIGHT	corners/r.gif	10	17
Fehlerzeichen				
Fehlerzeichen	IMAGE_ERROR_INPUT	errInput.gif	16	16

3.7.2 SuchDialog

Abbildung 8: Änderbare Grafiken - Suchformular

Image	ResourceId	Beispiel	Breite	Höhe
Linke Ecke	CORNER_FORMSEARCH_LEFT	corners/l.gif	10	17
Image Lupe	IMAGE_MAGNIFIER	magnifier.gif	20	20
Rechte Ecke	CORNER_FORMSEARCH_RIGHT	corners/r.gif	10	17

3.7.3 Meldungsdialog

Abbildung 9: Änderbare Grafiken - Meldungsdialog

Image	ResourceId	Beispiel	Breite	Höhe
Linke Ecke	CORNER_FORM_LEFT_COLOR	corners/l_{0}.gif	10	17
Image Info	IMAGE_INFORMATION	info.gif	14	23
Rechte Ecke	CORNER_FORM_RIGHT_COLOR	corners/r_{0}.gif	10	17
Aufzählungszeichen				
Aufzählungs-zeichen	IMAGE_DOT_COLOR	dots/dot_{0}.gif	5	5

3.7.4 Fehlerdialog

Abbildung 10: Änderbare Grafiken - Fehlerdialog

Image	ResourceId	Beispiel	Breite	Höhe
Linke Ecke	CORNER_FORM_LEFT_COLOR	corners/l_{0}.gif	10	17
Image Error	IMAGE_ERROR	error.gif	14	23
Rechte Ecke	CORNER_FORM_RIGHT_COLOR	corners/r_{0}.gif	10	17
Aufzählungszeichen				
Aufzählungs-zeichen	IMAGE_DOT_COLOR	dots/dot_{0}.gif	5	5

3.8 HeadLine Control

Die Farbanpassung für das Headline Kontrollelement erfolgt über die Anpassung der Farben innerhalb des StyleSheets

4 Support

Bei Fragen oder Problemen stehen wir Ihnen gerne zur Verfügung. Benutzen Sie bitte für Anfragen unser Service Formular auf unserer Homepage. Wir sind bemüht Ihre Anfragen zeitnah zu beantworten.