1. Changing the Portal Look and Feel
   1.1. Motivation and Benefits
   1.2. Examples

2. Fundamentals
   2.1. Portal Desktop and it’s Elements
   2.2. Themes and Style Sheets
   2.3. Portal Framework Pages

3. Changing the Portal Look and Feel – Step by Step

4. Summary
Why Customize the Portal Look & Feel?

Style Guides

- Usually released from the Corporate Communications department
- Are most of the time a binding requirement ("pixel perfect")
- Rule of thumb: The bigger the company and the more exposed the portal the more important is the compliance of the portal’s look & feel with the style guide.
Why Customize the Portal Look & Feel? Usability and Performance

Especially when exposing the portal into the Internet, the portal’s look & feel has to abide by the laws of the Internet, i.e. it needs to have a web-like behavior, be fast-loading, intuitive, accessible, etc.
Why Customize the Portal Look & Feel?
Accessibility

**Usability and** – for some customers of higher importance – **accessibility** are key factors for the success of the portal (project).

- Institutions and organizations in the public sector (government, military, hospitals, etc.), formerly state-owned companies, and companies with a strong labor or trade union are very sensitive to usability and accessibility requirements like stated in Section 508 (mainly for US), “Web Content Accessibility Guidelines 1.0” (Web Accessibility Initiative of the W3C), or Barrierefreie Informationstechnik Verordnung – BITV” (Germany).

- “Accessibility” covers many aspects:
  - Different types and grades of impairment (blind, spastic, deaf, …)
  - Different techniques to accomplish accessibility
  - A website/portal can be 0 – 100% accessible according to the guidelines.
Agenda

1. Changing the Portal Look and Feel
   1.1. Motivation and Benefits
   1.2. Examples

2. Fundamentals
   2.1. Portal Desktop and it’s Elements
   2.2. Themes and Style Sheets
   2.3. Portal Framework Pages

3. Changing the Portal Look and Feel – Step by Step

4. Summary
This presentation covers mainly the customizing of the look & feel of the portal framework, not the content.

- The portal framework comprises all elements that are necessary for navigation and for displaying certain information and/or functionality that are available on each page the user can navigate to. E.g. top level and detail navigation, header with login/logout link, search field, footer with copyright, imprint, etc.

- iView trays, though visually part of the content, are considered parts of the framework.

- The content finally comes from those iViews or pages that you assign to the navigation hierarchy within roles and worksets.
**Display Rules**
- Set of conditions defined by a portal administrator determining which portal is assigned to portal users at runtime.
- Allows organizations to allocate portal desktops with varying designs, branding, or layouts to different departments, sub-companies, or platform-specific scenarios.

**Desktops**
- Define the structural layout and design of the portal workspace displayed on the screen.
- Collection of framework pages and portal themes

**Themes**
- Define the visual appearance of the control elements, font size, colors, and contrast of the user interface elements of a portal desktop.
- Do *not* define the layout, structure, or contents of a portal desktop.

**Framework Pages**
- A portal page object containing user interface units, which together define the layout and structure of a portal desktop.
- A framework page contains the core iViews and pages that are required by a user to operate and navigate the portal.

**Layouts**
- Structures page content by providing a number of so called containers which hold the iViews you add to a page.
- Each layout has a specific arrangement of containers.
- If custom developed, each container can provide a specific tray design.
Logon Procedure - Resolution of Portal Display Rules

User X logs on or anonymously enters portal.

IF (condition)
- user
- group
- role
- connection bandwidth
- portal URL alias
- browser type
- browser version

THEN (desktop)
### 10 Most Active Users

<table>
<thead>
<tr>
<th>First Logon</th>
<th>ID</th>
<th>Type</th>
<th>Hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>18. Apr 2006 15:55 (GMT)</td>
<td>BuyerB</td>
<td>Authenticated</td>
<td>17767</td>
</tr>
<tr>
<td>03. May 2006 18:57 (GMT)</td>
<td>BurkeP</td>
<td>Authenticated</td>
<td>15888</td>
</tr>
<tr>
<td>15. Mar 2007 11:55 (GMT)</td>
<td>belli</td>
<td>Authenticated</td>
<td>7348</td>
</tr>
<tr>
<td>17. Jul 2006 08:55 (GMT)</td>
<td>d033035</td>
<td>Authenticated</td>
<td>7278</td>
</tr>
<tr>
<td>05. Jul 2006 14:56 (GMT)</td>
<td>DD045154</td>
<td>Authenticated</td>
<td>5168</td>
</tr>
</tbody>
</table>
Theme Editor in Detail

The theme editor of the portal is used to adjust the formatting of Web applications within the Unified Rendering Framework.

Choose from various elements, e.g. from portal and KM admin tools, BI, ITS, Web Dynpro

- Change the color scheme
- Change the font type and size
- Include your own images
Changing Look and Feel of the Logon Page

- The portal comes with a standard logon screen
- You can modify the sources - Java Server Pages (JSPs) - of the logon component
- The branding image can be exchanged more easily (ume.logon.branding_image)
Framework Concepts

Default (Standard) Concept
- Two-frame, nested framework
- Navigation links are rendered as “Javascript-links” using an onclick-attribute that triggers an navigation event via EPCM.

Light Concept
- Single-frame, flat framework
- Navigation links are rendered as simple anchors with an href-attribute.
Default Framework Page (Standard)
- Should be used for integrating advanced content such as SAP applications and out of the box business packages to registered users.

Light Framework Page
- Can be used to provide anonymous user community access to static information.
- Mostly B2C scenarios, where network performance is of great importance.
- Enables the usage of standard browser functionality.

Other, Custom-Built Framework Pages
- Can be built specifically by the customer by using the portal APIs.
- Such „intermediate“ frameworks can balance performance and functionality.
# Detailed Comparison of Portal Frameworks

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept / Characteristic</td>
<td>nested framework with isolated innerpage (Frame)</td>
<td>flat framework with embedded innerpage</td>
<td>customer specific framework: flat or nested</td>
</tr>
<tr>
<td>Scope</td>
<td>for business applications with advanced functionality support</td>
<td>for lightweight, static content, web-like navigation and improved network traffic (client performance)</td>
<td>best fit concerning functionality, Look &amp; Feel and performance</td>
</tr>
<tr>
<td>Support for</td>
<td>Enterprise Portal Client Framework (EPCF; provides an infrastructure for scripting used in Views and by the portal)</td>
<td>full support for EPCF (default EPCF level = 1) (System API, Event API, Client Data Bag API, WorkProtect API, Navigation API, EPCM Proxy)</td>
<td>by default no use of EPCF (EPCF level = 0) but resource sensitive framework, depending on content</td>
</tr>
<tr>
<td>- Portal Eventing (provides methods for event handling on the client)</td>
<td>supported</td>
<td>depending on content (i.e., EPCF level is set to 1 or 2. Portal Eventing is possible, regardless of the framework's EPCF level)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Work Protect Mode (handling of unsaved data on the page)</td>
<td>supported</td>
<td>not supported; links are rendered without onclick='EPCM.doNavigate()'; which is responsible for checking the dirty flag</td>
<td>depends on chosen navigation method</td>
</tr>
<tr>
<td>- Session Management (e.g., termination of backend sessions when logging off)</td>
<td>supported</td>
<td>not supported; missing EPCF within framework's HTML response and flat framework</td>
<td>customer specific</td>
</tr>
<tr>
<td>Navigation concepts</td>
<td>Navigation Method</td>
<td>byEPCM: link based on javascript onclick='EPCM.doNavigate()'</td>
<td>byURL: link based on href attribute ('href='URL')</td>
</tr>
<tr>
<td>- Object Based Navigation (based on EPCF)</td>
<td>supported</td>
<td>not supported (no subscriber for the event is available)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Relative navigation (based on EPCF)</td>
<td>supported</td>
<td>not supported (no subscriber for the event is available)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Flash URL (short URL)</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
</tr>
<tr>
<td>Quick Links</td>
<td>supported</td>
<td>supported</td>
<td>supported</td>
</tr>
<tr>
<td>Content / Applications</td>
<td>Web Dynpro</td>
<td>supported</td>
<td>depends on required features (e.g., application makes use of Portal Eventing, Work Protect Mode, Session Mgmt)</td>
</tr>
<tr>
<td>- Business Server Page</td>
<td>supported</td>
<td>depends on required features (e.g., application makes use of Portal Eventing, Work Protect Mode, Session Mgmt)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- HTML5</td>
<td>supported</td>
<td>depends on required features (e.g., application makes use of Portal Eventing, Work Protect Mode, Session Mgmt)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Knowledge Management</td>
<td>supported</td>
<td>supported with restrictions (KM uses HTML5, anonymous access requires special settings)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Collaboration</td>
<td>supported</td>
<td>supported</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Visual Composer (generates WXML or Flash application)</td>
<td>supported</td>
<td>depends on required features (e.g., application makes use of Portal Eventing, Work Protect Mode, Session Mgmt)</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Guided Procedure</td>
<td>supported</td>
<td>supported</td>
<td>customer specific</td>
</tr>
<tr>
<td>- Federated Portal Content (FPN)</td>
<td>support for Remote Role Assignment and Remote Delta Link</td>
<td>Remote Role Assignment not supported</td>
<td>customer specific</td>
</tr>
</tbody>
</table>

[https://www.sdn.sap.com/irj/sdn/nw-portalandcollaboration](https://www.sdn.sap.com/irj/sdn/nw-portalandcollaboration)
Possibilities to create your own Framework Page (1/2)

100% Light Framework Page

Light Framework Page with Eventing

100% Default Framework Page

“more functionality”
“less performance”

E.g. switch on client-eventing (EPCF-level = 1), but keep flat page (single frame concept)
Possibilities to create your own Framework Page (2/2)

100% Light Framework Page

Default Framework Page without Eventing

100% Default Framework Page

E.g. switch off client eventing (EPCF-level = 0), but keep the full navigation (two frame concept)

“more performance”
“less functionality”
Tools for Customization

Within the Portal:
- Portal Content Studio
  - Content Administration
  - Desktop & Display Rules
  - Portal Content Translation
- Theme Editor
- (PCD Inspector)
  - Only if and when Portal Content Studio insufficient

External Tools:
- NetWeaver Developer Studio
- Adobe® Photoshop® or alike
- Text editors

Released APIs:
- JSP tag libraries for
  - Navigation
  - Framework
  - Layout
- Java APIs
  - Navigation
  - Desktop Service
  - Alias Service
  - Look & Feel Service
  - User Management (UME)
  - Knowledge Management
  - etc.
Agenda

1. Changing the Portal Look and Feel
   1.1. Motivation and Benefits
   1.2. Examples

2. Fundamentals
   2.1. Portal Desktop and it’s Elements
   2.2. Themes and Style Sheets
   2.3. Portal Framework Pages

3. **Changing the Portal Look and Feel – Step by Step**

4. Summary
There are more or less 5 different levels of how the portal look & feel can be adapted.

The levels vary in the degree of “Standardness” and “Design Flexibility”.

The higher the level the higher the TCO might be.

- Customization only
- Addon-iViews
- Smaller modifications
- Custom developed replacements
- Completely rewritten framework
Customization only

- Addon-iViews
- Smaller modifications
- Custom developed replacements
- Completely rewritten framework

- Order, visibility, and appearance of iViews in Framework and Desktop Inner Page
- Masthead iView
- Toolarea iView
- Top Level Navigation iView
- Page Toolbar iView
- Detail Navigation iView
- Related Links iView
- Content Area iView
- Portal Favorites iView
- Custom Theme
**Customization only**

**Framework Page / Desktop Innerpage**

- Change visibility and order of the contained iViews
- Change appearance of iViews (show/hide tray)

---

**Page Content / Page Layout**

From the Portal Catalog, select the content you want to add to the page. In the Property Editor, edit the property values of the page and those of its iViews and pages.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>ID</th>
<th>Visible</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masthead iView</td>
<td>com.sap.portal.masthead</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tool Area</td>
<td>com.sap.portal.toolAreaView</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top-Level Navigation</td>
<td>com.sap.portal.topLevelNavigationiView</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Title Bar</td>
<td>com.sap.portal.pageToolbar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Desktop Innerpage</td>
<td>com.sap.portal.innerpage</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Page Content / Page Layout**

Use this screen to view the layouts currently assigned to the page and to arrange the content. You can also add more layouts, remove layouts, and define a different default layout.

**Show Layout**: Framework Page Layout -- Default

**Framework Column**

- Masthead View
- Tool Area
- Top-Level Navigation
- Page Title Bar
Customization only

Masthead
- Show/hide links: Help, Log On / Log Off, New Session, Personalize
- URL of Help Link
- Show/hide Language Personalization for Anonymous Users

Toolarea
- Enable: Collaboration Launch Pad, KM Search, KM Search Advanced, Real-Time Collaboration
- KM (Advanced) Search Parameters
- Component URLs for Collaboration Launch Pad, Search and Real-Time Collaboration

Top Level Navigation
- Hovering
- Filtering Entry Points
- Defining the Sort Sequence of Navigation Nodes
- Number of Display Levels: [0, 1, 2] → adapt also corresponding attributes in Detail Navigation and Content Area iViews
Page Toolbar
- Show Back/Forward Links
- Show Breadcrumb
- Show History List

Detail Navigation
- Configuring Navigation Zoom
- Clicking Folder Name Launches First Node
- Display Parent of Hidden Node
- Enable Context Menu in Detailed Navigation
- Enable Text Wrapping in Detailed Navigation
- Highlight Entire Row of Selected Navigation Node
- Open Folder when Launched
- Show 'Add to Portal Favorites' Option in Context Menu
- Show 'Help' Option in Context Menu
- Start at Level: [1,2,3] → see Top Level Navigation
Customization only

Related Links
- Enable Text Wrapping

Content Area
- Number of Display Levels in TLN: [0,1,2] → see Top Level Navigation

Portal Favorites
- Layout Set
- Layout Set for Root Collection
- Layout Set Mode
- Path to Initially Displayed Folder
- Path to Root Folder for Navigation

Collaboration Detailed Navigation
- Same as Detailed Navigation
- Used in Light Mode → no: in Default FW Page, yes: in Light FW Page
Custom Theme

- Use the Theme Editor to create and customize a new theme.
- You can use the semicolon-hack to add certain, missing CSS attributes to elements and thus enhance the theme a little bit. (No official support!)
Example: Custom Theme
Addon-iViews

**Purpose:** For “simpler“ adaptations that cannot be handled by the customization of the framework iViews and the theme, e.g. the width and borders of the design bar on the left hand side of the TLN.

**Idea:** Develop a component that enhances certain aspects of the framework. This component is added as a merely functional iView to the Framework Page or Desktop Innerpage.

**Benefit:** not yet a real modification, just an “enhancement”

**Pitfall:** Relies on the stability of the HTML/CSS/Javascript output → less upgrade safe.
Problem

- When adding a footer iView underneath the Desktop Innerpage, a scrollbar appears in the Framework Page and the footer is hidden initially.
- This is due to the fullpage-height adjustment of the Desktop Innerpage.

Solution

- Add a bit of Javascript to the footer iView which overwrites the standard function that adjusts the iFrame’s height such that the footer still fits into the window.
Example: “Footer iView in Framework Page” (cont.)

1. Create a component (in the NWDS)...

```xml
<?xml version="1.0" encoding="utf-8"?>
<application alias="itel0">
  <application-config>
    <property name="Vendor" value="sap.com"/>
    <property name="SecurityArea" value="NetWeaver.Portal"/>
  </application-config>
  <component>
    <component name="footer">
      <component-config>
        <property name="ClassName" value="com.sap.portal.itelo.addon.Footer"/>
        <property name="SafeLevel" value="no_safety"/>
      </component-config>
      <component-profile/>
    </component>
  </component>
</application>
```

```
package com.sap.portal.itelo.addon;

import com.sapportals.portal.prt.component. *
import com.sapportals.portal.prt.resource.IResource;

public class Footer extends AbstractPortalComponent {
  public void doContent(IPortalComponentRequest request, IPortalComponentResponse response) {
    response.include(request, request.getResource(IResource.CSS, "css/footer.css"));
    response.include(request, request.getResource(IResource.JSP, "jsp/components/footer.jsp"));
  }
}
```
2. ...and a JSP file with the footer's HTML output:

```html
<div id="footer">
    Copyright 2007 ITel0 Corp.
</div>
```

3. Add the following Javascript to the output:

```javascript
<script>
/**
 * var that can be adjusted to one's needs */
var fullPageHeightAdjust = 0;
var fullPageHeightMin = 80;

/**
 * some helper functions */

function $(e) {
    if(typeof e==='string') return e;
    return document.getElementById(e);
}

function offsetTop(e) {
    for(var y=e.offsetTop, p=e.offsetParent; p!==null; p=p.offsetParent) y+=p.offsetTop;
    return y;
}

function getFullPageHeight(frame) {
    return Math.max(
        EPCM.getUserAgent()=='EPCM.MSIE' ? document.body.clientHeight : window.innerHeight - offsetTop(frame) + self.fullPageHeightAdjust,
        self.fullPageHeightMin
    );
}

(continued on next slide)
```
(continued from previous slide)

```javascript
/** this function replaces pageSupport.adjustFullPageIViews() */
function _adjustFullPageIViews() {
    var iframes = document.getElementsByTagName('IFRAME');
    for (var f=iframes.length-1; f>=0; f--) {
        var iframe = iframes[f];
        if (iframe.getAttribute('fullPage')=='true') { // only for fullpage iViews
            try {
                var h=getFullPageHeight(iframe);
                if (iframe.getAttribute('hasTray')=='true')
                    h-=offsetTop($("end_trayBottomBorder_"+iframe.id))
                        -offsetTop($("start_trayBottomBorder_"+iframe.id));
                iframe.style.height=h+"px";
            }
            catch(e) { /* resizing of the container iframe failed */
            }
        }
    }
}

/** overwrite pageSupport.adjustFullPageIViews() and initialize footer */
if (self.pageSupport && pageSupport.adjustFullPageIViews) {
    pageSupport.adjustFullPageIViews = _adjustFullPageIViews;
}
function initFooter() {
    var exec = "self.fullPageHeightAdjust-="+get("footer").offsetHeight; _adjustFullPageIViews();";
    window.setTimeout(exec, 10);
}
EPCM.subscribeEvent('urn:com.sapportals.portal:browser', 'load', initFooter);
</script>
4. In the portal, go to the Portal Content Studio (Content Administration → Portal Content) and create a new folder (say “ITelO”) that'll contain your new footer and framework page.

5. Copy the “Default Framework Page” from “Portal Content/Portal Users/Standard Portal Users” and paste it to your new folder; rename the page to “ITelO Framework Page” (if you will).

6. Edit the new framework page:
   - Add the “ITelO Footer” to the page.
   - Go to the layout view and reassure yourself that the footer iView is placed at the bottom.
   - Save.

7. Navigate to “Desktops & Display Rules” (System Administration → Portal Display) and create a new desktop, adding both the framework page and a theme.

8. Add the desktop to your display rules (“Portal Content/Portal Administrators/Super Administrator/Master Rule Collection” or some other, subordinate rule collection).
Example: “Footer iView in Framework Page” (cont.)

Result

The footer fits into the Framework Page without causing a scrollbar to appear.
Smaller modifications

**Purpose:** For „simpler“ adaptations that cannot be handled by the customization of the framework iViews and the theme, nor by addon-iViews, e.g. an additional link in the header or rounded tabs in the TLN.

**Idea:** Download and modify a component, e.g. the Masthead iView or the Framework Layout. This component then replaces the original one within the Framework Page or Desktop Innerpage.

**Benefit:** only smaller modifications “here and there”, most framework iViews/layouts come in form of an easily editable JSP file.

**Pitfall:** Modification! Relies on the stability of the standard framework functionalities → less upgrade safe.
Example: “Additional Link in the Masthead iview”

Problem

- A contact link should be added to the header next to the help link, but neither the iview settings of the Masthead iview nor the Theme Editor provide ways to add custom links. An addon-iview is not feasible as well because of the ‘non-trivial’ HTML of the header which doesn’t allow for changes to the DOM via Javascript.

- This is the only change that is needed for the ITelO Framework Page.

Solution

- Since this is only a minor change, it is okay to modify one standard component, the Masthead iview.
Example: “Additional Link in the Masthead iView” (cont.)

1. Download the original com.sap.portal.navigation.masthead.par.bak
   (System Administration ➔ Support Desk ➔ Portal Runtime ➔ Browse Deployment ➔ ROOT/WEB-INF/deployment/pcd)
Example: “Additional Link in the Masthead iView” (cont.)

2. Either import the PAR into the NWDS (new namespace/prefix!, e.g. “com.itelo.portal.masthead”) or unzip it to your file system.

3. Open the file PORTAL-INF/jsp/HeaderiView.jsp and insert your code:

```html
...<TD class="portletHeaderFunctionsTable">
  <TABLE border="0" cellspacing="0" cellpadding="0"
        class="portletHeaderFunctionsContainer" height="100%">
    <TR>
      <!-- modification START: DO17963 added contact link --%>
      <TD nowrap >
        <hbj:link id="ContactLink" tooltip="&lt;%=contactTooltipStr%>">
          linkDesign="FUNCTION" reference="#"/>
          <% if (!isPreview) { HelpLink.setOnClientClick("javascript:openContact();"); } %>
        <hbj:textView nested="true" text="&lt;%=contactTextStr%"/>
      </hbj:link>
    </TD>
    <!-- modification END --%>
    <%
      if (showHelpLink)
      { %>
        <TD nowrap >
          <hbj:link id="HelpLink" tooltip="&lt;%=helpTooltipStr%>">
            linkDesign="FUNCTION" reference="#"/>
            <% if (!isPreview) { HelpLink.setOnClientClick("javascript:openHelp();"); } %>
            <hbj:textView nested="true" text="&lt;%=helpTextStr%"/>
          </hbj:link>
        </TD>
      { %>
        <TD nowrap>
          <\}%
      ...
Example: “Additional Link in the Masthead iView” (cont.)

4. Save and either deploy the archive with the NWDS or create a new ZIP file (e.g. “com.itelo.portal.masthead.zip”) adding all previously unzipped files and folders, then rename the ending to “.par” and upload the archive manually (“/irj/go/portal/prtroot/PortalAnywhere”).

5. In the portal, go to the Portal Content Studio (→ Content Administration → Portal Content) and create a new folder (say “ITelO”) that’ll contain your new header and framework page.

6. Create a new iView from the recently uploaded portal archive, say “ITelO Masthead”.

7. Copy the “Default Framework Page” from “Portal Content/Portal Users/Standard Portal Users” and paste it to our new folder, and rename the page to “ITelO Framework Page”.

8. Edit the new framework page:
   - Add the “ITelO Masthead” iView to the page.
   - Remove or at least hide the original “Masthead iView”.
   - Go to the layout view and put your new iView to the top.
   - Save.
Example: “Additional Link in the Masthead iView” (cont.)

9. Navigate to “Desktops & Display Rules” (→ System Administration → Portal Display) and create a new desktop, adding both the framework page and a theme.

10. Add the desktop to the display rules (“Portal Content/Portal Administrators/Super Administrator/Master Rule Collection” or some other, subordinate rule collection).

---

**Result**

![Screen capture showing the iView page with the additional link in the masthead.](image-url)
Purpose: For „more complicated“ adaptations that cannot be handled by the customization of the framework iViews and the theme, e.g. navigation with pull-down menus instead of TLN and DTN, or custom layouts with custom trays.

Idea: Develop the necessary components from scratch, using e.g. JSP tag libraries or APIs, and replace the corresponding components of the framework.

Benefit: no real modification, rather custom development.

Pitfall: Because of possible dependencies of the framework functionalities both good knowledge and a test strategy for upgrades is needed.
Example: “Pull-down Menu for the TLN iView”

Problem

- The standard Top Level Navigation iView can render up to two levels of navigation, but in a horizontal, tab-structured manner only.
- There is a feature/attribute called “Hover” which causes the 2nd level of an inactive 1st-level tab to show, but this is nothing compared to a real pull-down menu.

Solution

- Write a new component that makes use of the JSP tag library for navigation, plus a simple, “carb-free” pull-down solution called “Suckerfish DropDowns”.*

*(see e.g. the following article: [http://www.htmldog.com/articles/suckerfish/dropdowns/](http://www.htmldog.com/articles/suckerfish/dropdowns/)*)
Example: “Pull-down Menu for the TLN iView” (cont.)

1. Create a component (in the NWDS)...

```xml
<?xml version="1.0" encoding="utf-8"?>
<application alias="itelo">
  <application-config>
    <property name="Vendor" value="sap.com"/>
    <property name="SecurityArea" value="NetWeaver.Portal"/>
    <property name="SharingReference" value="com.sap.portal.navigation.navigationtaglibrary"/>
  </application-config>
  <components>
    <component name="topnav">
      <component-config>
        <property name="ClassName" value="com.sap.portal.itelo.components.TopNav"/>
        <property name="SafetyLevel" value="no_safety"/>
      </component-config>
      <component-profile>
        <property name="NavigationTagLibrary" value="/SERVICE/com.sap.portal.navigation.navigationtaglibrary/taglib/TagLibrary.tld"/>
      </component-profile>
    </component>
  </components>
  <services/>
</application>
```

```java
package com.sap.portal.itelo.components;

import com.sapportals.portal.prt.ccomponent.);
import com.sapportals.portal.prt.resource.IResource;

public class TopNav extends AbstractPortalComponent {
  public void doContent(IPortalComponentRequest request, IPortalComponentResponse response) {
    response.include(request, request.getResource(IResource.CSS, "css/topnav.css"));
    response.include(request, request.getResource(IResource.SCRIPT, "scripts/topnav.js"));
    response.include(request, request.getResource(IResource.JSP, "jsp/components/topnav.jsp"));
  }
}
```
Example: “Pull-down Menu for the TLN iView” (cont.)

2. …and a JSP file with the menu’s HTML output:

```jsp
<%@ taglib uri="NavigationTagLibrary" prefix="nav" %>

<ul id="nav">
	<%<nav:iterateInitialNavNodes>
		<li>
			<%<nav:nvNodeAnchor navigationMethod="byEPCM" urlParameters="InitialNodeFirstLevel=true" /%>
		<%<nav:ifNavNodeHasChildren>
			<ul>
			<%<nav:iterateNavNodeChildren>
				<li><%<nav:nvNodeAnchor navigationMethod="byEPCM" /></li>
			<%</nav:iterateNavNodeChildren>
		<%</ul>
	</%<nav:ifNavNodeHasChildren>
	</%<li>
  </%<nav:iterateInitialNavNodes>
</%<ul>

(the opening and closing empty scriptlets are there to keep the footprint of the rendered HTML low)
3. Create a CSS file with at least the following selectors:

```
#nav, #nav ul {
    padding: 0;
    margin: 0;
    list-style: none;
}

#nav a {
    display: block;
    width: 10em;
}

#nav li {
    float: left;
    width: 10em;
}

#nav li ul {
    position: absolute;
    width: 10em;
    left: -999em;
}

#nav li:hover ul, #nav li:hover ul {
    left: auto;
}
```

(add some styling for the look & feel as well)
4. Create a Javascript file with the following output:

```javascript
function makeHoverable() {
    var sfEls = document.getElementById("nav").getElementsByTagName("LI");
    for (var i=0; i<sfEls.length; i++) {
        sfEls[i].onmouseover=function() {
            this.className+= " hover";
        }
        sfEls[i].onmouseout=function() {
            this.className=this.className.replace(/\bhover\b/, "");
        }
    }
}

if (EPCM.getUAType()==EPCM.MSIE && EPCM.getUAVersion() < 7) {
    EPCM.subscribeEvent('urn:com.sapportals.portal:browser', 'load', makeHoverable);
}
```

(we need this JS function since for MS IE < 7.0 the :hover pseudo class is only applicable to anchor tags)
Example: “Pull-down Menu for the TLN iView” (cont.)

Result

![Image of pull-down menu for TLN iView]
Completely rewritten framework

**Purpose:** For „more complicated“ adaptations that cannot be handled by the customization of the framework iViews and the theme, e.g. navigation with pull-down menus instead of TLN and DTN, or custom layouts with custom trays.

**Idea:** Develop the necessary components from scratch, using e.g. JSP tag libraries or APIs, and replace the corresponding components of the framework.

**Benefit:** no real modification, rather custom development.

**Pitfall:** Because of possible dependencies of the framework functionalities both good knowledge and a test strategy for upgrades is needed.
Completely rewritten framework
1. Changing the Portal Look and Feel
   1.1. Motivation and Benefits
   1.2. Examples

2. Fundamentals
   2.1. Portal Desktop and its Elements
   2.2. Themes and Style Sheets
   2.3. Portal Framework Pages

3. Changing the Portal Look and Feel – Step by Step

4. Summary
There are several ways to adapt the look & feel of the portal.

A number of tools assist you in branding the portal.

The branding possibilities range from “customization only” to “completely rewritten framework”

There are mainly two different concepts of how to set up a portal framework.

A custom developed portal framework can range from “100% Standard” to “100% Light”, differing mainly in the provision of typical portal functionality and performance/resource heaviness.
Thank you!