JHeadstart 11g

Applicaties genereren met custom look and feel

Na de komst van Oracle JDeveloper en ADF 11g is er ook een nieuwe release van Oracle JHeadstart beschikbaar. In dit artikel zal aan de hand van voorbeelden nader worden ingegaan op het gebruik van Velocity templates en een custom look and feel van de gegeneerde applicatie.

Wanneer vanuit JDeveloper met ADF wizards en drag and drop functionaliteit een applicatie wordt gebouwd, zal ADF code snippets aanmaken in JSF pagina's, paginadefinities (en de bindings daarbinnen) en managed bean definities. Al deze applicatie onderdelen worden ook aangemaakt bij het gebruik van JHeadstart. Op elk moment in het ontwikkelproces kan worden gestart met het gebruik van de visuele design-time tools en code editors in JDeveloper om functionaliteit te ontwikkelen die niet out-of-the-box door JHeadstart wordt gegenereerd. Wanneer er op die manier aanpassingen gedaan worden aan eerder gegenereerde pagina's, paginadefinities of de adfc-config.xml file en de applicatie wordt opnieuw gegenereerd vanuit JHeadstart, dan zouden alle gedane wijzigingen weer verloren gaan.

Er zijn drie keuzes om hiermee om te gaan:

- Niet meer genereren van de applicatie na de gedane aanpassingen;
- Uitschakelen van het genereren van de aangepaste bestanden. Op zowel service-level als group-level nivo kan in de Application Defintion Editor (Expert mode, Generation Settings) het genereren van specifieke output (bijvoorbeeld 'Generate Page Definition?') worden uitgeschakeld;
- Verplaats alle aanpassingen naar custom templates en configureer JHeadstart om die custom templates te gebruiken en blijf doorgaan met het genereren van de applicatie.

Zoals in de JHeadstart Developer's Guide is te lezen heeft deze derde optie de voorkeur, ook al is dit in het begin iets meer werk en moet je iets van de JHeadstart templating architectuur afweten.

In dit artikel wordt aan de hand van voorbeelden getoond hoe een applicatie met een custom look and feel wordt gegenereerd met gebruikmaking van custom template binding files, custom Velocity templates, ADF Faces skinning en ADF Faces Page (Fragment) templates. Hierbij wordt gebruik gemaakt van schermen gebaseerd op tabellen van het HR schema.
In de voorbeelden in dit artikel worden de volgende custom templates gebruikt:

```
..\ViewController\templates\customlaf\ApplicationLevel\misc\file\file-
Generator.vm
..\ViewController\templates\customlaf\ApplicationLevel\misc\file\home-
Page.vm
..\ViewController\templates\customlaf\ApplicationLevel\page\dataPage.vm
..\ViewController\templates\customlaf\ApplicationLevel\page\tablePage-
Content.vm
..\ViewController\templates\customlaf\GroupCountriesLevel\page\dataPage.
vm
```

Generator Templates

Oracle JHeadstart 11g genereert zeer geavanceerde ADF applicaties waarbij de JHeadstart Application Generator gebruik maakt van Velocity, een open source op Java-gebaseerde template engine van de Apache Foundation (zie http://velocity.apache.org/engine/releases/velocity-1.6.2/user-guide.html).
In directory ..\ViewController staan de bestanden die de basis vormen voor de look and feel van de door JHeadstart te genereren bestanden.
De JHeadstart templates worden opgeslagen in de templates directory van je JHeadstart project welke de volgende bestanden bevat:

- config/jag-config.xml. Te configureren instellingen voor de JHeadstart Application Generator.
- config/defaultTemplateBinding.jtp. JHeadstart Template Properties file die definieert welke Velocity templates gebruikt worden en waarvoor.
- default/*/*.vm. De default Velocity template bestanden die gebruikt worden voor het genereren van een applicatie.

Let op: Bij een upgrade naar een nieuwere versie van

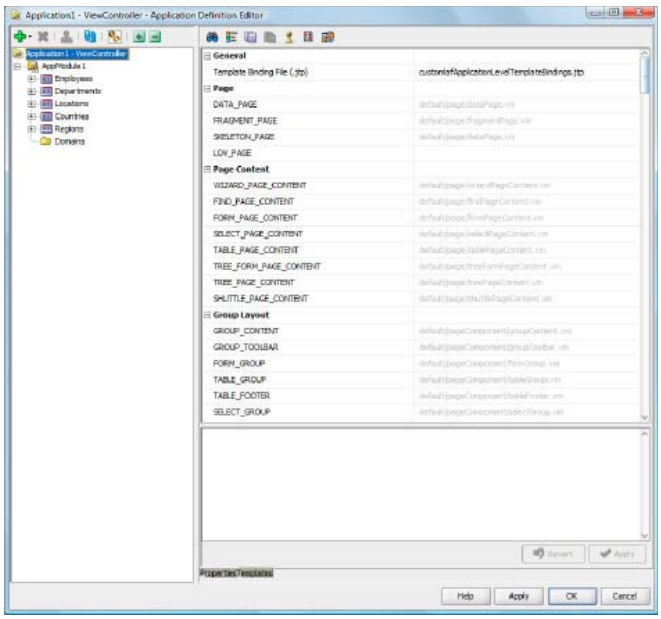
JHeadstart worden de default templates overschreven, zorg er daarom voor dat de custom templates in een aparte directory staan. Dit geeft tevens een sneller overzicht van alle template aanpassingen die gedaan zijn in het project.

Default look and feel

Indien de applicatie wordt gegenereerd met de default look and feel, dan is er per (top level) groep (bijvoorbeeld Countries) in de service (AppModule1) een tabblad gegenereerd. Een klik op het tabblad (menu item) zal een actie starten in de unbounded task flow, zoals bijvoorbeeld 'StartCountries'.
In afbeelding 1 is de Countries pagina te zien (door middel van pagina fragment CountriesTable.jsff), met een tabel met countries.

Het defaultTemplateBinding.jtp bestand beschrijft alle beschikbare templates en heeft pointers naar de lokatie van de templates. Een custom Template Binding File kan gebruikt worden voor de hele applicatie, of voor een service, of voor enig ander nivo. In de Application Definition Editor kan in de Templates tab de Template Binding File property ingesteld worden. De custom Template Binding File hoeft niet de hele lijst van templates te bevatten, het volstaat om alleen de aangepaste templates op te nemen. De andere templates worden overgenomen van de Template Binding File op een hoger nivo, of als die er niet is, van de JHeadstart default settings.

Naast het gebruik van een custom Template Binding File kunnen in de Application Definition Editor ook individuele templates voor een group of item worden overschreven. De defaultnaam van een bepaalde template is in grijs weergegeven



Afbeelding 1. Application Definition Editor, application level, templates

en kan in de Application Definition Editor gewijzigd worden, evenals de inhoud van de template.

In de Application Definition Editor is op het hoogste nivo (application level) de Template Binding File property ingesteld op 'customlafApplicationLevelTemplateBindings.jtp' (zie afbeelding 1), met als inhoud:

```
# --------------
# PAGE TEMPLATES
# --------------
DATA_PAGE=customlaf/ApplicationLevel/page/dataPage.vm
SKELETON_PAGE=customlaf/ApplicationLevel/page/dataPage.vm

# ----------------------
# PAGE CONTENT TEMPLATES
# ----------------------
TABLE_PAGE_CONTENT=customlaf/ApplicationLevel/page/tablePageContent.vm

#-------
# COMMON
#-------
FILE_GENERATOR=customlaf/ApplicationLevel/misc/file/fileGenerator.vm
```

Deze property instelling wordt als xml element <TemplateBinding> opgeslagen in bestand JHeadstartApplicationDefinition.xml in directory ..\ViewController\properties:

```
<?xml version="1.0" encoding="windows-1252"?>
<Application name="Application1 - ViewController" asfVersion="1.0"
templatesBaseDir="\templates\" viewPackage="application1.view"
viewType="adfFaces" nlsBundle="application1.view.ApplicationResources"
xmlns="http://www.oracle.com/jheadstart/applicationStructure" defaultGro
upRegionAccess="groupUIShell" regionTemplate="/common/pageTemplates/
JhsRegionTemplate.jspx" pageTemplate="/customlaf/pageTemplates/
JhsTreeMenuPageTemplate.jspx">
    <ServiceDefinitionRef location="AppModule1ServiceDefinition.xml"
name="AppModule1"/>
    <Templates>
      <TemplateBinding templateBindingsFile="customlafApplicationLevel
TemplateBindings.jtp"/>
    </Templates>
</Application>
```

Bij de groep Countries (group level) is de Template Binding File property ingesteld op "customlafGroupCountriesLevelTemplateBindings.jtp", met als inhoud:#

```
--------------
 # PAGE TEMPLATES
 # --------------
 DATA_PAGE=customlaf/GroupCountriesLevel/page/dataPage.vm
 SKELETON_PAGE=customlaf/GroupCountriesLevel/page/dataPage.vm
```

Deze property instelling wordt als xml element <TemplateBinding> bij Group "Countries" opgeslagen in bestand AppModule1ServiceDefinition.xml in directory ..\ViewController\properties:

```
<Group name="Countries" …>
        <Item attributeName="CountryId" …/>
        …
        <Templates>
            <TemplateBinding templateBindingsFile="/config/customlaf-
GroupCountriesLevelTemplateBindings.jtp"/>
        </Templates>
```

```
    </Group>
```

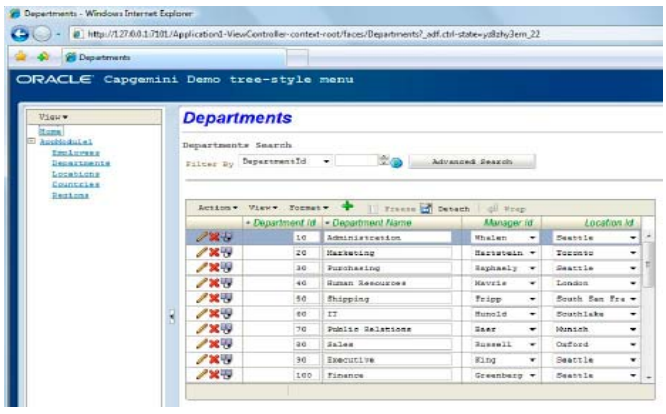
De default en custom Template Binding Files staan in directory ..\ViewController\templates\config. Dit is dan ook de "root" directory voor de Template Binding File property. Geeft men in deze property een relatief pad op dan is dat een subdirectory van de config directory.
Vanuit 'customlafApplicationLevelTemplateBindings.jtp' is er een verwijzing naar het gebruik van dataPage.vm (..\ViewController\templates\customlaf\ApplicationLevel\page\dataPage.vm) welke de volgende afwijkende inhoud heeft t.o.v. de default template:

```
<?xml version='1.0' #ENCODING_PROP()?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:trh="http://myfaces.apache.org/trinidad/html"
             xmlns:tr="http://myfaces.apache.org/trinidad">
  <jsp:directive.page contentType="text/html;charset=${JHS.encoding}"/>
  <!-- This is extra comment added at the begin of the customized data-
Page.vm -->
  <f:view>

    <af:pageTemplate id="pt" viewId="${JHS.application.pageTemplate}">
      …

    #if( $JHS.current.group.isUseAsStandalonePage )
      #JHS_PARSE(${JHS.page.contentTemplateIdentifier} ${JHS.current.
model})
    #else
      <f:facet name="pageContent">
        <af:group id="g1">
         <af:outputLabel value="$JHS.current.group.name" id="o11" styleC
lass="customlafOutputLabelBlue"/>
          <af:separator id="sep1" rendered="true"/>
          …
        </af:group>
      </f:facet>
    #end


    </af:pageTemplate>
  </f:view>
  <!-- This is extra comment added at the end of the customized dataPa-
ge.vm -->
</jsp:root>
```



Afbeelding 2. Pagina Departments (custom templates).

Doel hierbij is om voor alle groepen in de applicatie, in de bijbehorende pagina, aan de groepsnaam (getoond door middel van een af:outputLabel component) de styleclass "customlafOutputLabelBlue" te koppelen, voor gebruik met de skin "customklafskin" (zie afbeelding 3).

Vanuit 'customlafApplicationLevelTemplateBindings.jtp' is er een verwijzing naar het gebruik van tablePageContent.vm (..\ViewController\templates\customlaf\ApplicationLevel\page\tablePageContent.vm) welke de volgende afwijkende inhoud heeft t.o.v. de default template:

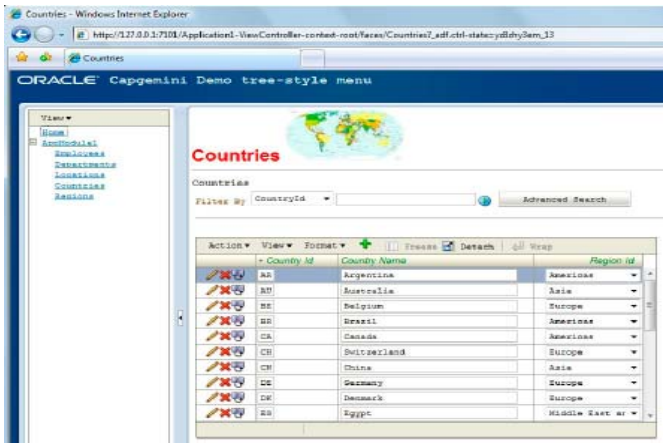
```
<!-- This is extra comment added at the begin of the customized table-
PageContent.vm -->
<f:facet name="pageContent">
  <af:panelHeader id="pcph" text="#PAGE_TITLE() Search" #CUR_GROUP_
PARTIAL_TRIGGERS()>
    #if (!$JHS.current.group.useAsLov)
    <f:facet name="toolbar">
       #JHS_PARSE("GROUP_TOOLBAR" ${JHS.current.model})
    </f:facet>
    <f:facet name="context">
       #JHS_PARSE("PARENT_CONTEXT" ${JHS.current.model})
    </f:facet>
    #end
    #JHS_PARSE("GROUP_CONTENT" ${JHS.current.model})
  </af:panelHeader>
</f:facet>
<!-- This is extra comment added at the end of the customized tablePa-
geContent.vm -->
```

Doel hierbij is om voor alle groepen in de applicatie, in de bijbehorende pagina, in het zoek gedeelte boven de tabel met data de term "Search" toe te voegen aan de groepsnaam (zie afbeelding 2). Vanuit 'customlafGroupCountriesLevelTemplateBindings.jtp' is er een verwijzing naar het gebruik van dataPage.vm (..\ViewController\templates\customlaf\GroupCountriesLevel\page\dataPage.vm) welke de volgende afwijkende inhoud heeft t.o.v. de default template:

```
<?xml version='1.0' #ENCODING_PROP()?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:trh="http://myfaces.apache.org/trinidad/html"
             xmlns:tr="http://myfaces.apache.org/trinidad">
  <jsp:directive.page contentType="text/html;charset=${JHS.encoding}"/>
  <!-- This is extra comment added at the begin of the customized data-
Page.vm -->
  <f:view>

    <af:pageTemplate id="pt" viewId="${JHS.application.pageTemplate}">
      …

    #if( $JHS.current.group.isUseAsStandalonePage )
      #JHS_PARSE(${JHS.page.contentTemplateIdentifier} ${JHS.current.
model})
    #else
      <f:facet name="pageContent">
        <af:group id="g1">
         <af:outputLabel value="$JHS.current.group.name" id="o11" styleC
lass="customlafOutputLabelRed"/>
          <tr:image source="/customlaf/images/countries.jpg"/>
          <af:separator id="sep1" rendered="true"/>
```



Afbeelding 3. Pagina Countries (custom templates) File Generator Template.

```
          …
        </af:group>
      </f:facet>
    #end


    </af:pageTemplate>
  </f:view>
  <!-- This is extra comment added at the end of the customized dataPa-
ge.vm -->
</jsp:root>
```

De templates die gekoppeld zijn aan het group level (bijvoorbeeld Countries) overschrijven dus de templates die gebruikt worden op een hoger nivo (bijvoorbeeld application level). Doel hierbij is om specifiek voor de groep 'Countries' in de bijbehorende pagina aan de groepsnaam (getoond door middel van een af:outputLabel component) de styleclass 'customlafOutputLabelRed' te koppelen, voor gebruik met de skin "customklafskin". Alsmede het tonen van een afbeelding (tr:image component) naast de groepsnaam (zie afbeelding 3).
JHeadstart gebruikt een speciaal template, default/misc/file/fileGenerator.vm, als een middel om aanvullende bestanden te genereren, welke niet direct gerelateerd zijn aan een groep. Elk bestand dat gegenereerd wordt door middel van de fileGenerator.vm template heeft zijn eigen template die gebruikt wordt om de inhoud van het bestand te genereren. Het pad en de naam van de template is voor elk bestand hardcoded in fileGenerator.vm.
Het configureren van JHeadstart om een custom File Generator Template te gebruiken kan in de Application Definition Editor in de Templates tab ingesteld worden op application of service level nivo (direct of via een template binding file).
Vanuit 'customlafApplicationLevelTemplateBindings.jtp' is er een verwijzing naar het gebruik van fileGenerator.vm (..\ViewController\templates\customlaf\ApplicationLevel\misc\file\fileGenerator.vm) welke de volgende afwijkende inhoud heeft t.o.v. de default template:

```
## Generate Index.jsp and Home Page
#set ($parsedContent = "#JHS_PARSE_NO_DEBUG('default/misc/file/indexJsp.
vm' ${JHS.service})")
$JHS.createFile("${JHS.htmlRootDir}/index.jsp", $parsedContent)
#set ($parsedContent = "#JHS_PARSE_NO_DEBUG('customlaf/ApplicationLevel/
misc/file/homePage.vm' ${JHS.service})")
$JHS.createOrReplaceFile("${JHS.htmlRootDir}${JHS.application.commonPa-
gesDir}Home.jspx", $parsedContent)
```

Vanuit 'fileGenerator.vm' is er een verwijzing naar het gebruik van homePage.vm (..\ViewController\templates\customlaf\ApplicationLevel\misc\file\homePage.vm) welke de volgende inhoud heeft:

```
## revision_history
## 09-sep-2008    Steven Davelaar
## 1.1             R11 upgrade
## 17-jun-2007    Steven Davelaar
## 1.0             Initial creation
<?xml version='1.0' #ENCODING_PROP()?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:trh="http://myfaces.apache.org/trinidad/html"
          xmlns:tr="http://myfaces.apache.org/trinidad">
  <jsp:directive.page contentType="text/html;charset=${JHS.encoding}"/>
  <!-- This is extra comment added at the begin of the customized home-
Page.vm -->
  <f:view>
    <af:pageTemplate viewId="${JHS.application.pageTemplate}">
      <f:attribute name="pageTitle" value="Welcome #{jhsUser!=null ?
 jhsUser.displayName : facesContext.externalContext.userPrincipal.name}
 to the Capgemini Demo Application!"/>
      #if (${JHS.application.runtimeCustomizationMenuAllowed})
        <f:attribute name="dynamicMenu" value="true"/>
        <f:attribute name="headerImage1" value="#{jhsDynamicMenu.cur-
rentModule.uisHeaderImage1}"/>
        <f:attribute name="headerImage2" value="#{jhsDynamicMenu.cur-
rentModule.uisHeaderImage2}"/>
      #else
        <f:attribute name="menuModel" value="#{RootMenu}"/>
        <f:attribute name="menuStartLevel" value="0"/>
      #end

      <f:facet name="pageContent">
        <af:panelGroupLayout layout="vertical" halign="center">
          <af:panelHeader text="Welcome #{jhsUser!=null ? jhsUser.dis-
playName :
 facesContext.externalContext.userPrincipal.name} to the Capgemini Demo
Application!">
          </af:panelHeader>
          <af:spacer width="5" height="200"/>
          <tr:image source="/customlaf/images/capgemini.jpg"/>
        </af:panelGroupLayout>
      </f:facet>

    </af:pageTemplate>
  </f:view>
  <!-- This is extra comment added at the end of the customized home-
Page.vm -->
</jsp:root>
```

Hiermee wordt pagina Home.jspx gegenereerd:

```
<?xml version='1.0' encoding="windows-1252"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:f="http://java.sun.com/jsf/core"
```

```
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:trh="http://myfaces.apache.org/trinidad/html"
          xmlns:tr="http://myfaces.apache.org/trinidad">
  <jsp:directive.page contentType="text/html;charset=windows-1252"/>
  <!-- This is extra comment added at the begin of the customized home-
Page.vm -->
  <f:view>
    <af:pageTemplate viewId="/customlaf/pageTemplates/
JhsTreeMenuPageTemplate.jspx">
      <f:attribute name="pageTitle"

value="Welcome #{jhsUser!=null ? jhsUser.displayName : facesContext.
externalContext.userPrincipal.name} to the Capgemini Demo
Application!"/>
      <f:attribute name="menuModel" value="#{RootMenu}"/>
      <f:attribute name="menuStartLevel" value="0"/>
      <f:facet name="pageContent">
        <af:panelGroupLayout layout="vertical" halign="center">

<af:panelHeader text="Welcome #{jhsUser!=null ? jhsUser.displayName :
facesContext.externalContext.userPrincipal.name} to the Capgemini Demo
Application!"></af:panelHeader>
          <af:spacer width="5" height="200"/>
          <tr:image source="/customlaf/images/capgemini.jpg"/>
        </af:panelGroupLayout>
      </f:facet>
    </af:pageTemplate>
  </f:view>
  <!-- This is extra comment added at the end of the customized home-
Page.vm -->
</jsp:root>
```

Doel hierbij is om de pagina titel alsmede de panelHeader tekst van het rechter panel, anders in te stellen. Alsmede het tonen van een afbeelding (tr:image component) in het rechter panel.

Custom look and feel

De look and feel van een applicatie kan veranderd worden door het gebruik van een tweetal concepten:

ADF Faces Skinning.
Met een custom skin, kunnen de layout karakteristieken (zoals fonts, colors, icons, images, margins, enz.) gewijzigd worden, welke meestal gedefinieerd worden door middel van cascading style sheets. Het voordeel hierbij is dat de gewenste look and feel eenmalig in de skin wordt gedefinieerd en gebruikt kan worden door één of meerdere applicaties. Bij het gebruik van een skin bij een bestaande ADF Faces applicatie, hoeft de applicatie zelf niet gewijzigd te worden. Voor nadere informatie hierover verwijs ik naar de later genoemde documentatie.
De skin welke ADF Faces gebruikt, is vastgelegd in een bestand genaamd trinidad-config.xml in de WEB-INF directory:

```
<?xml version="1.0" encoding="windows-1252"?>
<trinidad-config xmlns="http://myfaces.apache.org/trinidad/config">

  <skin-family>customlafskin</skin-family>

</trinidad-config>
```

De skin met naam 'customlafskin' welke in dit voorbeeld gebruikt wordt is vastgelegd in een nieuw aangemaakt bestand in de WEB-INF directory genaamd trinidad-skins.xml, met de volgende inhoud:

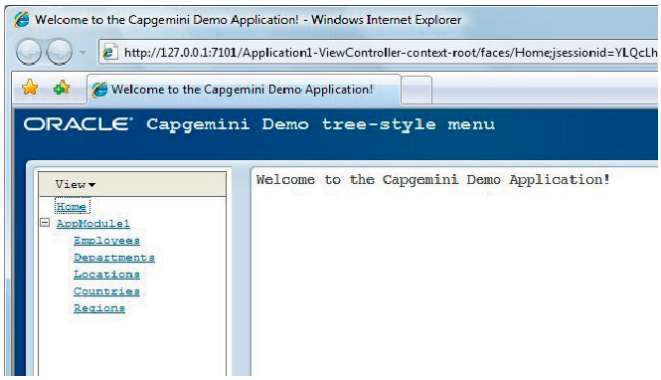
```
<?xml version="1.0" encoding="windows-1252" ?>
<skins xmlns="http://myfaces.apache.org/trinidad/skin">
  <skin>
    <id>customlafskin.desktop</id>
    <family>customlafskin</family>
    <extends>blafplus-rich.desktop</extends>
    <style-sheet-name>customlaf/css/customlaf.css</style-sheet-name>
  </skin>
</skins>
```

De customlafskin is gerelateerd aan de cascading stylesheet ..\ViewController\public_html\customlaf\css\customlaf.css met de volgende inhoud:

```
.AFDefaultFontFamily:alias {
font-family:'Courier New', Courier, monospace;
}
af|outputLabel
{
   color:Green; font-family:Arial, Helvetica, sans-serif; font-
size:small;font-style:italic;
}
af|outputLabel.customlafOutputLabelRed
{
    color:Red; font-size:x-large; font-weight:bold; font-style:normal;
}
af|outputLabel.customlafOutputLabelBlue
{
    color:Blue; font-size:x-large; font-weight:bold;
}
```

De regels in de cascading stylesheet bevatten een skin style selector, welke een element identificeert en een set van style properties, welke het uiterlijk van een component beschrijven. ADF Faces components bevatten twee categorieën van skin style selectors:

- Global selectors. Global selectors bepalen de style properties voor meerdere ADF Faces componenten. Wanneer de global selector naam eindigd in de :alias pseudo-class, dan wordt die selector naar alle waarschijnlijkheid gebruikt bij andere component specifieke selectors en zal het dus effect hebben op het uiterlijk van meer dan één component. Zo wordt bijvoorbeeld door nagenoeg alle componenten de .AFDefaultFontFamily:alias gebruikt om het lettertype (font family) te specificeren. Als een eigen custom skin deze selector overschrijft, dan zal dat effect hebben op alle componenten die deze selector definitie gebruiken.
- Component selectors. Component-specifieke selectors zijn selectors die een skin kunnen koppelen aan een specifieke ADF Faces component. Met specifiek wordt dan bijvoorbeeld bedoeld dat de component een bepaalde styleClass heeft of dat de component onderdeel uitmaakt van een bepaalde andere component. In bovenstaand css bestand is het default lettertype voor componenten op 'Courier New' ingesteld (zie bijvoorbeeld tekst 'Capgemini Demo tree-style



Afbeelding 4. Pagina Home (met tree-style menu en custom templates).

menu'. Componenten van het type af:outputLabel worden echter in 'Arial', cursief (italic) en kleur groen weergegeven. Componenten van het type af:outputLabel die de styleClass 'customlafOutputLabelRed' hebben erven de layout eigenschappen van af:outputLabel (zoals 'Arial'), maar worden in kleur rood weergegeven (zie afbeelding 3). Componenten van het type af:outputLabel die de styleClass 'customlafOutputLabelBlue' hebben erven de layout eigenschappen van af:outputLabel (zoals "Arial" en cursief), maar worden in kleur blauw weergegeven (zie afbeelding 2).

ADF Faces Page (Fragment) Templates

Wanneer groepen gegenereerd worden als herbruikbare regions met pagina fragmenten (zoals in dit voorbeeld), dan worden er twee pagina templates gebruikt:

- Page Template. De algemene page template welke generieke componenten bevat zoals: application title, branding, logo's, global menu options, application menu en footer. JHeadstart wordt geleverd met twee page templates:
  1. JhsPageTemplate.jspx (de standaard). Deze template gebruikt een tab-style menu.
  2. JhsTreeMenuPageTemplate.jspx. Deze template gebruikte een tree-style menu.

  In de Application Definition Editor is op application level de property "Page Template" in te stellen.
- Region Template (JhsRegionTemplate.jspx). De template welke gebruikt wordt om pagina fragmenten van de groep te genereren, deze template maakt de breadcrumbs (menu pad) zichtbaar. In de Application Definition Editor is op application level de property "Region Template" in te stellen. In dit voorbeeld is gekozen voor de default zijnde: /common/pageTemplates/JhsRegionTemplate.jspx

De custom page template ..\ViewController\public_html\customlaf\pageTemplates\JhsPageTemplate.jspx in dit voorbeeld heeft de volgende afwijkende inhoud heeft t.o.v. de default:

```
<af:outputText noWrap="true" value="Capgemini Demo tab-style menu"/>
```

Hiermee wordt in de applicatie, boven het menu deel, steeds de tekst 'Capgemini Demo tab-style menu' getoond.
De custom page template ..\ViewController\public_html\customlaf\pageTemplates\JhsTreeMenuPageTemplate.jspx in dit voorbeeld heeft de volgende afwijkende inhoud heeft t.o.v. de default:

```
<af:outputText noWrap="true" value="Capgemini Demo tree-style menu"/>
```

Hiermee wordt in de applicatie, boven het menu deel, steeds de tekst 'Capgemini Demo tree-style menu' getoond (zie afbeelding 4).

ADF registreert pagina templates in een xml bestand genaamd \ViewController\src\META-INF\pagetemplate-metadata .xml. In dit voorbeeld heeft dit bestand de volgende inhoud:

```
<?xml version="1.0" encoding="windows-1252" ?>
<pageTemplateDefs xmlns="http://xmlns.oracle.com/adf/faces/rich/pagetemplate">
  <pagetemplate-jsp-ui-def>/customlaf/pageTemplates/JhsTreeMenuPageTemplate.jspx</pagetemplate-jsp-ui-def>
  <pagetemplate-jsp-ui-def>/common/pageTemplates/JhsRegionTemplate.jspx</pagetemplate-jsp-ui-def>
  <pagetemplate-jsp-ui-def>/customlaf/pageTemplates/JhsPageTemplate.jspx</pagetemplate-jsp-ui-def>
</pageTemplateDefs>
```

JHeadstart biedt mogelijkheden om eenvoudig een skin, en/of een page template te wijzigen op zowel design-time als runtime (door de eindgebruiker).

Samenvatting

In dit artikel is aan de hand van voorbeelden getoond hoe een applicatie met een custom look and feel kan worden gegenereerd met behulp van Oracle JHeadstart 11g in combinatie met ADF 11g en met gebruikmaking van custom template binding files, custom Velocity templates, ADF Faces skinning en ADF Faces Page (Fragment) templates.

Referenties

- *ORACLE JHEADSTART 11g for ADF (RELEASE 11.1.1 - DRAFT), DEVELOPER'S GUIDE, MARCH 2009.*
- *Oracle® Fusion Middleware, Web User Interface Developer's Guide for Oracle Application Development Framework, 11g Release 1 (11.1.1), B31973-02, Hoofdstuk "Customizing the Appearance Using Styles and Skins", November 2008*
- *Meer info op: http://www.oracle.com/technology/products/jheadstart/index.html*

Marc Lameriks is Oracle CoP leader en thought leader van de Oracle ADF Focus Group bij Capgemini en op projecten werkzaam als Software Architect (e-mail: marc.lameriks@capgemini.com).