Scoren met JSF

Project met JDeveloper en JavaServer Faces

Het is intussen een traditie bij Capgemini om gedurende een groot voetbaltoernooi een bedrijfsbrede voetbalpool te bouwen. Tijdens het Europees kampioenschap van 2004 is dit aangegrepen om de mogelijkheden van HTML DB (nu: Application Express) te onderzoeken. Voor het wereldkampioenschap van dit jaar is hetzelfde gedaan voor JDeveloper in combinatie met JavaServer Faces (JSF). Dit artikel is een verslag van de bevindingen van de bouw en het gebruik van een JSF voetbalpool.

JavaServer Faces is een framework voor server-side user interface componenten ten behoeve van web applicaties die gebaseerd zijn op de Java technologie. Het bevat een set van API's voor weergave van user interface componenten met mechanismen voor afhandeling van events, validaties en paginanavigatie. Daarnaast bevat het een JavaServer Pages (JSP) custom tag library om JSF-componenten te kunnen gebruiken in een JSP-pagina.

Open standaard

JavaServer Faces technologie is ontwikkeld onder JSR-127. Java Specification Request (JSR) is de manier om via een formele weg wijzigingen en/of aanvullingen door te kunnen voeren in de Java Technologie standaard. Het feit dat JSF via een JSR is ontwikkeld, betekent dat het JSF framework een officiële standaard betreft en zal als zodanig door de Java Community en Vendors sneller geadopteerd kunnen worden. Deze officiële status ontbreekt overigens voor menig ander framework.

Oracle ADF Faces

Oracle ADF Faces is een componentenbibliotheek, die volledig voldoet aan de JSF standaard, en die een uitgebreide set UI-componenten biedt voor JSF applicatie-ontwikkeling. Oracle ADF Faces levert bijvoorbeeld geavanceerde tabellen, kleur en datum kiezers en algemene componenten zoals menu's en knoppen. Deze componenten kunnen in elke IDE worden gebruikt, mits die ook JSF ondersteund. Oracle ADF Faces verzekert een consistente *look and feel* voor je applicatie, waardoor er meer op user interface interactie gefocust kan worden dan op look and feel.

Bouwen van de voetbalpool

Met het bouwen van de voetbalpool is onderzocht of het eenvoudig is met behulp van ADF Faces een rijke grafische user interface (GUI) op te zetten. Interessant was ook om te onderzoeken of er een groot verschil was met de voorloper van Oracle ADF Faces. Met de UIX-technologie liep Oracle in voorgaande versies van JDeveloper al vooruit op JSF.

> ***Oracle ADF Faces is een componentenbibliotheek, die volledig voldoet aan de JSF standaard***

Het belangrijkste in een voetbalpool is dat iemand zijn persoonlijke voorspellingen kan ingeven. Dit betekent dat er user-authenticatie opgezet moest worden. Vervolgens moest op elke pagina de eigen gegevens van deze persoon worden getoond (bijvoorbeeld: eigen voorspellingen of eigen positie op de ranglijst). Hiervoor dienden de inloggegevens door de applicatie heen te worden meegenomen. Een volgend punt was dat de generieke onderdelen eenmalig gedefinieerd moesten worden. In de rest van dit artikel zullen deze verschillende onderdelen worden besproken.

User authenticatie en autorisatie

De voetbalpool kent drie autorisatieniveaus. Allereerst het algemeen toegankelijke deel dat niet persoons- of rolgebonden is. Daarnaast het persoonsgebonden deel, dat is bedoeld om deelnemers voorspellingen in te laten voeren. Vervolgens een rolgebonden deel, dat is bedoeld voor beheerwerkzaamheden, zoals het invoeren van uitslagen. De benodigde authenticatie en auto-

risatie zijn geregeld volgens de JAAS-standaard. In OC4J is container managed security geconfigureerd met een custom login module. Hierbij wordt login informatie geauthentiseerd tegen gebruikers informatie in database-tabellen. Daartoe is gebruik gemaakt van een custom login module die als voorbeeld bij het how-to document van de Oracle JAAS implementatie beschikbaar is. Zie `http://www.oracle.com/technology/products/jdev/howtos/10g/jaassec/index.htm` voor een duidelijke stap-voor-stap beschrijving. Om deze security-functionaliteit ook in de embedded OC4J van JDeveloper te configureren is de beschrijving gevolgd op `http://stegemanoracle.blogspot.com/2006_02_01_stegemanoracle_archive.html`. In tegenstelling tot BASIC authenticatie kan, indien er gebruik wordt gemaakt van FORM based authenticatie, een applicatiespecifieke login-pagina worden samengesteld. In deze login-pagina kunnen JSP-elementen en JSTL-tags worden gebruikt. Door een beperking in relatie met JSF en container security kunnen op de login-pagina echter geen JSF-componenten worden gebruikt. De look-and-feel die op overige pagina's wordt samengesteld met behulp van JSF-componenten zal dus moeten worden nagebouwd op de login-pagina met standaard HTML en JSP-elementen. Voor de voetbalpool is gekozen voor de FORM based methode.

Voor de aanmelding van nieuwe deelnemers bij de voetbalpool zijn twee JSF-webpagina's gebouwd; één voor het invoeren van een aanmelding en één voor het activeren van de aanmelding. Na het invoeren en submitten van de aanmelding wordt het opgegeven wachtwoord gecodeerd en met de overige gegevens opgeslagen in de database. Dan wordt een bevestigingsmail voor de deelnemer gegenereerd en verstuurd, met een gepersonaliseerde link naar de deelname activeringswebpagina. De link bevat een parameter, waarmee in de activeringswebpagina de deelnemer wordt geïdentificeerd. Met deze techniek wordt de geldigheid van het emailadres en goedkeuring van de eigenaar van het emailadres gewaarborgd, waarna de deelname wordt geactiveerd.

Afbeelding 1. Inloggen deelnemer webpagina.

Persoonsgebonden webpagina's

Voor het merendeel van de webpagina's in de voetbalpool is het noodzakelijk gegevens te tonen behorend bij de deelnemer die is ingelogd. Hiertoe is onder andere het mechanisme van Managed beans gebruikt. *Managed beans* zijn applicatie *JavaBeans* die geregistreerd worden in de JSF `faces-config.xml` file. Een ander mechanisme dat in dit verband vermeldt kan worden, zijn de *Backing beans*. Dit zijn managed beans die logica en property's (eigenschappen) bevatten voor enkele of alle UI-componenten op een JSF-pagina. Als bijvoorbeeld validatie en het afhandelen van events (gebeurtenissen) in een backing bean wordt opgenomen, dan heeft de code programmatisch toegang tot de UI componenten op de JSF pagina. Alle backing beans zijn managed beans. Er zijn ook managed beans die geen backing bean zijn. Dit is het geval bij een *JavaBean* die geen property's en property getter en setter methoden voor UI componenten bevat, maar waarbij de bean wel geregistreerd is in `faces-config.xml`. Een veel gebruikte toepassing waarbij een managed bean gebruikt wordt die geen backing bean is, is het opvragen van geauthentificeerde gebruikers informatie vanuit de container security.

Het belangrijkste in een voetbalpool is dat iemand zijn persoonlijke voorspellingen kan ingeven

Wanneer op runtime er aan de managed bean wordt gerefereerd vanaf een pagina middels een JSF EL waarde of method binding expression, zal de JSF-implementatie automatisch de

Afbeelding 2. Overzicht geregistreerde Managed Beans

	Positie	Naam	Practice	Punten
1	2	Russell Stocks	P70	65
7	38	Marc Lameriks	P70	58
28	38	Marcel van den Berg	P70	58
51	74	Cindy Kommerkamp	P70	55
34	91	Arjan Zeeman	P70	54
19	91	Jan Maarten van der Valk	P70	54
8	115	Renate Tjee	P70	53
12	137	Sjoerd Aalbers	P70	52
19	165	Leon Smiers	P70	51
16	234	Nienke Vonk	P70	48
43	289	Mirjana Vucic	P70	46
27	305	Ruben Spekle	P70	45
4	328	Ingrid Dahler	P70	44
14	359	Richard Kraan	P70	42
–	473	Gert Jan de Pender	P70	0

Afbeelding 3. Poolstand webpagina.

bean instantiëren, en haar van gedeclareerde default waarden voorzien. Vervolgens wordt de managed bean in de scope geplaatst zoals gedefinieerd in `faces-config.xml`.
Om in de voetbalpool gegevens te tonen behorend bij de deelnemer die is ingelogd is er een managed bean geregistreerd, genaamd UserInfo (zie afbeelding 2). Hierbij is session als scope gebruikt. Er zijn een viertal scopes op te geven:

- *application*: De bean is beschikbaar voor de duur van de web applicatie. Dit is handig voor global beans zoals LDAP-directory's.
- *request:* De bean is beschikbaar vanaf het tijdstip van instantiatie, totdat een response is teruggestuurd naar de client. Dit is gewoonlijk de levensduur van de huidige pagina. Backing beans voor pagina's gebruiken meestal deze scope.
- *session:* De bean is beschikbaar voor de client gedurende de hele client-sessie.
- *none:* De bean wordt elke keer geïnstantieerd, als eraan wordt gerefereerd.

De bean is geregistreerd in het 'overview'-tabblad van bestand `faces-config.xml`. Met dit JSF-configuratiebestand worden JSF-applicaties resources geregistreerd, zoals managed beans en navigatie regels (navigation rules).

In onderstaande code wordt de managed bean UserInfo gedefinieerd. De deelnemer en zijn rol worden met gebruikmaking van de methods getRemoteUser() respectievelijk isUserInRole() bepaald.

```
package com.capgemini.wkpool2006.view;

…

public class UserInfo
{
  private String deelnemer = null;
  private boolean isAdministrator = false;

  private static final String ROLE_USER = new String("User");
  private static final String ROLE_ADMINISTRATOR = new
String("Administrator");

  public UserInfo()
  {
    FacesContext ctx = FacesContext.getCurrentInstance();
    ExternalContext ectx = ctx.getExternalContext();

    //Ask the container who the user logged in as
    deelnemer = ectx.getRemoteUser();

    //Default the value if not authenticated
    if (deelnemer == null || deelnemer.length() == 0)
    {
      deelnemer = "Not Authenticated";
    }
    else
    {
      isAdministrator = ectx.isUserInRole(ROLE_ADMINISTRATOR);

    }
  }

  public String getDeelnemer()
  {
    StringBuffer name = new StringBuffer(deelnemer);
    return name.toString();
  }

  public boolean isAdministrator()
  {
    return isAdministrator;
  }
}
```

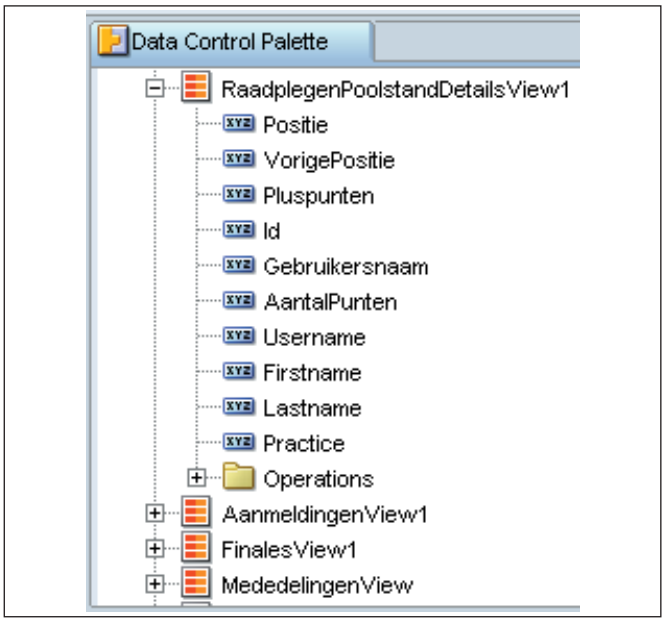
In afbeelding 3 is een voorbeeld te zien van een webpagina waarop de deelnemer zijn positie in de pool kan zien (zie A) en de posities van overige deelnemers kan opvragen (zie B).

In de hierna volgende code, een gedeelte van bestand RaadplegenPoolstand.jsp is te zien dat, voor het tonen van de positie van een deelnemer in de pool op de Poolstand-webpagina, het ADF binding mechanisme wordt gebruikt.

```
<h:panelGrid columns="1" border="0" …>
  <af:outputText styleClass="TEXT"
                 value="#{bindings.Firstname.inputValue}, je staat op
                                                              positie
                        #{bindings.Positie.inputValue} met
                        #{bindings.AantalPunten.inputValue}
                             punten."/>
  <af:outputText/>
  <af:outputText/>
</h:panelGrid>
<h:panelGrid columns="3" border="0" …>
  <af:outputText styleClass="TEXT"
                 value="Positie vorige speeldag:"/>
  <af:outputText styleClass="TEXT"
                 value="#{bindings.VorigePositie.inputValue}"/>
  <af:outputText/>
  <af:outputText styleClass="TEXT"
                 value="Punten op laatste speeldag:"/>
  <af:outputText styleClass="TEXT"
                 value="#{bindings.Pluspunten.inputValue}"/>
  <af:outputText/>
</h:panelGrid>
```

Aan de pagina zijn databound UI componenten toegevoegd met behulp van de Data Control Palette (zie afbeelding 4). Zo zijn bijvoorbeeld de attributen 'Firstname' en 'Positie' van het view object RaadplegenPoolstandDetailsView gekoppeld aan de pagina via outputText componenten.

Wanneer in JDeveloper aan een webpagina een databound UI component wordt toegevoegd met behulp van de Data Control Palette, zal JDeveloper metadata toevoegen aan het bestand `PageNamePageDef.xml` (het paginadefinitie XML-bestand). Deze PageDef XML bestanden definiëren de Oracle ADF binding container voor elke webpagina in de applicatie. De binding container verzorgt de toegang tot de bindings binnen de pagina. Daarom is er maar één PageDef XML bestand voor elke databound webpagina.



Afbeelding 4. Data Control Palette.

```
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="10.1.3.36.73" id="RaadplegenPoolstandPageDef"
                Package="com.capgemini.wkpool2006.view.pageDefs"
                ControllerClass="com.capgemini.wkpool2006.view.
                                 RaadplegenPoolstandPageController">
  <parameters/>
  <executables>
    …
    <iterator id="RaadplegenPoolstandDetailsView1Iterator"
              RangeSize="10"
              Binds="RaadplegenPoolstandDetailsView1"
              DataControl="WKPool2006ServiceDataControl"/>
  </executables>
  <bindings>
    …
    <attributeValues id="Positie"
          IterBinding="RaadplegenPoolstandDetailsView1Iterator">
      <AttrNames>
        <Item Value="Positie"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="VorigePositie"
          IterBinding="RaadplegenPoolstandDetailsView1Iterator">
      <AttrNames>
        <Item Value="VorigePositie"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="Pluspunten"
           IterBinding="RaadplegenPoolstandDetailsView1Iterator">
      <AttrNames>
        <Item Value="Pluspunten"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="AantalPunten"
           IterBinding="RaadplegenPoolstandDetailsView1Iterator">
      <AttrNames>
        <Item Value="AantalPunten"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="Firstname"
          IterBinding="RaadplegenPoolstandDetailsView1Iterator">
      <AttrNames>
        <Item Value="Firstname"/>
      </AttrNames>
    </attributeValues>
    …
  </bindings>
</pageDefinition>
```

Bestand RaadplegenPoolstandPageDef.xml

De gegevens op de webpagina zijn gebaseerd op de deelnemer die is ingelogd. Daarom is er een *whereclauseparameter* gebruikt (genaamd deelnemer) bij het onderliggende view-object om de deelnemer in te stellen. Het volgende SQL statement behoort bij view object RaadplegenPoolstandDetailView.

```
SELECT Poolstand.POSITIE,
       Poolstand.VORIGE_POSITIE,
       Poolstand.PLUSPUNTEN,
       Poolstand.ID,
       Poolstand.GEBRUIKERSNAAM,
       Poolstand.AANTAL_PUNTEN,
       Aanmeldingen.USERNAME,
       Aanmeldingen.FIRSTNAME,
       Aanmeldingen.LASTNAME,
       Aanmeldingen.PRACTICE
FROM POOLSTAND Poolstand, AANMELDINGEN Aanmeldingen
WHERE Poolstand.GEBRUIKERSNAAM = Aanmeldingen.USERNAME
AND Poolstand.GEBRUIKERSNAAM LIKE :deelnemer
```

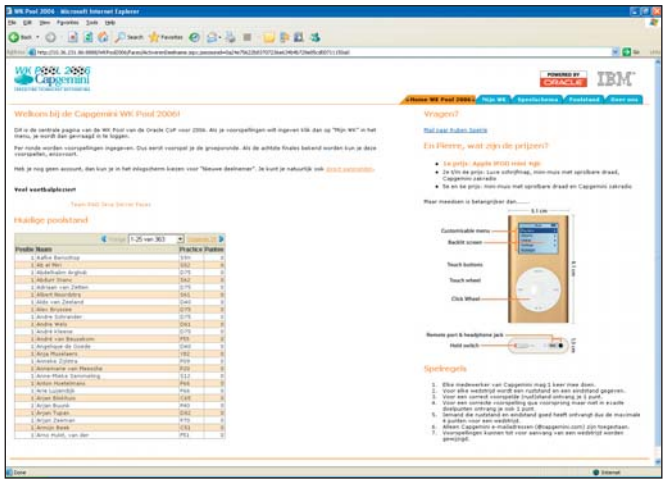
Omdat bij het openen van de Poolstand webpagina het gewenst is dat de deelnemer direct zijn positie in de pool ziet (zie afbeelding 3 punt A), dient voorafgaand aan het tonen van de webpagina de *whereclauseparameter* ingesteld te worden en de query uitgevoerd te worden. In onderstaande code wordt met gebruikmaking van de method filterPoolstandOpdeelnemer in de applicatiemodule (WKPool2006ServiceImpl), de deelnemer doorgegeven aan het view-object.

```
public class WKPool2006ServiceImpl extends ApplicationModuleImpl
implements WKPool2006Service
{
  /**This is the default constructor (do not remove)
  */
  public WKPool2006ServiceImpl()
  {
  }

  …

  public void filterPoolstandOpDeelnemer(String deelnemer)
  {
    getRaadplegenPoolstandDetailsView1().setdeelnemer(deelnemer);
    getRaadplegenPoolstandDetailsView1().executeQuery();
  }

  …
}
```

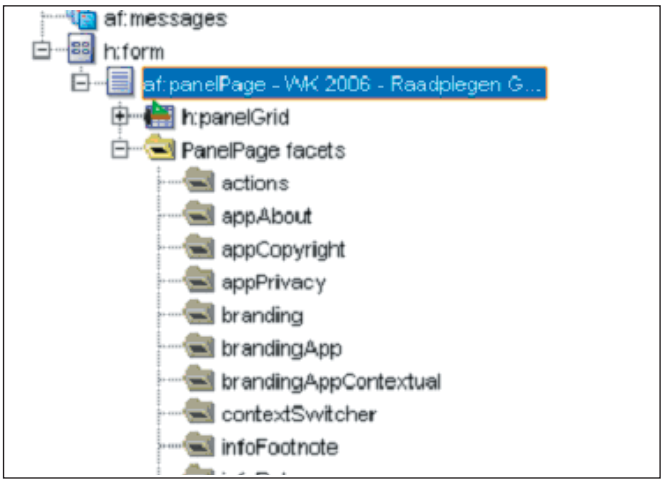
Om direct de positie in de pool van de deelnemer zichtbaar te maken, is er gebruik gemaakt van de zogenaamde ADF-lifecycle en de hierboven besproken pagina definitie (PageDef). De ADF-lifecycle regelt de voorbereiding en wijzigingen van het data model, validatie van de data in de model laag, en het uitvoeren van methods op de business laag. De ADF lifecycle gebruikt de paginadefinitie (PageDef) voor het ophalen en tonen van data, waarbij de data lokaal worden bewaard voordat de pagina wordt gerenderd. Met het voorkomen van extra round trips naar de database voordat de webpagina wordt gerenderd, verbetert de lifecycle de performance van de applicatie gedurende het rendering proces. In de paginadefinitie `RaadplegenPoolstandPageDef.xml` is een ControllerClass gedefinieerd, genaamd "com.capgemini.wkpool2006.view. RaadplegenPoolstandPageController". Deze zal dan door de ADF-controller worden gebruikt wanneer die een PageController object nodig heeft voor deze bindingContainer. In onderstaande code is te zien dat method prepareModel wordt gebruikt om de method filterPoolstandOpDeelnemer aan te roepen. Hierbij wordt managed bean UserInfo gebruikt om uit te vragen wie de deelnemer is. Het bestand RaadplegenPoolstandPageController.java ziet er als volgt uit:

```
public class RaadplegenPoolstandPageController extends PageController
{
  public RaadplegenPoolstandPageController()
  {
  }

  public void prepareModel(LifecycleContext context)
  {
    super.prepareModel(context);

    AdfFacesContext afContext = AdfFacesContext.getCurrentInstance();

    // deelnemer ophalen via managed bean
    FacesContext ctx = FacesContext.getCurrentInstance();
    Application app = ctx.getApplication();
    ValueBinding bind = app.createValueBinding("#{UserInfo}");
    Object o = bind.getValue(ctx);

    UserInfo userInfo = (UserInfo)o ;
    String deelnemer = userInfo.getDeelnemer();

    //Default the value if not authenticated
    if (deelnemer == null || deelnemer.length() == 0)
    {
      deelnemer = "Not Authenticated";
    }

    WKPool2006ServiceImpl wkPoolService = (WKPool2006ServiceImpl)
    context.getBindingContext().
      findDataControl("WKPool2006ServiceDataControl")
      .getDataProvider();

    wkPoolService.filterPoolstandOpDeelnemer(deelnemer);
    …
  }
}
```

In afbeelding 3 punt B is te zijn dat na het opvragen van de posities van overige deelnemers, het record van de deelnemer die is ingelogd in oranje wordt weergegeven. De attributen 'Positie', 'Naam', 'Practice' en 'Punten' van het view object RaadplegenPoolstandView zijn gekoppeld aan de pagina via outputText componenten. Bij de outputText component is het attribuut inlineStyle gebruikt om een kleur en letterype te



Afbeelding 5. WK Pool 2006 webpagina.



Afbeelding 6. Een gedeelte van de lijst van facets wordt weergegeven voor een PanelPage.

koppelen aan de inhoud van de component. Een gedeelte van het bestand RaadplegenPoolstand.jsp ziet er als volgt uit:

```
<h:panelGrid columns="1" border="0" …>
  <af:outputText styleClass="TEXT"
                 value="Laatst meegetelde uitslag:
                     #{bindings.UitslagenView1Wedstrijd.inputValue}"/>
  <af:table value="#{bindings.RaadplegenPoolstandView1.collectionModel}"
…>
    …
    <af:column headerText="Positie"
               sortable="false">
      <afh:tableLayout halign="right">
        <afh:rowLayout valign="middle">
          <afh:cellFormat>
            <af:outputText styleClass="TEXT"
                           value="#{row.Positie}"
                             inlineStyle="#{(row.Lastname eq bindings.
                                             Lastname.inputValue) and
                                           (row.Firstname eq bindings.
                                              Firstname.inputValue)
                                           ?'color:orange;font-weight:
                                                          bold;':''}">
              <f:convertNumber groupingUsed="false"
                               pattern="#{bindings.
RaadplegenPoolstandView1.formats.Positie}"
              />
            </af:outputText>
          </afh:cellFormat>
        </afh:rowLayout>
      </afh:tableLayout>
    </af:column>
    <af:column headerText="Naam"
               sortable="false"
               width="390px">
      <afh:tableLayout halign="left">
        <afh:rowLayout valign="middle">
          <afh:cellFormat>
            <af:outputText styleClass="TEXT"
                           value="#{row.Firstname} #{row.Lastname}"
                             inlineStyle="#{(row.Lastname eq bindings.
                                             Lastname.inputValue) and
                                           (row.Firstname eq bindings.
                                              Firstname.inputValue)
                                           ?'color:orange;font-weight:
                                                          bold;':''}"/>
          </afh:cellFormat>
        </afh:rowLayout>
      </afh:tableLayout>
    </af:column>
  …
  </af:table>
</h:panelGrid>
```

Generieke lay-out

De lay-out van de applicatie zag eruit als in afbeelding 5. Een aantal elementen in deze lay-out kwamen op elke pagina terug. De logo's en het menu zijn vaste onderdelen van alle pagina's in de applicatie. Tijdens het bouwen van de applicatie is er gezocht naar een manier om deze generieke onderdelen eenmalig te definiëren en vervolgens in alle pagina's toe te passen. De oplossing is gevonden in het toepassen van regions. Een region is een generiek bouwblok van een pagina, dat kan worden toegepast op verschillende pagina's. Een region kan echter alleen worden toegepast op een klein onderdeel van de pagina. Een JSF-component is de PanelPage, een basiselement dat er voor zorgt dat alle onderdelen van een pagina kunnen worden geplaatst. Denk hierbij aan een logo, een menu-tabbar en een pagina-footer. Alle onderdelen die we kennen van een webpagina in de standaard Oracle-portal lay-out kunnen worden vastgelegd met behulp van de PanelPage. Elk kleine onderdeel van deze pagina is een Facet. In afbeelding 6 is een gedeelte van de lijst van facets weergegeven voor een PanelPage.

Voor elk facet kan een stuk code worden gedefinieerd dat generiek is opgebouwd met behulp van regions. Hierbij wordt eerst een pagina gemaakt met alleen het specifieke onderdeel

dat we willen toevoegen, bijvoorbeeld het menu. Vervolgens dient in de META-INF directory in bestand `region-metadata.xml` de definitie te worden opgenomen van de region:

```
<component>
<component-type>view.region.WKPoolMenu</component-type>
  <component-class>oracle.adf.view.faces.component.UIXRegion
  </component-class>
  <component-extension>
    <region-jsp-ui-def>/Regions/MenuRegion.jspx</region-jsp-ui-def>
  </component-extension>
  <attribute>
    <attribute-name>SelectedTab</attribute-name>
    <attribute-class>java.lang.String</attribute-class>
    <attribute-extension>
      <required>false</required>
    </attribute-extension>
  </attribute>
</component>
```

In deze code staat bij component-type de naam vermeld van de region. Achter region-jsp-ui-def staat de pagina die de region definieert. Vervolgens is het mogelijk om attributen door te geven aan de region. In bovenstaand voorbeeld is dat SelectedTab, zodat een onderdeel van het menu actief kan worden gemaakt.

```
<af:panelPage title="">
…
  <f:facet name="brandingApp">
    <af:region id="branding" regionType="view.region.Logo"/>
  </f:facet>
  <f:facet name="menu1">
    <af:region id="menu1" regionType="view.region.WKPoolMenu">
      <af:attribute name="SelectedTab" value="raadplegen_poolstand"/>
    </af:region>
  </f:facet>
  <f:facet name="menuSwitch">
    <af:region id="sponsor" regionType="view.region.Sponsor"/>
  </f:facet>
</af:panelPage>
```

Binnen de facets in een panelPage van een webpagina kunnen de regions worden opgenomen (zie code hierboven, een gedeelte van het bestand RaadplegenPoolstand.jsp). De region wordt aangeroepen met zijn naam (regionType) en tevens kunnen de waarden voor de attributen worden doorgegeven. Na enig zoeken op het internet, kregen we uiteindelijk de oplossing, zijnde regions, van een Oracle Consultant die ons begeleidde tijdens de bouw van de voetbalpool.

JSF versus UIX

Voordat Oracle aansloot bij de JSF-standaard was er UIX waarmee webpagina's konden worden gedefinieerd. De bouwers van UIX hebben meegeholpen met het opzetten van de JSF-standaard. JSF lijkt dan ook veel op UIX. De GUI-componenten kennen ze allebei, JSF heeft echter een eenvoudiger manier van het omgaan met variabelen. Eenvoudig is het om variabelen pagina-breed of process-breed mee te nemen. Bij UIX was daar geen standaardoplossing voor aanwezig. Een zeer handige manier voor een generieke lay-out, was in UIX de template. In JSF is het alleen mogelijk om een template (een region) te gebruiken voor een facet. In UIX was het mogelijk om dit te gebruiken voor een hele pagina. In de template kon dan ook worden aangegeven waar eventuele content moest worden toegevoegd. De grootste stap van UIX naar JSF is het feit dat JSF een geregistreerde standaard is en UIX een Oracle Propriety-product. Voor het migreren van UIX naar JSF zal op het gebied van templates de nodige inspanning moeten worden geleverd.

Conclusie

De meeste tijd in het bouwen van de voetbalpool is besteed aan het vinden en laten werken van Authenticatie, doorgeven van parameters en generieke lay-out. Met de aangegeven oplossingen kon er snel en eenvoudig een goede lay-out worden opgezet met behulp van JSF. Er was slechts een minimale hoeveelheid Java voor nodig om deze pagina's te bouwen. De sterke kant van ADF Faces is de productiviteitswinst die ten opzichte van vele andere Java- technologieën is te halen, Oracle zou hier meer reclame voor kunnen maken. Voor de bouw van de voetbalpool kwam het te laat, maar de JSF-versie van JHeadstart zorgt er voor dat de productiviteit steeds dichter komt bij die van Designer/Developer.

Referenties

- Declarative J2EE authentication and authorization with JAAS, An Oracle JDeveloper How To Document, Frank Nimphius and Duncan Mills, Oracle Corporation, juli 2005.
- Real-world ADF Faces, Using a custom login module with JDev 10.1.3, John Stegeman, februari 2006.

De auteurs zijn werkzaam bij Capgemini:
Ruben Spekle (ruben.spekle@capgemini.com),
Ronald Roest (ronald.roest@capgemini.com),
Jan Maarten van der Valk
(janmaarten.vander.valk@capgemini.com),
Marc Lameriks (marc.lameriks@capgemini.com).