Project met JDeveloper en ADF

Data binding in ADF JClient

Bij het bedrijf TenneT werd ter ondersteuning van het plannen van de elektriciteitsbalans een Oracle webforms-applicatie gebruikt, waarnaast nog allerlei handmatige acties moesten worden uitgevoerd. Er ontstond behoefte aan een applicatie met een meer logische indeling van de user-interface. Daarbij zouden de medewerkers, door het automatiseren van handelingen, de toestand van het proces op elk moment moeten kunnen checken. In een tweetal artikelen, waarvan dit het eerste is, zal nader worden ingegaan op de ontwikkeling van de nieuwe Libra In Balans Dashboard-applicatie: een J2EE rich client-applicatie. In dit eerste artikel zal de nadruk liggen op het data binding mechanisme.

Als enige organisatie in Nederland is TenneT verantwoordelijk voor het beheer van het landelijk hoogspanningsnet en tevens voor het bewaken van de betrouwbaarheid en continuïteit van de Nederlandse elektriciteitsvoorziening. TenneT is verantwoordelijk voor een veilig, betrouwbaar en doelmatig systeem. Kort gezegd komt dit neer op het handhaven en beschermen van de balans tussen vraag en aanbod en het mogelijk maken van gevraagde transporten.
In de voorbereidingsfase (dag Dplus1) worden vraag en aanbod zo nauwkeurig mogelijk ingeschat. Voor deze inschatting is er het systeem van programmaverantwoordelijkheid. Elke dag ontvangt TenneT de Energieprogramma's van de programmaverantwoordelijken waarin staat hoeveel elektriciteit zij de volgende dag per kwartier verwachten te leveren of te ontvangen. Dit gebeurt via het Centraal Postbus Systeem (CPS) waarmee de programmaverantwoordelijken hun E-programma-berichten digitaal kunnen aanleveren.

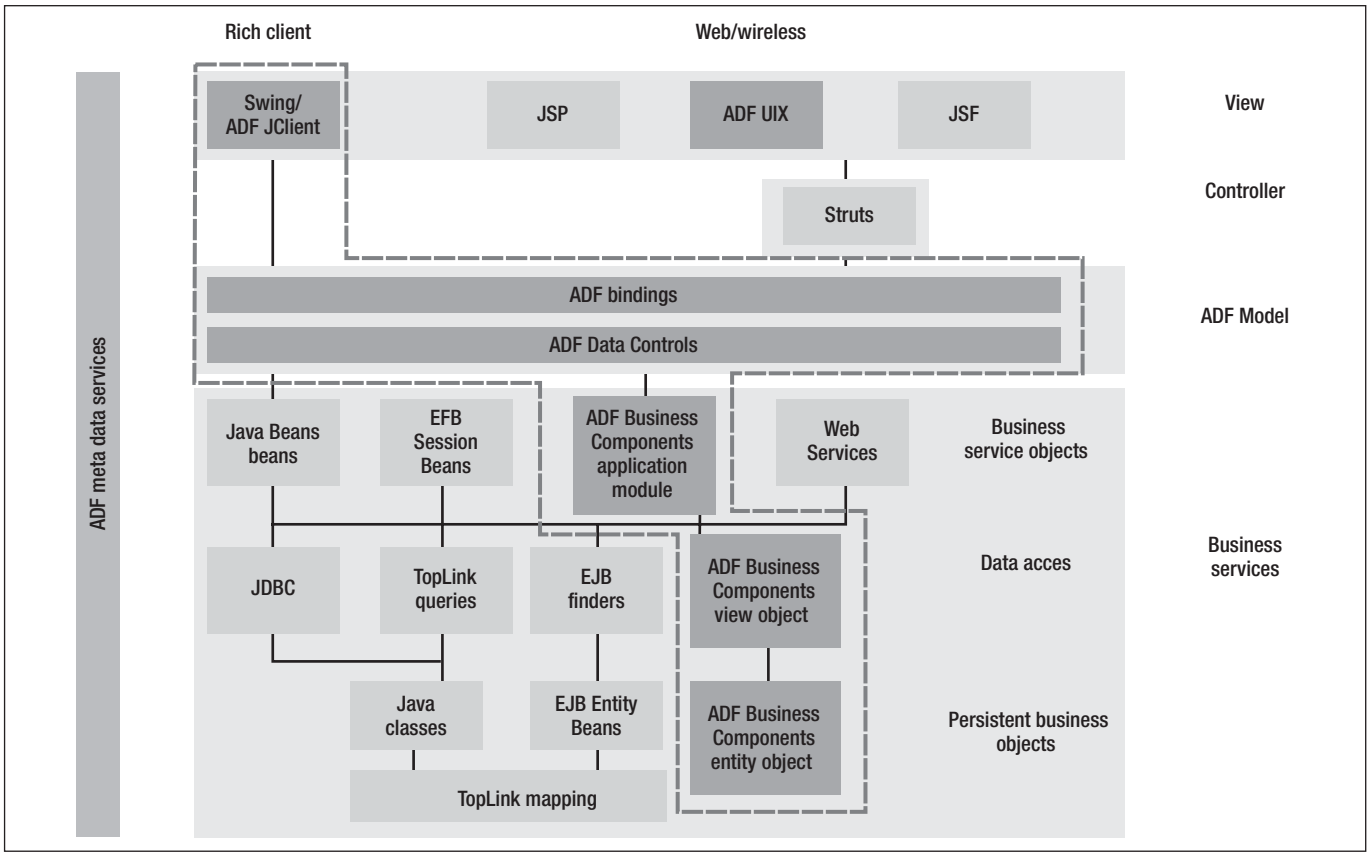
Toekomstvisie

Op de dag van uitvoering van de Energieprogramma's (dag D) controleert TenneT continu de mate van onbalans en stuurt deze bij. Voor het handhaven en beschermen van de balans werd als ondersteuning een Oracle webforms-applicatie gebruikt (Libra) en werden allerlei handmatige acties eromheen uitgevoerd. Voorbeelden hiervan waren het handmatig starten van een deelproces vanuit een webform, alsmede het verzamelen van informatie volgens het 'knippen en plakken'-principe.
Om te zorgen dat de medewerkers van de afdeling Operationele Besturing het voorbereidingsproces zo efficiënt mogelijk kunnen realiseren, is besloten de ondersteuning van dat proces te optimaliseren door te automatiseren. Hierbij speelde ook mee dat de medewerkers via verschillende schermen de benodigde informatie moesten betrekken, terwijl ze de informatie over de toestand van de processen in één oogopslag wilden zien. Er ontstond daarom behoefte aan een applicatie met een meer logische indeling van de user-interface waarbij, door de handelingen te automatiseren, de medewerkers de toestand van het proces op elk moment kunnen checken. In lijn met de toekomstvisie van de afdeling Informatie en Automatisering is de nieuwe Libra In Balans Dashboard-applicatie ontwikkeld als J2EE-applicatie.

Architectuur

De Libra In Balans Dashboard-applicatie is gebouwd met behulp van het Oracle Application Development Framework. J2EE biedt een conceptueel architectuurmodel dat is opgedeeld in een aantal lagen, waarbij elke laag verantwoordelijk is voor specifieke functionaliteit. Het architectuurmodel bevat een viertal lagen:

1. Client-laag (*Client-Side Presentation Tier*).
Voor de Client-laag is gebruik gemaakt van ADF JClient met Swing-componenten.
2. Web-laag *(Server-Side Presentation Tier*).
Er is geen gebruik gemaakt van een Web laag.
3. Business-laag (*Server Business Logic Tier*).
Voor de Business laag is gebruik gemaakt van één ADF Business Components application module en diverse ADF Business Components view-objecten en ADF Business Components entity-objecten.
4. EIS-laag (*Enterprise Information System Tier*).
Als EIS-laag is gebruik gemaakt van een Oracle 9i database.



Afbeelding 1. Oracle ADF-technologieën.

In afbeelding 1 zijn de gekozen technologieën omkaderd weergegeven.

Opbouw dashboard

De Libra In Balans Dashboard-applicatie is een venster (JFrame) samengesteld uit een aantal panelen (JPanel). Elk paneel is hierbij via een data binding (JUPanelBinding) gekoppeld aan een view-object. Hoe dit binding-mechanisme werkt zal hieronder nader worden toegelicht aan de hand van de gegevens, die zichtbaar zijn op het tabblad Eprogramma's in afbeelding 2. Dit tabblad is opgebouwd uit een aantal panelen die in afbeelding 3 zijn benoemd.

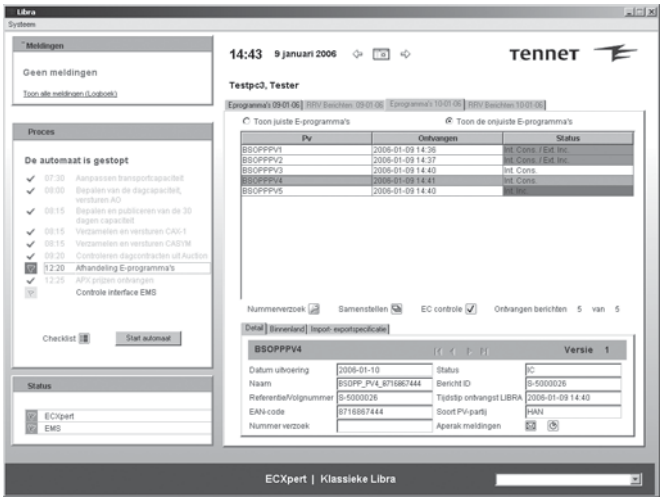
Data binding in ADF JClient

Met data binding in een ADF JClient-applicatie is het mogelijk om Swing-containers (bijvoorbeeld een JPanel) en componenten (bijvoorbeeld een JTable of een JTextField) te creëren, welke verbonden zijn aan data in de ADF Business serviceslaag (zie afbeelding 1). Om met data binding in een ADF JClient-applicatie te kunnen werken, dient iedere swing-container (JFrame of JPanel) een binding container object (ook wel panel-binding genoemd) te creëren. Alle binding containers van een applicatie worden geregistreerd in de binding context. Een binding container object kan bestaan uit action bindings, value bindings en iterator bindings. Deze zullen hierna kort besproken worden.

Binding context object (BindingContext.class)

Het bestand `DataBindings.cpx` definieert de Oracle ADF binding context voor de gehele applicatie. Het `.cpx` bestand (zie voorbeeld hieronder) levert de metadata van waaruit de Oracle ADF binding objecten worden gecreëerd op runtime. De binding context levert toegang tot alle binnen de applicatie gebruikte data bindings.

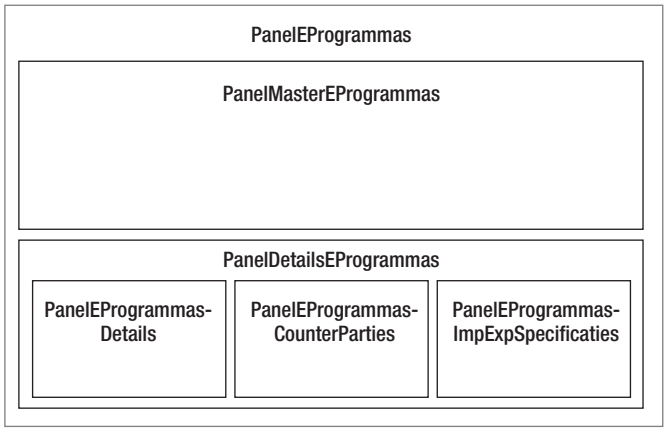
```
<?xml version='1.0' encoding='windows-1252' ?>
<JboProject
   id="DataBindings"
   xmlns="http://xmlns.oracle.com/adfm/application"
   version="10.1.2.17.96"
   SeparateXMLFiles="false"
   Package=""
   ClientType="JClient" >
   <Contents >
      <DataControl
         id="LibraInBalansServiceDataControl"
         SubType="DCBC4J"
         SupportsFindMode="true"
         SupportsTransactions="true"
         Package="org.tennet.lib.model.service"
```



Afbeelding 2. Libra In Balans Dashboard applicatie.

```
            FactoryClass="oracle.adf.model.bc4j.
            DataControlFactoryImpl"
            Configuration="LibraInBalansServiceLocal" >
        </DataControl>
        <Containee
            id="PanelMasterEProgrammasUIModel"
            ObjectType="BindingContainerReference"
            FullName="org.tennet.lib.view.components.
            tablespanel.eprogrammas.
            PanelMasterEProgrammasUIModel" >
        </Containee>
        <Containee
            id="PanelEProgrammasDetailsUIModel"
            ObjectType="BindingContainerReference"
            FullName="org.tennet.lib.view.components.
            tablespanel.eprogrammas.
            PanelEProgrammasDetailsUIModel" >
        </Containee>
        <Containee
            id="PanelEProgrammasCounterPartiesUIModel"
            ObjectType="BindingContainerReference"
            FullName="org.tennet.lib.view.components.
            tablespanel.eprogrammas.
            PanelEProgrammasCounterPartiesUIModel" >
        </Containee>
        …
        <Containee
            id="FormUIModel"
            ObjectType="BindingContainerReference"
            FullName="org.tennet.lib.view.FormUIModel" >
        </Containee>
    </Contents>
</JboProject>
```

Om data-binding mogelijk te maken, levert JClient een API welke samenwerkt met de Oracle ADF-modellaag (zie afbeelding 1). In de Libra In Balans Dashboard-applicatie wordt in het hoofdscherm (Form.class) door middel van de API, de binding context gecreëerd met gebruikmaking van het bestand `DataBindings.cpx`. In onderstaande code wordt dit geïllustreerd.



Afbeelding 3. Overzicht panelen bij tabblad Eprogramma's.

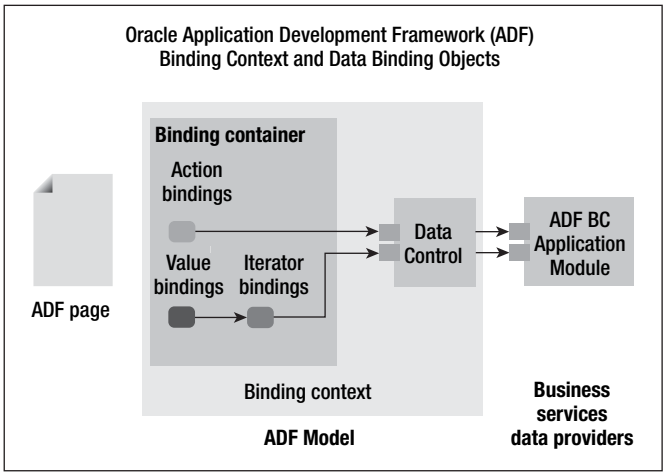
Het is gebruikelijk om het scherm (JFrame) dat de eerste panel-binding creëert ook de binding context te laten creëren.

```
try
{
  // bootstrap application
  JUMetaObjectManager.setErrorHandler(new
  JUErrorHandlerDlg());
  JUMetaObjectManager mgr = JUMetaObjectManager.
  getJUMom();
  mgr.setJClientDefFactory(null);
  BindingContext ctx = new BindingContext();
  ctx.put(DataControlFactory.APP_PARAM_ENV_INFO, new
  JUEnvInfoProvider());
  ctx.setLocaleContext(new DefLocaleContext(null));
  HashMap map = new HashMap(4);
  map.put(DataControlFactory.APP_PARAMS_BINDING_CONTEXT,
  ctx);
  mgr.loadCpx("DataBindings.cpx", map);
  Form frame = new Form();
  frame.setBindingContext(ctx);
}
catch(Exception ex)
{
  JUErrorHandlerDlg dlg = new JUErrorHandlerDlg();
  dlg.reportException(null, ex, true);
  System.exit(1);
}
```

Hiermee is vanuit één centrale plek de binding-context bekend. Steeds wanneer een paneel wordt gecreëerd, wordt door middel van het aanroepen van de setBindingContext-method, de binding-context doorgegeven.

Binding container objects (DCBindingContainer.class)

Iedere swing-container (JFrame of JPanel) dient een binding container object te creëren. Deze binding container wordt ook wel panel binding genoemd, waarbij JUPanelBinding een subclass is van DCBindingContainer.

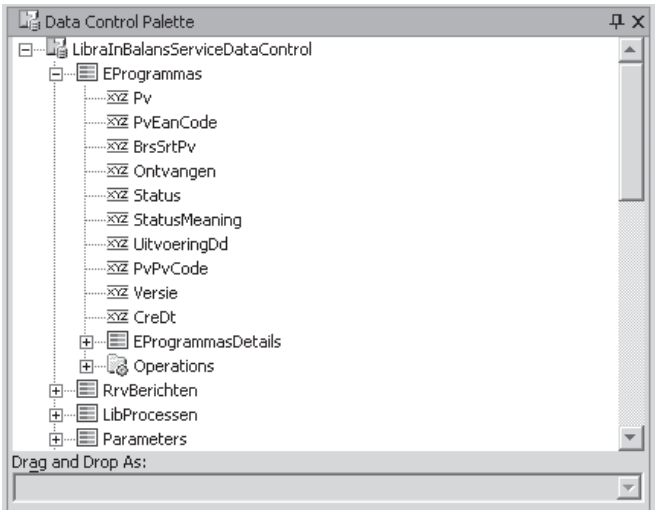


Afbeelding 4. Overzicht binding context.

```
public class PanelMasterEProgrammas extends JPanel
implements JUPanel
{
  private JUPanelBinding panelBinding =
  new JUPanelBinding("PanelMasterEProgrammasUIModel");
  …
}
```

Wanneer in JDeveloper aan een swing container een data-bound UI-component wordt toegevoegd met behulp van de Data Control Palette, zal JDeveloper metadata toevoegen aan het bestand `PanelNameUIModel.xml`. Door middel van dit bestand wordt de binding container geïnitialiseerd. Indien het bestand nog niet aanwezig is, wordt dit automatisch aangemaakt. Zo bevat de Libra In Balans Dashboard-applicatie onder andere de bestanden `PanelMasterEProgrammasUIModel.xml` (zie voorbeeld hieronder) en `PanelEProgrammasDetailsUIModel.xml`. In de Libra In Balans Dashboard-applicatie is aan het paneel PanelMasterEProgrammas het view-object EProgrammas gekoppeld via een JTable-component.

```
<?xml version='1.0' encoding='windows-1252' ?>
<DCContainer
    id="PanelMasterEProgrammasUIModel"
    xmlns="http://xmlns.oracle.com/adfm/uimodel"
    version="10.1.2.17.96"
    Package="org.tennet.lib.view.components.tablespanel.
    eprogrammas"
    FindMode="false"
    EnableTokenValidation="true" >
    <Contents >
       <DCIterator
          id="EProgrammasIterator"
          Binds="LibraInBalansServiceDataControl.
          EProgrammas" >
       </DCIterator>
       <DCControl
          id="EProgrammas"
          DefClass="oracle.jbo.uicli.jui.JUTableDef"
          SubType="DCTable"
          BindingClass="oracle.jbo.uicli.jui.
          JUTableBinding"
          IterBinding="EProgrammasIterator"
          ApplyValidation="false"
          isDynamic="false" >
          <AttrNames>
             <Item Value="Pv" />
             <Item Value="Ontvangen" />
             <Item Value="StatusMeaning" />
          </AttrNames>
       </DCControl>
    </Contents>
</DCContainer>
```



Afbeelding 5. Data Control Palette.

In de Libra In Balans Dashboard-applicatie zijn aan paneel PanelEProgrammasDetails een aantal attributen van het view-object EProgrammasDetails gekoppeld via JTextField-componenten:

```
<?xml version='1.0' encoding='windows-1252' ?>
<DCContainer
    id="PanelEProgrammasDetailsUIModel"
    xmlns="http://xmlns.oracle.com/adfm/uimodel"
    version="10.1.2.17.96"
    Package="org.tennet.lib.view.components.tablespanel.
    eprogrammas"
    FindMode="false"
    EnableTokenValidation="true" >
    <Contents >
       <DCIterator
          id="EProgrammasDetailsIterator"
          Binds="LibraInBalansServiceDataControl.
          EProgrammasDetails" >
       </DCIterator>
       <DCControl
          id="UitvoeringDd"
          DefClass="oracle.jbo.uicli.jui.JUTextFieldDef"
          SubType="DCTextField"
```

```
         IterBinding="EProgrammasDetailsIterator"
         ApplyValidation="false"
         isDynamic="false" >
         <AttrNames>
            <Item Value="UitvoeringDd" />
         </AttrNames>
      </DCControl>
      <DCControl
         id="PvNaam"
         DefClass="oracle.jbo.uicli.jui.JUTextFieldDef"
         SubType="DCTextField"
         IterBinding="EProgrammasDetailsIterator"
         ApplyValidation="false"
         isDynamic="false" >
         <AttrNames>
            <Item Value="PvNaam" />
         </AttrNames>
      </DCControl>
      <DCControl
         id="RefNr"
         DefClass="oracle.jbo.uicli.jui.JUTextFieldDef"
         SubType="DCTextField"
         IterBinding="EProgrammasDetailsIterator"
         ApplyValidation="false"
         isDynamic="false" >
         <AttrNames>
            <Item Value="RefNr" />
         </AttrNames>
      </DCControl>
      …
   </Contents>
</DCContainer>
```

Iterator binding objects (DCIteatorBinding.class)

Een iterator binding object handelt de events af welke gegenereerd worden vanuit de row iterator behorend bij het view object (Business services laag) en verstuurt de current row naar de individuele control binding objecten zodat deze de data behorend bij de current row kunnen tonen. De control binding objecten zijn value binding en action binding objecten (zie afbeelding 4).

Value binding objects (JUCtrlValueBinding.class)

De value binding staat een databound UI-component toe om de waarde van een view object attribuut te verkrijgen en afhankelijk van het type UI-component, kan de gebruiker de waarde alleen zien of ook wijzigen. In de Libra In Balans Dashboard applicatie is aan paneel PanelMasterEProgrammas het view-object EProgrammas gekoppeld via een JTable-component (tableEProgrammas) met gebruikmaking van de method setModel (zie afbeelding 6).

```
public class PanelMasterEProgrammas extends JPanel
implements JUPanel
{
  …
  public void jbInit() throws Exception
  {
    …
    tableEProgrammas.setModel((TableModel)panelBinding.
    bindUIControl("EProgrammas", tableEProgrammas));
    …
  }
  …
}
```

In de Libra In Balans Dashboard-applicatie is een aantal attributen van het view-object EProgrammasDetails gekoppeld aan paneel PanelEProgrammasDetails via JTextField componenten (mUitvoeringDd, mPvNaam) met gebruikmaking van de method setDocument (zie afbeelding 7).

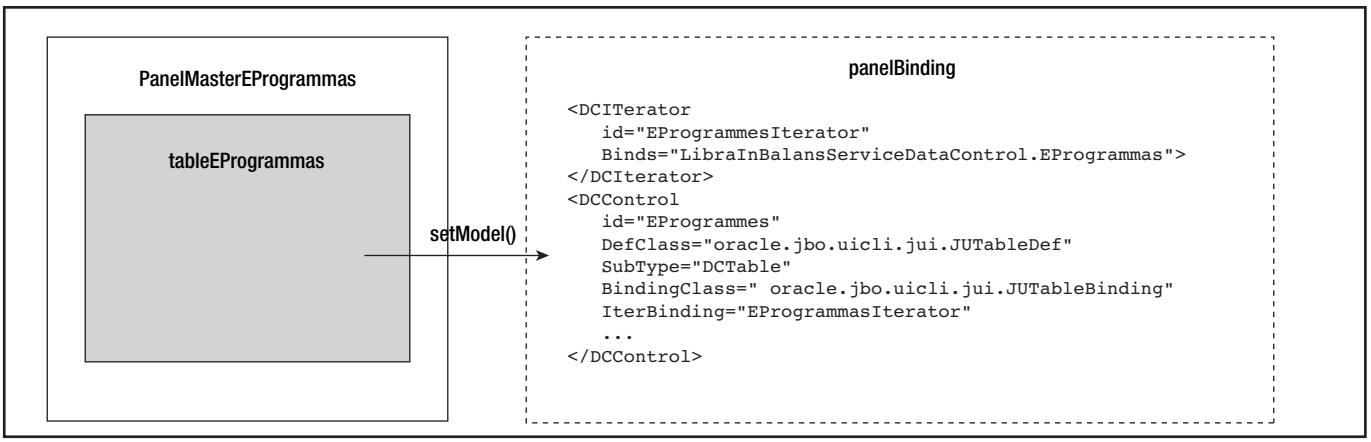
```
public class PanelEProgrammasDetails extends JPanel
implements JUPanel
{
  …
  public void jbInit() throws Exception
  {
    …
    mUitvoeringDd.setDocument((Document)panelBinding.
    bindUIControl("UitvoeringDd", mUitvoeringDd));
    mPvNaam.setDocument((Document)panelBinding.
    bindUIControl
   ("PvNaam", mPvNaam));
    …
  }
  …
}
```

Action binding objects (JUCtrlActionBinding)

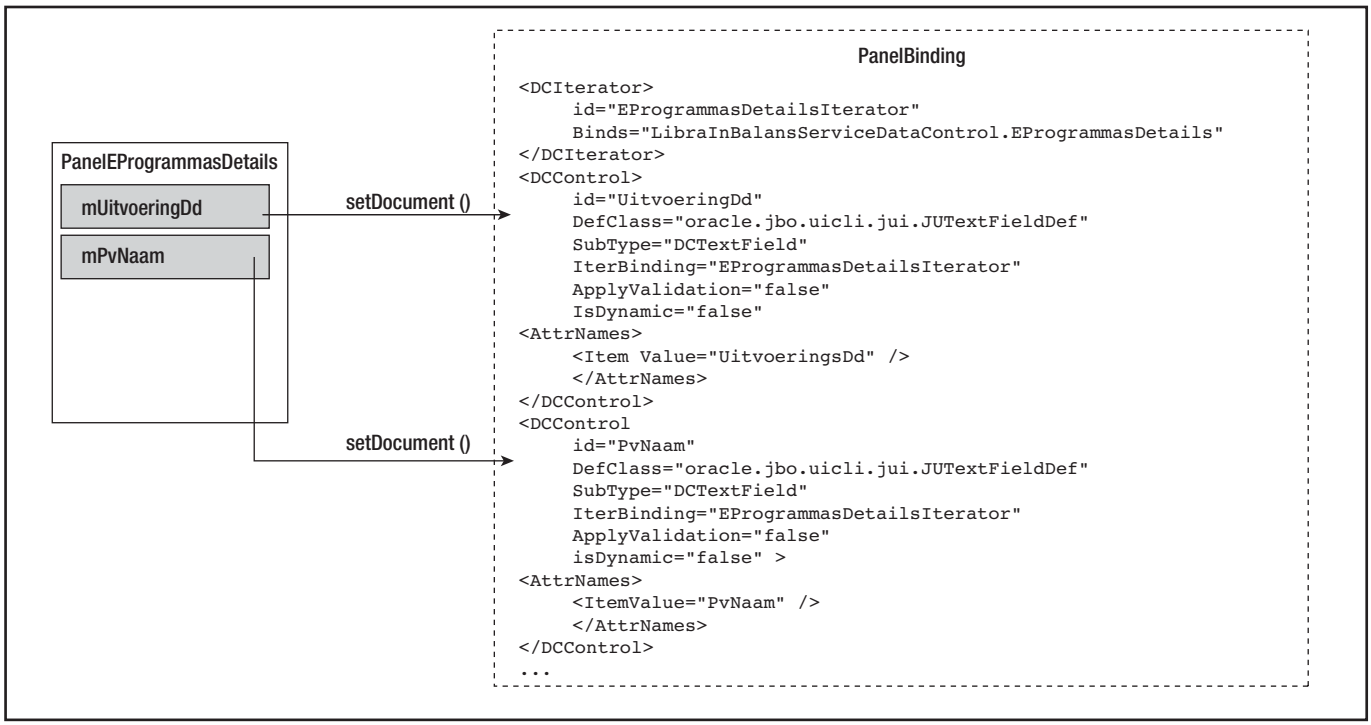
Een action binding voert acties uit op de row iterator behorend bij het view object (Business services laag). Zo kan bijvoorbeeld een JButton component gekoppeld worden aan één van de operations behorend bij een view object (zoals First of Next).

Conclusie

Met data binding in een ADF JClient-applicatie is het mogelijk om Swing-containers (bijvoorbeeld een JPanel) en componenten (bijvoorbeeld een JTable of een JTextField) te creëren, welke verbonden zijn aan data. Iedere swing container is hierbij gekoppeld aan een binding container object (ook wel panel binding genoemd), waarin weer action-, value- en iterator bindings kunnen worden opgenomen. Alle binding containers van een applicatie worden geregistreerd in de binding context. In het eerste artikel uit deze serie is aan de hand van voorbeelden nader ingegaan op het data binding mechanisme. In het tweede deel zal onder meer nader worden ingegaan op het gebruik van meerdere instanties van een JClient-paneel, alsmede het



Afbeelding 6. Panel binding bij paneel PanelMasterEProgrammas.



Afbeelding 7. Panel binding bij paneel PanelEProgrammasDetails.

gebruik van zogenaamde cellRenderers en cellEditors om de representatie van een cell in een JTable te manipuleren.

Referenties

- Oracle Application Development Framework Overview, An Oracle Technical White Paper, August 2005.
- Oracle Application Development Framework, Development Guidelines, Oracle JDeveloper 10g (9.0.5.2), August 2004

Marc Lameriks is werkzaam als Senior Consultant bij Capgemini.