Project met JDeveloper en ADF

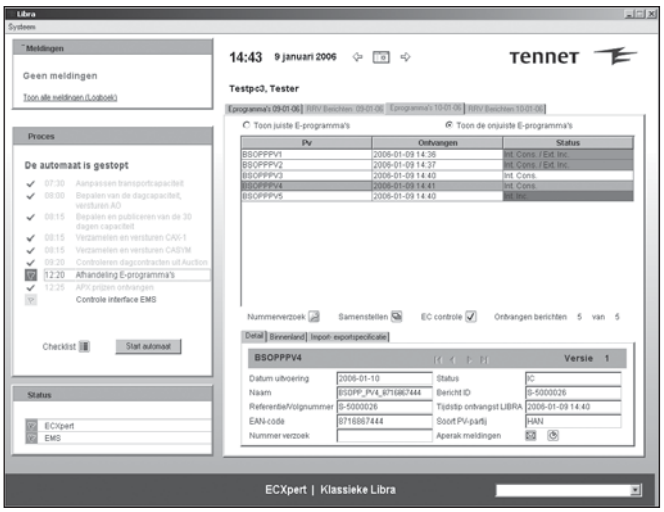
Scherm lay-out en ADF JClient

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 serie van twee artikelen wordt nader ingegaan op de ontwikkeling van de nieuwe Libra In Balans Dashboard-applicatie: een J2EE rich client-applicatie. In dit tweede artikel zal de nadruk liggen op de scherm lay-out.

In lijn met de toekomstvisie van de afdeling Informatie en Automatisering is de nieuwe Libra In Balans Dashboard-applicatie als zogenaamde J2EE-applicatie ontwikkeld. In het eerste artikel lag de nadruk op het data binding mechanisme. In dit artikel zal nader worden ingegaan op het gebruik van meerdere instanties van een JClient-paneel, alsmede het gebruik van zogenaamde cellRenderers en cellEditors om de representatie van een cell in een JTable te manipuleren.

Gebruik van meerdere instanties

In afbeelding 1 is te zien dat er een tweetal tabbladen zijn waarop de zogenaamde Energieprogramma's te zien zijn. Eenmaal voor de voorbereidingsfase (dag Dplus1, Eprogramma's 10-01-06) en eenmaal voor de uitvoeringsdag zelf (dag D, Eprogramma's 09-01-06). Beide tabbladen tonen dezelfde gegevens, alleen de dag die het betreft, is verschillend. Een voor de hand liggende oplossing hiervoor (mede gelet op het onderhoud), is het op runtime aanmaken van een tweetal instanties van het betreffende paneel (inclusief panel binding) en het gebruik van *whereclauseparameters* bij het onderliggende view object om bijvoorbeeld de datum van de uitvoeringsdag in te stellen.



Afbeelding 1. Libra In Balans Dashboard-applicatie.

```
private PanelMasterEProgrammas panelMasterEProgrammasD =
        new PanelMasterEProgrammas("D");

private PanelMasterEProgrammas panelMasterEProgrammasDplus1 =
        new PanelMasterEProgrammas("Dplus1");
```

Volgens bovenstaande code zijn de paneel instanties panelMasterEProgrammasD en panelMasterEProgrammasDplus1 aangemaakt, gebaseerd op paneel definitie panelMasterEProgrammas (zie afbeelding 2).

In onderstaande code wordt de datum van de uitvoeringsdag opgehaald (method getDayD of getDayDplus1) en met gebruikmaking van de method queryEProgrammas in de applicatiemodule (libraInBalansServiceImpl), wordt de datum (09-jan-2006 dan wel 10-jan-2006) doorgegeven aan het view object. Hierbij is variabele dayIdentifier gelijk aan "D" of "Dplus1".

```
if (this.dayIdentifier.equalsIgnoreCase("D"))
{
  libraInBalansService.queryEProgrammas(settings.getDayD());
}
else
{
  libraInBalansService.queryEProgrammas(settings.getDayDplus1());
}
```

Method queryEProgrammas in de applicatiemodule (libraInBalansServiceImpl) roept method queryEProgrammas in het view object (EProgrammasImpl) aan, zoals in onderstaande code is te zien.

```
public class LibraInBalansServiceImpl extends ApplicationModuleImpl
    implements org.tennet.lib.model.service.common.LibraInBalansService
{
  ...

  /**
    *
    *  Container's getter for EProgrammas
    */
  public EProgrammasImpl getEProgrammas()
  {
    return (EProgrammasImpl)findViewObject("EProgrammas");
  }

  ...

  /**
    * Query correct EProgrammas, and set runtime where clause.
    *
    */
  public void queryEProgrammas(Date uitvoeringDd)
  {
    this.getEprogrammas().queryEProgrammas(uitvoeringDd);
  }
  ...
}
```

Method queryEProgrammas in het view object (EProgrammasImpl) stelt de *whereclauseparameters* in (setWhereClauseParam-method) en voert de query uit (executeQuery-method).

```
public class EProgrammasImpl extends ViewObjectImpl
{
  ...

  /**
    * Query EProgrammas, and set where clause parameters.
    *
    */
  public void queryEProgrammas(Date uitvoeringDd)
  {
    try
    {
      SimpleDateFormat simpleDateFormatter =
      new SimpleDateFormat("yyyy-MM-dd");
      this.setWhereClauseParam(0, simpleDateFormatter.
      format(uitvoeringDd));
      ...
      this.executeQuery();
    }
    catch (Exception e)
    {
    ...
    }
  }
  ...
}
```

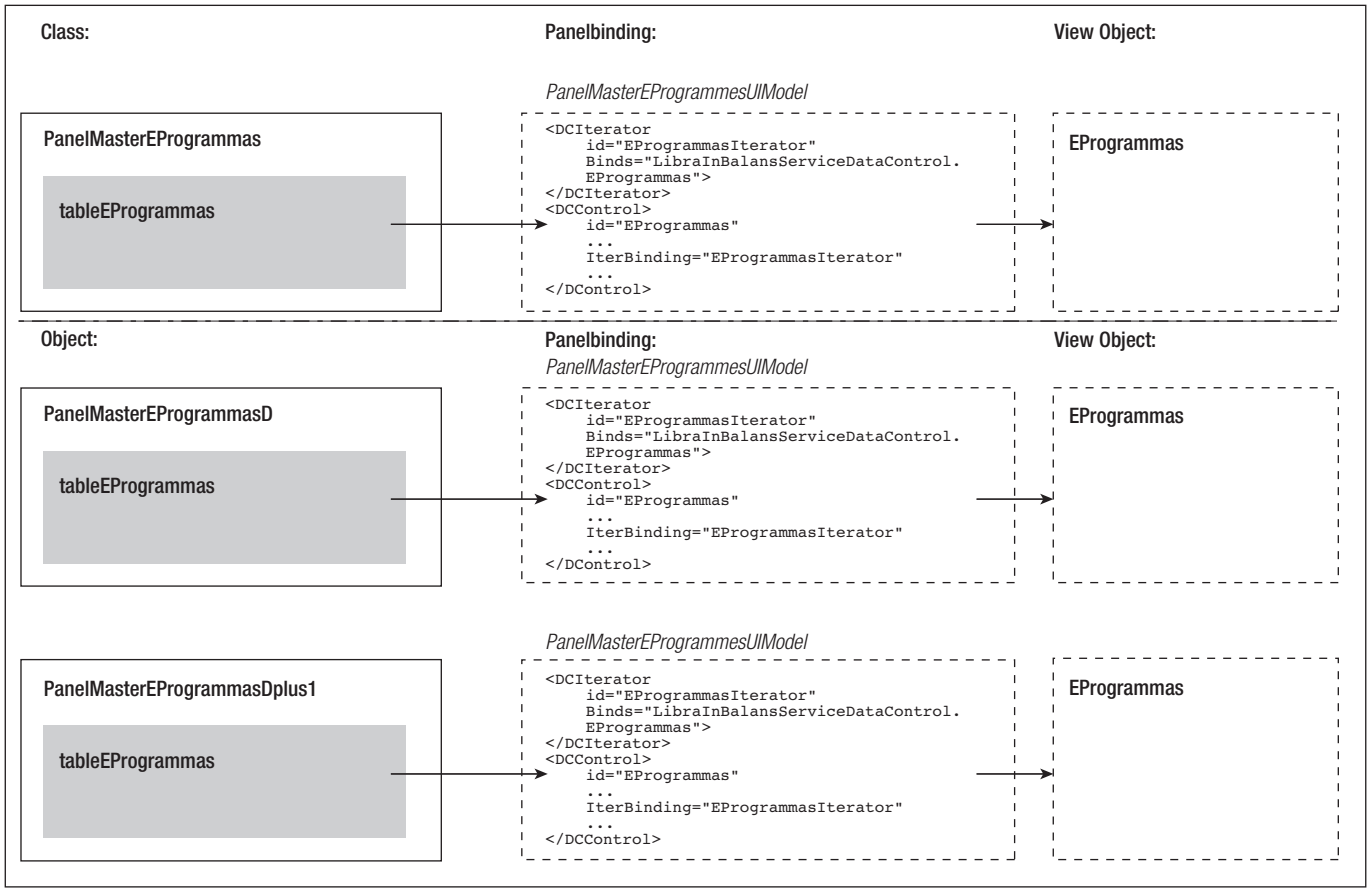
Omdat in de Libra In Balans Dashboard-applicatie sommige panelen tegelijkertijd zichtbaar kunnen zijn, is deze constructie echter niet zonder meer bruikbaar. Immers bij gebruik van hetzelfde view object in verschillende panelen wordt ook dezelfde inhoud van het view object getoond. De oplossing hiervoor is om op basis van het view object (metadata) een view object instantie aan te maken (in onderstaande code voEProgrammasCopy genoemd), met gebruikmaking van de createViewObject-method die standaard in de applicatiemodule (ApplicationModuleImpl) aanwezig is. Hetzelfde principe gaat ook op voor de view links (createViewLink-method).

```
EProgrammasImpl voEProgrammasCopy = (EProgrammasImpl) libraInBalansSer-
vice.createViewObject(
"EProgrammas" + this.dayIdentifier, "org.tennet.lib.model.dataaccess.
EProgrammas");
```

Op deze wijze zijn de view object instanties EProgrammasD en EProgrammasDplus1 aangemaakt, gebaseerd op de metadata van view object EProgrammas. Per view object instantie zijn daarna de *whereclauseparameters* ingesteld. Hiertoe is method queryEProgrammas in de applicatiemodule (libraInBalansServiceImpl) gewijzigd. Deze maakt gebruik van method findViewObject in plaats van method getEProgrammas.

```
public class LibraInBalansServiceImpl extends ApplicationModuleImpl
    implements org.tennet.lib.model.service.common.LibraInBalansService
{
  ...

  /**
    * Query correct EProgrammas, and set runtime where clause.
    *
    */
  public void queryEProgrammas(Date uitvoeringDd, String dayIdentifier)
  {
    ...
```



Afbeelding 2. Paneel-instanties.

```
    EProgrammasImpl voEProgrammasCopy = (EProgrammasImpl) findView
    Object(
    "EProgrammas" + dayIdentifier);

    voEProgrammasCopy.queryEProgrammas(uitvoeringDd);
  }
  ...
}
```

Bovenstaande oplossing had vanwege het onderhoud de voorkeur boven het op design time aanmaken van de view objecten EProgrammasD en EProgrammasDPlus1 (alsmede de bijbehorende view links). Dit zou dan voor alle view objecten/view links in de Libra In Balans Dashboard-applicatie dienen te gebeuren. Het op runtime aanmaken van instanties van panelen ging niet zonder slag of stoot. Hierbij ontstond onderstaande foutmelding:

```
oracle.jbo.JboException:35001 being raised when
adding multiple instance of an existing JClient
panel to another JClient panel or when launching
multiple instances of a panel from the same bin-
ding context.
```

Deze melding komt er op neer, dat binnen dezelfde binding context geen binding containers (panel binding) met dezelfde naam gedefinieerd mogen worden. Dit is opgelost door op runtime binding container instanties aan te maken met een unieke naam, waarbij de instanties zijn gebaseerd op een bepaalde binding container metadata (`PanelNameUIModel.xml`). Door middel van onderstaande code zijn zo voor de panelen PanelMasterEProgrammasD en PanelMasterEProgrammasDplus1 de standaard bijbehorende binding containers (PanelMasterEProgrammasUIModel) vervangen door de op runtime aangemaakte binding container instanties PanelMasterEProgrammasUIModelD respectievelijk PanelMasterEProgrammasUIModelDplus1 (zie afbeelding 3, punt 1). Waarbij de op runtime aangemaakte binding container instanties in de binding context worden geregistreerd (met behulp van de setBindingContext-method). In de code wordt PanelMasterEProgrammas het blueprintPanel genoemd en de panelen PanelMasterEProgrammasD en PanelMasterEProgrammasDplus1 steeds het copyPanel.

```
// UI Model ophalen vanuit blueprint panel
JUPanelBinding blueprintPanelBinding = blueprintPanel.getPanelBinding();

// Aanmaken instances van UI Model gebaseerd op UI Model behorend bij
```

```
blueprint panel
// en het koppelen daarvan aan de panel instances
blueprintPanel.createNewInstance(blueprintPanelBinding,
blueprintUIModelName + dayIdentifier, copyPanel);

//Instellen naam van de panel instances
copyPanel.setName(blueprintPanelName + dayIdentifier);

copyPanel.setBindingContext(panelBinding.getBindingContext());
```

Hierbij is gebruik gemaakt van de in het blueprintPanel gedefinieerde method createNewInstance.

```
public void createNewInstance(JUPanelBinding pb, String secondName
, BluePrintADFPanel secondPanel)
{
  String secondPanelName = secondName;

  pb.getBindingContext().put(secondPanelName,
  new DCBindingContainerReference(pb.getBindingContext(),
  secondPanelName, this.getPanelBinding().getDef().getFullName()));
  secondPanel.getPanelBinding().setName(secondPanelName);
}
```

Tot slot is voor elke binding container instantie, de juiste view object instantie gekoppeld aan de iterator binding (middels de bindRowSetIterator-method) (zie afbeelding 3, punt 2) en is de iterator hernoemd (middels de setName-method) (zie afbeelding 3, punt 3). Zo is bijvoorbeeld voor de binding container instantie PanelMasterEProgrammasUIModelDplus1, view object instantie EprogrammasDplus1 gekoppeld aan de iterator (welke hernoemd is van EprogrammasIterator naar EprogrammasIteratorDplus1).

```
// Aanmaken instance van iterator en koppelen aan panel instance
JUIteratorBinding iterCopy = (JUIteratorBinding) copyPanel.
getPanelBinding().findIteratorBinding(blueprintIteratorBindingName);

ViewObject voCopy = getApplicationModule().findViewObject(
blueprintViewObjectName + dayIdentifier);

iterCopy.bindRowSetIterator(voCopy, true);

iterCopy.setName(iterCopy.getName() + dayIdentifier);
```

Aanpassen representatie datatype

Om de representatie in een JTable voor een willekeurige cell in een bepaalde kolom (lees datatype) te wijzigen ten opzichte van de default, kunnen zogenaamde cellRenderers en cellEditors aan een kolom van een tabel gekoppeld worden. De renderer wordt gebruikt als de cell in presentatie modus is en de editor als de cell in edit modus is. Bij bijvoorbeeld een JTextField wordt standaard de edit modus bereikt door een dubbel klik in het veld. Of een kolom editable is, is enerzijds afhankelijk van de waarde van de Updateable property van het bijbehorende attribute in het view object en anderzijds van de return waarde van de method isEditable van de cellEditor.

In de Libra In Balans Dashboard-applicatie is aan het paneel Subprocessen (afbeelding 4) het viewobject LibSubProcessen gekoppeld via een JTable-component. In de tabel is te zien welke attributen van het view object in het paneel zichtbaar zijn. De eerste kolom van de JTable toont altijd een icoontje afhankelijk van de waarde van het attribuut Status. Hiertoe is een zelf gedefinieerde cellRenderer (genaamd ProcessListIconCellRenderer) gekoppeld aan de kolom van de tabel.

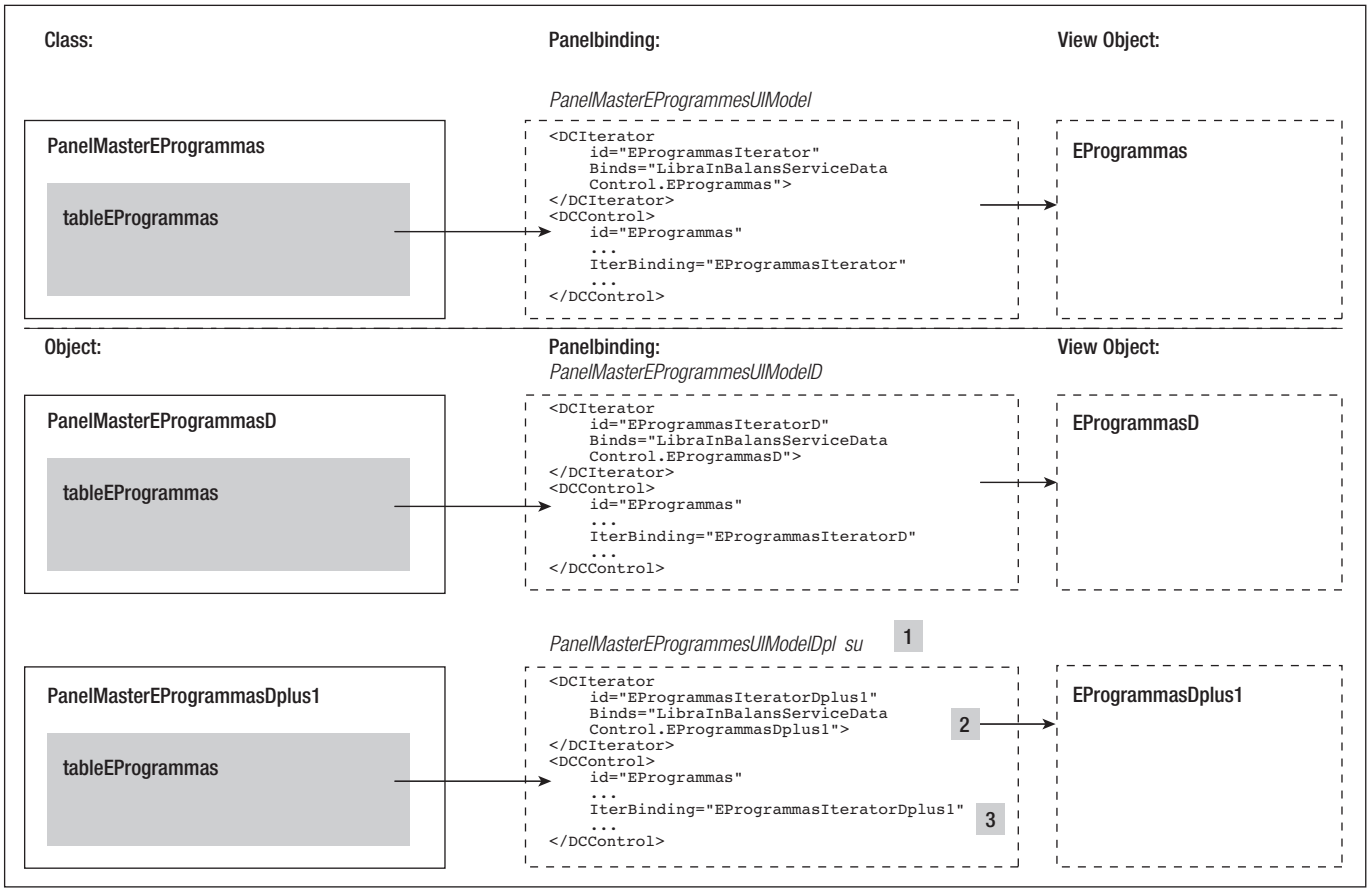
```
tableLibSubProcessen.getColumnModel().getColumn(0).setCellRenderer
(new ProcessListIconCellRenderer());
```

Een cellRenderer dient een getTableCellRendererComponent method implementatie te bevatten. De ProcessListIconCellRenderer rendert de cellen in de kolom naar een JLabel component waarmee het icoontje wordt getoond.

```
public class ProcessListIconCellRenderer extends JLabel implements
TableCellRenderer
{
  ...
  public Component getTableCellRendererComponent(JTable table, Object
  value, boolean isSelected, boolean hasFocus, int row, int column)
  {
    Color newForeground = new Color(0, 0, 0);
    Icon newIcon = null;

    // get the status value of the status column of this row
```

Attribuut Name	Attribuut Type	Updateable	Query Column	Column Type	JTable Column
Status	String	Always	STATUS	VARCHAR2(20)	0
Tijdstip	String	Never	TIJDSTIP	VARCHAR2(5)	1
Omschrijving	String	Never	OMSCHRIJVING	VARCHAR2(240)	2
HandmatigJn	Boolean	Always	-	-	3 (niet zichtbaar)
HandmatigAfgevinktJn	Boolean	Always	-	-	4



Afbeelding 3. Binding container instanties.

```
    if (value == null)
    {
      this.setText("No status");
      this.setIcon(null);
    }
    else
    {
      String status = (String) value;
      this.setVerticalAlignment(JLabel.TOP);

      if ("COMPLETED".equals(status))
      {
        newForeground = LibConst.COLOR_INACTIVE;
        newIcon = LibConst.ICON_COMPLETED;
      }
      ...
      else if ("HAS_ERROR".equals(status))
      {
        newForeground = LibConst.COLOR_ERROR;
        newIcon = LibConst.ICON_HAS_ERROR;
      }
      ....

      this.setForeground(newForeground);
      this.setIcon(newIcon);
    }
    return this;
  }
}
```

Alhoewel het attribuut Status updateable is en dient te zijn, dient het icoontje (eerste kolom) niet wijzigbaar te zijn. Hiertoe is een zelfgedefinieerde cellEditor (genaamd ProcessLis tStatusCellEditor) gekoppeld aan de kolom van de tabel.

```
tableLibSubProcessen.getColumn(0).setCellEditor(
new ProcessListStatusCellEditor());
```

Een cellEditor dient een isCellEditable, getCellEditorValue en getTableCellEditorComponent method implementatie te bevatten. Door de isEditable–method van de ProcessListStatusCellEd itor *false* terug te laten geven is de betreffende cell niet meer wijzigbaar, ook al is het bijbehorend attribuut van het view object dat wel.

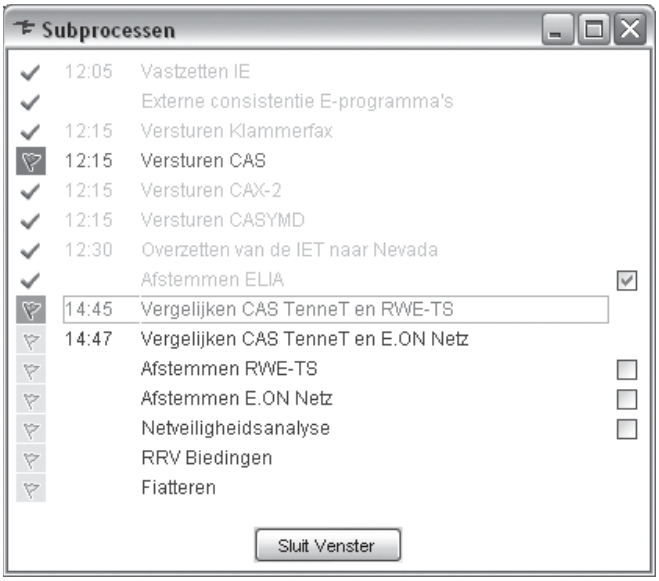
```
public class ProcessListStatusCellEditor extends AbstractCellEditor
implements TableCellEditor
{
  public ProcessListStatusCellEditor()
  {
  }

  public boolean isCellEditable(EventObject e)
```

```
  {
    // TODO:  Override this javax.swing.AbstractCellEditor method
    return false;
  }

  public Object getCellEditorValue()
  {
    // TODO:  Implement this javax.swing.CellEditor abstract method
    return null;
  }

  public Component getTableCellEditorComponent(JTable table, Object
  value, boolean isSelected, int row, int column)
  {
    // TODO:  Implement this javax.swing.table.TableCellEditor abstract
    //         method
    return null;
  }
}
```



Afbeelding 4. Paneel Subprocessen.

De laatste kolom van de JTable toont eventueel een checkbox afhankelijk van de waarde van het attribuut HandmatigJn. Ook de waarde van attribuut Status is van belang, onder andere voor het instellen van kleuren. Hiertoe is een zelfgedefinieerde cellRenderer (genaamd ProcessListHandmatigAfgevinktJnCellRenderer) gekoppeld aan de kolom van de tabel.

```
tableLibSubProcessen.getColumnModel().getColumn(4).setCellRenderer(
new ProcessListHandmatigAfgevinktJnCellRenderer(0,3));
```

De ProcessListHandmatigAfgevinktCellRenderer rendert de cellen in de kolom naar een JTextArea of een JCheckBox component.

```
public class ProcessListHandmatigAfgevinktJnCellRenderer implements
TableCellRenderer
{
  private int statusColumnIndex;
  private int handmatigJnColumnIndex;
  private JTextArea dummyTextArea = new JTextArea();
  private JCheckBox checkBox = new JCheckBox();

  public ProcessListHandmatigAfgevinktJnCellRenderer(int statusColumn,
        int handmatigJnColumn)
  {
    this.statusColumnIndex = statusColumn;
    this.handmatigJnColumnIndex = handmatigJnColumn;
    dummyTextArea.setText("");
  }

  public Component getTableCellRendererComponent(JTable table, Object
  value, boolean isSelected, boolean hasFocus, int row, int column)
  {
    // get the value of the status column of this row
    String status = (String) table.getValueAt(row, statusColumnIndex);

    ...

    // Get the value of the handmatigJn column of this row.
    // This value is used to determine what type of component
    // should be shown.
    boolean handmatigJn = false;
    Boolean booleanObject = (Boolean) table.getValueAt(row,
    handmatigJnColumnIndex);

    if (booleanObject != null)
    {
      handmatigJn = booleanObject.booleanValue();
    }

    if (handmatigJn)
    {
      // De te tonen checkbox dient unchecked te zijn
      // zodat de gebruiker het handmatig subproces kan afvinken
      boolean waarde = ((Boolean) value).booleanValue();


      checkBox.setSelected(waarde);

      return checkBox;
    }
    else
    {
      ...
      return dummyTextArea;
    }
  }
  ...
}
```

Conclusie

In het tweede artikel uit deze serie is aan de hand van een voorbeeld nader ingegaan op een manier om meerdere instanties van een JClient-paneel aan te maken, waarbij de instanties zijn gebaseerd op één orginele paneeldefinitie (inclusief panel binding) Het hierbij gebruikte mechanisme is het op runtime aanmaken van view object instanties, waarbij het uitgangspunt

één orginele view object definitie (metadata) is, gevolgd door het op runtime aanmaken van binding container instanties (panel binding), welke in de binding context zijn geregistreerd en vervolgens het voor elke binding container instantie koppelen van de juiste view object instantie aan de iterator binding alsmede het hernoemen van de iterator binding. Daarnaast is in dit artikel aan de hand van voorbeelden nader ingegaan op het gebruik van zelf gedefinieerde 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
- Adding Multiple Instances of JClient Panels, Oracle JDeveloper 10g, Release Notes, Version 9.0.5.1, March 2004

Marc Lameriks is werkzaam als senior consultant bij Capgemini (e-mail: marc.lameriks@capgemini.com).

Artikelen met praktische informatie, geschreven door en bestemd voor Oracle-professionals vindt u in het Online Archief van Array Publications. Vaktijdschriften als Database Magazine, Software Release en Java Magazine hebben hun artikelenarchief online gezet. Met een heldere zoekstructuur vindt u snel wat u zoekt op www.optimize.nl.

Oracle introduceert nieuw partnerprogramma voor MKB-markt

Oracle introduceert een nieuw partnerprogramma voor de MKB-markt: Oracle Partner Network (OPN) QuickStart Plus. Het programma is bedoeld voor nieuwe VAR's (Value Added Resellers) en ISV's (Independent Software Vendors) die Oracle's database en applicatieserver, de Standard Edition One en Standard Edition, willen verkopen. Oracle wil hiermee zijn partnerkanaal dat zich richt op het leveren van oplossingen en diensten voor het MKB verder uitbreiden. Het programma wordt geïntroduceerd in Europa, het Midden Oosten en Afrika (EMEA).
Partners die deelnemen aan het OPN QuickStart Plus programma hebben volledige toegang tot het OPN Competency Center, kunnen technologie-trainingen volgen, ontvangen verkoop- en marketing materiaal en krijgen speciale toolkits voor ontwikkelaars. Daarnaast hebben ze toegang tot de online bibliotheek van de Oracle University en krijgen ze korting op het volgen van lessen op de Oracle University. Deelname aan het programma kost 250 euro per jaar. Voor dit bedrag kunnen nieuwe partners deelnemen aan een seminar voor de MKB-markt en krijgen ze een MKB-verkooptraining. Daarnaast worden ze opgenomen in het incentive programma, worden ze onderdeel van lokale verkoopcampagnes en krijgen ze ondersteuning voor migraties en tools, waaronder migration centers voor ISV's.
OPN QuickStart Plus bouwt voort op het succes van OPN QuickStart, waar meer dan 400 partners aan deelnamen sinds de introductie in september 2004. Met OPN QuickStart kunnen partners Standard Edition One products aanbieden aan de MKB-markt. Met QuickStart Plus kunnen ze ook de Standard Edition producten verkopen. Oracle stelt zich ten doel in het komende halfjaar 500 partners te recruteren via het nieuwe QuickStart Plus-programma.

Oracle Database 10g Express Edition gratis verkrijgbaar

Oracle maakte eind februari de algemene beschikbaarheid bekend van de Oracle Database 10g Express Editie (Oracle Database XE), de gratis starter edition van de Oracle database. Sinds het bèta-debuut in oktober 2005 doet Oracle Database XE het goed onder developers, studenten en solution providers, getuige de honderdduizenden downloads.
Oracle Database XE is gebouwd op dezelfde code-base als Oracle Database 10g Release 2 en heeft dezelfde performance, betrouwbaarheid en security-functionaliteit. Dankzij de compatibiliteit met de volledige Oracle Database-productfamilie maakt Oracle Database XE het gebruikers eenvoudig om klein te beginnen en te upgraden naar andere edities van de database, wanneer dat noodzakelijk is. Ontwikkelaars kunnen via Oracle Database XE maximaal voordeel halen uit Oracle Application Express voor snelle ontwikkeling en deployment van webgebaseerde applicaties. Oracle Database XE is beschikbaar op 32-bit Windows en een serie Linux-besturingssystemen waaronder Mandriva Linux 2006 Power Pack+, Novell's SUSE Linux Enterprise Server 9 en SUSE Linux 10, Red Hat Enterprise Linux 4, Fedora en Ubuntu. De software kan gratis worden gedownload van het Oracle Technology Network: www.oracle.com/technology/xe. Tevens biedt Oracle een gratis online forum voor geregistreerde gebruikers om onderwerpen te bediscussiëren die betrekking hebben op Oracle Database XE-gebruik. Het forum wordt gehost en gemonitord door Oracle Database-experts en is te vinden op www.oracle.com/technology/products/database/xe/forum.html.