Oracle WebForms en Java UI Components

Eigen componenten in Oracle WebForm Applet

Met de komst van Forms 9i is aantal mogelijkheden, om de standaard Forms User Interface aan te passen, uitgebreid. In dit artikel zullen aan de hand van voorbeelden zowel de reeds bestaande als ook de nieuwe mogelijkheden nader worden toegelicht.

Oracle Forms biedt de mogelijkheid om de standaard Forms User Interface aan te passen middels gebruik van zelfgemaakte Java componenten. Vanaf Forms 6i wordt hiertoe het "Pluggable Java Components (PJC)" mechanisme aangeboden en vanaf Forms 9i is hier het "Enhanced JavaBean Support" mechanisme aan toegevoegd. Bij de voorbeelden is steeds tabel EMP als basis gebruikt.

Pluggable Java Components

Pluggable Java Components bieden een "low level API", waarmee het mogelijk is de Forms Java client uit te breiden. Elke standaard Oracle Forms UI component, zoals aanwezig in Forms Builder (bijvoorbeeld een Button of een Text Item), heeft een gelijkwaardige vertegenwoordiger in Java. Deze vertegenwoordiger (Oracle Forms Java UI component) is daarbij gebaseerd op het zogenaamde "Java lightweight component model".
Elke Oracle Forms Java UI component is gebouwd met gebruikmaking van twee verschillende classes (zijnde een variant van de standaard Model-View-Controller architectuur):

- De Handler class (handelend als Model en Controller) welke verantwoordelijk is voor het onderhouden van de huidige waarde van gegevens alsmede het besturen van de visuele weergave van de gegevens.
- De View class (handelend als View) welke verantwoordelijk is voor het op enige wijze representeren van het gegeven naar de gebruiker en het afhandelen van acties van de gebruiker.

Zoals reeds in mijn eerder verschenen artikel "Oracle WebForms regelt interactie met externe toepassingen" genoemd, is de techniek die hierbij gebruikt wordt: het instellen en opvragen van kenmerken ("properties"), het aanroepen van functies/procedures ("methods") en het voortbrengen en afhandelen van gebeurtenissen ("events"). Voor nadere informatie omtrent deze techniek verwijs ik dan ook naar dat artikel.

Eisen gesteld aan een PJC

Een PJC dient een implementatie van de IView interface te bevatten en moet ondersteuning bieden aan een vooraf gedefinieerde set van kenmerken ("properties") en gebeurtenis afhandelaars ("events listeners") specifiek voor het type item waarvoor de PJC wordt gebruikt. Voor een uitgebreid overzicht hiervan verwijs ik naar het Oracle Form Builder online help onderwerp "Properties and listeners needed for user interface component classes". Voor bijvoorbeeld Buttons zijn de volgende specifieke kenmerken en gebeurtenis afhandelaars verplicht:

Properties	Type	Requirements
IMAGE	Image	Gettable, Settable
IS_DEFAULT	Boolean	Gettable, Settable
LABEL	String	

Listeners
ActionListener

Daar bovenop zijn de standaardkenmerken en gebeurtenis afhandelaars verplicht welke voor alle interface elementen gelden, zoals bijvoorbeeld:

Properties	Type	Requirements
BACKGROUND	Color	Gettable, Settable

Listeners
FocusListener, KeyListener, MouseListener, MouseMotionListener

Een PJC maken

Er zijn een aantal manieren om in Java een PJC te maken ten bate van gebruik binnen een Oracle Webform Applet, te weten:

1. Gebruik maken van subclassing van een standaard Oracle Forms Java UI component.

```
public class MyClass extends VButton
```

Deze manier is geschikt indien de JavaBean slechts een kleine wijziging betreft van een standaard Oracle Forms Java UI component en waarbij de JavaBean uitsluitend in een Oracle Forms omgeving gebruikt zal worden.

2. Gebruik maken van een (niet-standaard Oracle Forms) Java UI component en hierbij de IView interface te implementeren:
   • door subclassing van de VBean class

```
public class MyClass extends VBean
```

   Waarbij binnen de class de Java UI component geïnstantieerd wordt, bijvoorbeeld:

```
slHorizontal = new JSlider(JSlider.HORIZONTAL);
```

   • binnen de class zelf

```
public class MyClass extends JSlider implements IView
```

Deze manier is geschikt indien de JavaBean helemaal vanaf niets gebouwd moet worden (of wanneer de broncode van een JavaBean beschikbaar is) en waarbij de JavaBean uitsluitend in een Oracle Forms omgeving gebruikt zal worden.

3. Gebruik maken van een container/wrapper class. Het implementeren van de IView interface kan hierbij gebeuren:
   • door subclassing van de VBean class

```
public class MyWrapperClass extends VBean
```

   • binnen de class zelf

```
public class MyWrapperClass implements IView
```

Deze manier is geschikt voor gebruik van kant-en-klare JavaBeans (meestal van derden) waarvan de broncode niet beschikbaar is en de JavaBean zelf voor algemeen gebruik bedoeld is (dus ook voor niet Oracle Forms omgevingen).

4. Het maken van een nieuw soort interface element. Dit valt verder buiten de scope van dit artikel.

PJC gemaakt via subclassing van een standaard Oracle Forms Java UI component

Maak met behulp van subclassing gebruik van een standaard Oracle Forms Java UI component en gebruik indien nodig de setProperty() en/of getProperty() functies om kenmerken te beïnvloeden.

De standaard Oracle Forms Java UI componenten zijn:
• oracle.forms.ui.VButton.class
• oracle.forms.ui.VCheckbox.class
• oracle.forms.ui.VComboBox.class
• oracle.forms.ui.VImage.class
• oracle.forms.ui.VPopList.class
• oracle.forms.ui.VRadioButton.class
• oracle.forms.ui.VRadioGroup.class
• oracle.forms.ui.VTextArea.class
• oracle.forms.ui.VTextField.class
• oracle.forms.ui.VTList.class

Deze Java componenten zijn terug te vinden in het standaard meegeleverde Java archief bestand f60all.jar (Forms 6i).

Button PJC in een single-record Form

Als voorbeeld van subclassing wordt een button gebruikt met een aangepast uiterlijk (linkerzijde afgerond) en standaard gedrag, welke een subclass is van VButton (zie afbeelding 1).

```
public class MyButton extends VButton
{

  ..
}
```

Alle vereisten die gelden voor een PJC worden in dit geval door de VButton class verwezenlijkt (waaronder het implementeren van de IView interface en het instellen/opvragen van properties "IMAGE", "IS_DEFAULT" en "LABEL").

Afbeelding 1. Button PJC in een single-record Form

Handelingen verricht aan de forms kant
Er is een item BUTTON1 aangemaakt met als kenmerken Item Type = Push button en Implementation Class = cgey.nl.demo.MyButton.

Handelingen verricht aan de Java kant
In de init procedure wordt de setLeftmost procedure (standaard aanwezig in de VButton class) aangeroepen.

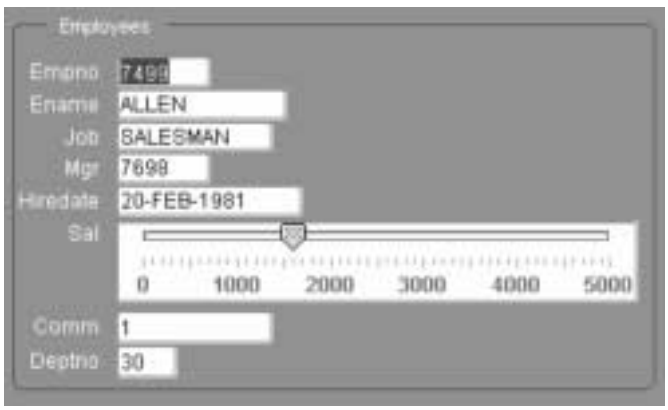
```
public void init(IHandler handler)
{
  super.init(handler);
  mHandler = handler;

  setLeftmost(true);
}
```

PJC op basis van (niet-standaard Oracle Forms) Java UI component

Maak gebruik van een (niet-standaard Oracle Forms) Java UI component en implementeer hierbij de IView interface. Het implementeren van de IView interface geschiedt het eenvoudigst door de PJC als subclass van class VBean te definiëren. Gebruik indien nodig de setProperty() en/of getProperty() functies om kenmerken te beïnvloeden.

Slider PJC in een single-record Form

Als voorbeeld van een PJC gemaakt op basis van een (niet-standaard Oracle Forms) Java UI component wordt een slider gebruikt met een eigen uiterlijk en gedrag (zie afbeelding 2).



Afbeelding 2. Slider PJC in een single-record Form

```
public class MySlider extends VBean implements ChangeListener
{
  private JSlider slHorizontal;

  ..
}
```

Waar het Enhanced JavaBean Support mechanisme niet bruikbaar is, biedt het PJC mechanisme uitkomst

Alle vereisten gesteld aan een PJC worden in dit geval door de VBean class verwezenlijkt (waaronder het implementeren van de IView interface).

Handelingen verricht aan de forms kant
Bij item SAL zijn als kenmerken ingesteld Item Type = Bean Area, Implementation Class = cgey.nl.demo.MySlider en Database Item = Yes. Er is op form niveau een WHEN-NEW-FORM-INSTANCE trigger aangemaakt, om de slider mee in te stellen.

```
begin
  set_custom_property('EMP.SAL',1,'SLIDER_MINIMUM',0);
  set_custom_property('EMP.SAL',1,'SLIDER_MAXIMUM',5000);
  set_custom_property('EMP.SAL',1,'SLIDER_VALUE',0);
  set_custom_property('EMP.SAL',1,'SLIDER_MAJORTICKSPACING',1000);
  set_custom_property('EMP.SAL',1,'SLIDER_MINORTICKSPACING',100);
end;
```

Om te kunnen reageren op het instellen van een nieuwe waarde middels de slider is op form niveau een WHEN-CUSTOM-ITEM-EVENT trigger aangemaakt.

```
declare
  eventName varchar2(30);
  value number;
begin
  eventName := :system.custom_item_event;
  if (eventName = 'SLIDER_STATECHANGED')
  and (:system.record_status not in ('NEW','QUERY'))
  then
    value :=
to_number(get_custom_property('EMP.SAL',1,'SLIDER_VALUE'));
    :EMP.SAL := value;
  end if;
end;
```

Handelingen verricht aan de Java kant
In de init procedure wordt het object van class JSlider geïnstantieerd.

```
public void init(IHandler handler)
{
  super.init(handler);
  mHandler = handler;

  slHorizontal = new JSlider(JSlider.HORIZONTAL);
  add(slHorizontal,BorderLayout.SOUTH);
  slHorizontal.setSnapToTicks(true);
  slHorizontal.setPaintTicks(true);
  slHorizontal.setBackground(Color.white);
  slHorizontal.addChangeListener(this);
  slHorizontal.setPaintLabels(true);
}
```

Naast het opvragen van zelf geregistreerde, alsmede standaard en specifieke kenmerken is het ook mogelijk deze in te stellen.

```
public boolean setProperty(ID pid, Object value)
{
  if (value != null)
  {
    String data = String.valueOf(value);

    if (pid == p_SLIDER_MINIMUM_ID)
    {
      iMinimum_waarde = new Integer(data);
      slHorizontal.setMinimum(iMinimum_waarde.intValue());
      return true;
    }
    ..
    else if (pid == p_SLIDER_VALUE_ID)
    {
      iValue_waarde = new Integer(data);
      slHorizontal.setValue(iValue_waarde.intValue());
      return true;
    }
    ..
    else
    {
      // Laat de VBean superclass de andere kenmerken afhandelen
      return(super.setProperty(pid,value));
    }
  }

  return false;
}
```

Het instellen van een nieuwe waarde middels de slider wordt door de stateChanged procedure afgevangen en middels een CustomEvent aan het WebForm doorgegeven waar de WHEN-CUSTOM-ITEM-EVENT trigger wordt geactiveerd.

```
public void stateChanged(ChangeEvent e)
{
  JSlider slTemp = (JSlider)e.getSource();
  iValue_waarde = new Integer(slTemp.getValue());
  CustomEvent ce = new CustomEvent(mHandler, e_SLIDER_STATECHANGED_ID);
  dispatchCustomEvent(ce);
}
```

Checkbox PJC in een multi-record Form

Als een ander voorbeeld van een PJC gemaakt op basis van een (niet-standaard Oracle Forms) Java UI component wordt een checkbox gebruikt met een eigen uiterlijk (plaatje in plaats van kruisje) en gedrag (zie afbeelding 3).

```
public class MyCheckBox extends JCheckBox implements IView,
ItemListener
{

  ..
}
```

Alle vereisten gesteld aan een PJC worden in dit geval door de class zelf verwezenlijkt.

Handelingen verricht aan de forms kant
Bij item PRESENT zijn als kenmerken ingesteld Item Type = CheckBox, Implementation Class = cgey.nl.demo.MyCheckBox, Value when Checked = 1, Value when Unchecked = 0 en Database Item = Yes.

Handelingen verricht aan de Java kant
In de init procedure worden de benodigde plaatjes ingelezen.

```
public void init(IHandler handler)
{
  mHandler = handler;

  try
  {
    java.net.URL url = getClass().getResource("green.gif");
    img = Toolkit.getDefaultToolkit().getImage(url);
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
  imgIconYes = new ImageIcon(img);

  try
  {
    java.net.URL url = getClass().getResource("red.gif");
    img = Toolkit.getDefaultToolkit().getImage(url);
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
  imgIconNo = new ImageIcon(img);
}
```

Het instellen van een nieuwe waarde middels de checkbox wordt door de itemStateChanged procedure afgevangen, welke het bijbehorende plaatje en tekst instelt.

Afbeelding 3. Checkbox PJC in een multi-record Form

```
public void itemStateChanged(ItemEvent e)
{
  if (isCheckBoxSelected() == false)
  {
    setCheckBoxSelected(true);
  }
  else
  {
    setCheckBoxSelected(false);
  }
}
```

Aangezien in dit voorbeeld de IView interface door de class zelf is geïmplementeerd, zijn de onderstaande (door de IView interface verplichte) procedures opgenomen.

```
public Object getProperty(ID pid)
{
  if (pid == ID.BACKGROUND)
  {
    return getBackground();
  }
  ..
}

public boolean setProperty(ID pid, Object value)
{
  ..
    if (pid == ID.BACKGROUND)
    {
      Color bgColor = (Color) value;
      setBackground(bgColor);
      return true;
    }
    ..
    else if (pid == ID.VALUE)
    {
      setCheckBoxSelected(((Boolean)value).booleanValue());
      return true;
    }
  ..
}

public void addListener(Class type, EventListener listener)
{
  ..
  else if ( type == ItemListener.class )
  {
    addItemListener((ItemListener) listener);
    addItemListener(this);
  }
}

public void removeListener(Class type, EventListener listener)
{
  ..
}

public void add(Object child, int index)
{
  ..
}

public void remove(Object child)
{
  ..
}

public void removeAll()
{
  ..
}
```

PJC gemaakt op basis van container/ wrapper class

Maak gebruik van een container/wrapper class en implementeer hierbij de IView interface. Het implementeren van de IView interface geschiedt het eenvoudigst door de PJC als subclass van class VBean te definiëren. Gebruik indien nodig de setProperty() en/of getProperty() functies om kenmerken te beïnvloeden.

Textfield PJC in een single-record Form

Als voorbeeld van een PJC gemaakt op basis van een container/ wrapper class wordt een textfield gebruikt met een eigen uiterlijk en gedrag (zie afbeelding 4 en 5). Hierbij wordt een zogenaamde voorwaardelijke opmaak toegepast. Dat wil zeggen dat dit, afhankelijk van de inhoud van het textfield, een bepaalde achtergrond kleur krijgt (groen of oranje).

```
public class MyWrapper extends VBean
{
  private ThirdPartyBean bean;
  ..
}
```

Een PJC dient een implementatie van de IView interface te bevatten

Alle vereisten gesteld aan een PJC worden in dit geval door de VBean class verwezenlijkt (waaronder het implementeren van de IView interface).

Afbeelding 4. Textfield PJC in een single-record Form

Afbeelding 5. Textfield PJC in een single-record Form

De hierbij gebruikte JavaBean afkomstig van derden ziet er als volgt uit:

```
package leverancier.nl;

import javax.swing.*;
import java.awt.*;

public class ThirdPartyBean extends JTextField
{
  private Color color = Color.green;
  private String sText;

  public ThirdPartyBean()
  {
    setBackground(color);
  }

  public Color getColor()
  {
    return color;
  }

  public void setColor(Color newColor)
  {
    color = newColor;
    setBackground(color);
  }

   public String getText()
  {
    return sText;
  }

  public void setText(String sValue)
  {
    sText = sValue;
    super.setText(sValue);
  }
}
```

Handelingen verricht aan de forms kant
Bij item SAL zijn als kenmerken ingesteld Item Type = Text Item, Implementation Class = cgey.nl.demo.MyWrapper en Database Item = Yes.

Handelingen verricht aan de java kant
In de init procedure wordt de JavaBean (ThirdPartyBean) geïnstantieerd.

```
public void init(IHandler handler)
{
  super.init(handler);
  mHandler = handler;

  bean = new ThirdPartyBean();
  add(bean);
}
```

Naast het opvragen van de standaard kenmerken is het ook mogelijk specifieke kenmerken op te vragen.

```
public Object getProperty(ID pid)
{
  ..
  else if (pid == ID.VALUE)
  {
    return(bean.getText());
  }
  ..
}
```

Naast het instellen van de standaard kenmerken is het ook mogelijk specifieke kenmerken in te stellen.

Package FBean maakt via PL/SQL interactie mogelijk met een JavaBean

```
public boolean setProperty(ID pid, Object value)
{
  if (value != null)
  {
    String data = String.valueOf(value);

    ..
    else if (pid == ID.VALUE)
    {
      double dbData;
      bean.setText(data);
      if (data.length() > 0)
      {
        dbData = (new Double(data)).doubleValue();
        if (dbData > 3000)
        {
          bean.setColor(Color.orange);
        }
        else
        {
          bean.setColor(Color.green);
        }
      }
      return true;
    }
    ..
  }

  return false;
}
```

Enhanced JavaBean Support

In tegenstelling tot PJC's hoeft er bij de "Enhanced JavaBean Support" geen enkele extra Java code geschreven te worden. Vanaf Forms 9i is er een package genaamd FBean welke via PL/SQL interactie mogelijk maakt met publieke kenmerken, functies, procedures en gebeurtenissen van een JavaBean of Applet.

Textfield PJC in een single-record Form

Als voorbeeld van "Enhanced JavaBean Support" wordt een textfield gebruikt met een eigen uiterlijk en gedrag. Dit is hetzelfde voorbeeld als bij een PJC gemaakt op basis van een container/wrapper class (zie afbeelding 4 en 5).

Handelingen verricht aan de forms kant

Bij item SAL zijn als kenmerken ingesteld Item Type = Bean Area en Database Item = Yes. De Implementation Class hoeft hierbij niet ingesteld te worden. Er is op form niveau een WHEN-NEW-FORM-INSTANCE trigger aangemaakt. Om de JavaBean te registreren is de FBEAN.REGISTER_BEAN built-in gebruikt met de volgende parameters: item_name, item_instance en bean_class. Bij gebruik van een JavaBean dient ook een zogenaamde BeanInfo class aanwezig te zijn, waarin is aangegeven welke specifieke kenmerken, functies/procedures en gebeurtenissen van de JavaBean gebruikt kunnen worden.

```
begin
  FBean.register_bean('EMP.SAL',1,'cgey.nl.demo.ThirdPartyBean');
end;
```

Om de voorwaardelijke opmaak te kunnen toepassen is op block niveau een WHEN-NEW-RECORD-INSTANCE trigger aangemaakt. Om kenmerken in te stellen is de FBEAN.SET_PROPERTY built-in gebruikt met de volgende parameters: item_name, item_instance, property_name en value. Tevens is één van de zogenaamde "predefined data type encoders" gebruikt welke automatisch een object in java (bijvoorbeeld java.awt.Color) omzet naar een VARCHAR2, NUMBER of BOOLEAN type in PL/SQL en omgekeerd.

```
begin
  if not :EMP.SAL is null
  then
    if :EMP.SAL > 3000
    then
      FBean.set_property('EMP.SAL',1,'color','255 127 0');
    else
      FBean.set_property('EMP.SAL',1,'color','0 255 0');
    end if;
    FBean.set_property('EMP.SAL',1,'text',to_char(:EMP.SAL));
  end if;
end;
```

Onderdelen van de FBean package welke in bovenstaand voorbeeld niet of slechts ten dele aan de orde zijn gekomen zijn:

- het instellen en opvragen van kenmerken van de JavaBean door middel van:
  FBean.set_property / FBean.get_property en
  FBean.set_indexed_property / FBean.get_indexed_property
- het aanroepen van functies/procedures van de JavaBean middels:
  FBean.invoke en FBean.arglist
- het reageren op gebeurtenissen van de JavaBean middels:
  FBean.enable_event
- het weergeven van debug gegevens in de Java Consol middels:
  FBean.set_logging_mode
- het werken met complexe datatypen of gebeurtenissen van de JavaBean

Voor deze onderdelen verwijs ik naar het Oracle Forms Builder online-help onderwerp "How to Add JavaBeans Using Enhanced JavaBean Support".

Conclusie

Voor het aanpassen van de standaard Forms User Interface middels Java componenten zijn er momenteel een tweetal mechanismen beschikbaar, te weten: "Pluggable Java Components (PJC)" (vanaf Forms 6i) en "Enhanced JavaBean Support" (vanaf Forms 9i). Bij het beschikbaar zijn van kant en klare JavaBeans (van derden) ligt vanwege de eenvoud, het gebruik van het "Enhanced JavaBean Support" mechanisme voor de hand. Er hoeft namelijk geen extra Java code geschreven te worden en de functionaliteit kan via PL/SQL gerealiseerd worden.

Daar waar het "Enhanced JavaBean Support" mechanisme niet bruikbaar is, biedt het PJC mechanisme uitkomst. Bij een geringe aanpassing ten opzichte van de standaard User Interface kan gebruik worden gemaakt van subclassing van een stan-

Bij beschikbaarheid van kant en klare JavaBeans ligt vanwege de eenvoud, gebruik van "Enhanced JavaBean Support" voor de hand

daard Oracle Forms Java UI component (voor zover deze de gewenste functionaliteit biedt). Indien subclassing van de standaard component geen optie is, kan gebruik worden gemaakt van een (niet-standaard Oracle Forms) Java UI component. Waarbij vanwege het gemak de implementatie van de IView interface bij voorkeur met behulp van subclassing van de VBean class bewerkstelligd kan worden.

Referenties

- Using Java Components in Oracle Forms Applications, An Oracle Technical White Paper, January 2000.
- Oracle 9i Forms in a Java Wold, An Oracle Technical White Paper, July 2002.
- Oracle WebForms regelt interactie met externe toepassingen, Optimize, Februari 2003, jaargang 6, nummer 1.

Marc Lameriks
is werkzaam als Senior Consultant bij Cap Gemini Ernst & Young (e-mail: marc.lameriks@cgey.nl). Voor de volledige broncode van de voorbeelden kan contact met de schrijver worden opgenomen.