In gesprek met externe toepassingen

Oracle WebForms regelt interactie

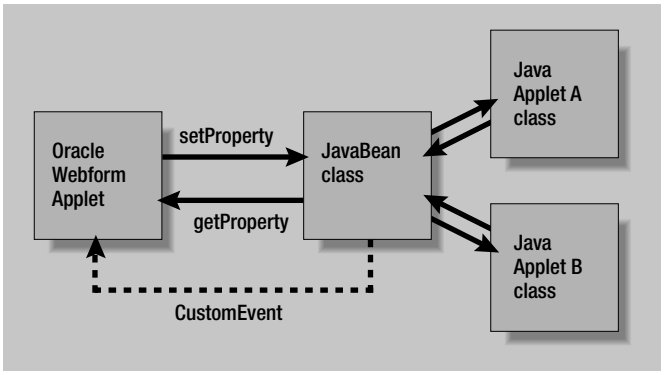
Steeds vaker wordt gekozen voor een Oracle WebForms applicatie (three tier architecture) als oplossing voor een aan vervanging toe zijnde Oracle Forms applicatie in een client server architectuur. Vaak bestaat dan wel de wens om externe toepassingen aan te kunnen sturen en interactie met Oracle WebForms te onderhouden. Dit artikel gaat in op een technische oplossing om een externe toepassing in de vorm van een Java Applet aan te sturen vanuit een Oracle WebForm Applet.

Indien een bepaalde functionaliteit in een Oracle WebForms applicatie gewenst is, welke niet met behulp van een Forms ontwikkeltool te realiseren is, is men aangewezen op andere ontwikkelomgevingen. Java speelt hierbij steeds vaker een belangrijke rol, mede door de nauwe band die Oracle hiermee inmiddels heeft (denk hierbij alleen al aan Oracle JDeveloper). De interactie van een Java Applet met een Oracle WebForm is dan ook, dankzij reeds aanwezige technieken, een relatief eenvoudige zaak. Voordeel hierbij is dat de ontwikkelaar van het Oracle WebForm nauwelijks Java kennis nodig heeft.

Vanaf Oracle Forms Server 6i is het mogelijk Java componenten ("pluggable Java components") in een form op te nemen en te gebruiken. In het algemeen gaat het hierbij om aanpassingen van het gedrag van standaard Developer Java UI componenten, zoals bijvoorbeeld een Text Item. Of het gaat hierbij om het geheel vervangen van de standaard Developer Java User Interface componenten door zelf gemaakte Java UI componenten, met hun eigen (uiterlijke) kenmerken en gedrag. Denk hierbij bijvoorbeeld aan een Check Box waarbij een plaatje in plaats van een kruisje gebruikt wordt. In grote lijnen is de techniek die hierbij gebruikt wordt: het instellen en opvragen van kenmerken ("property's"), het aanroepen van functies/procedures ("methods") en het voortbrengen en afhandelen van gebeurtenissen ("events"). Deze techniek is in één van de Oracle projecten van Koninklijke Boskalis Westminster NV toegepast om externe toepassingen (Java Applets) aan te sturen. Hierbij gaat het om het tonen van grafieken met behulp van de Java tool JClass Chart en het tonen van een driedimensionale afbeelding met behulp van Java 3D vanuit een Oracle WebForm. De benodigde data worden hierbij vanuit het Oracle WebForm doorgegeven aan de externe toepassing.

Om de communicatie tussen een Oracle WebForm Applet en een externe toepassing te bewerkstelligen zijn een aantal onderdelen van belang (zie afbeelding 1), te weten:

- Een Forms item van het type Bean Area in een Oracle WebForm
- Een JavaBean class
- Een Java Applet



Afbeelding 1. Overzicht van de communicatie tussen een OracleWebForm Applet en Java Applets

Aan de hand van een eenvoudig voorbeeld wordt nader uitleg gegeven over bovenstaande onderdelen en de onderlinge communicatie. De in afbeelding 1 weergegeven setProperty, getProperty en CustomEvent communicatielijnen zijn in het voorbeeld terug te vinden in de JavaBean class en hebben steeds een relatie met een bepaalde PL/SQL procedure in het Oracle WebForm.

Bean Area item

Het Forms item van het type Bean Area, instantiëert de JavaBean class en verzorgt daarmee de relatie tussen het Oracle Form en de JavaBean. De werkwijze voor het aanmaken van de Bean Area verschilt enigszins bij gebruik van Oracle

Developer en Oracle Designer (honderd procent generatie). Beide ontwikkeltools kunnen echter gebruikt worden. Bij gebruik van Oracle Developer (Oracle Forms Builder) kan bij het aanmaken van een nieuw item, direct Bean Area als Item Type ingesteld worden en als Implementation Class de naam van de JavaBean class (bijvoorbeeld myJavabean.class), eventueel voorafgegaan door een directory als de JavaBean class is opgenomen in een package (zie afbeelding 2).

Afbeelding 2. Property Palette van het Bean Area item in het Form

Object Library

In tegenstelling tot het gebruik van Oracle Developer kan bij het gebruik van bijvoorbeeld Oracle Designer 6i, het item niet als type Bean Area en een bepaalde Implementation Class ingesteld worden. Hiervoor dient een Object Library gebruikt te worden. Als basis voor zo'n Object Library wordt in het algemeen een objecten bibliotheek onderhoudsform ("object library maintenance form") gebruikt, van waaruit met behulp van programma FM2LIB61.EXE een Object Library aangemaakt kan worden. Oracle adviseert om ofwel een kopie te gebruiken van de standaard Object Library ofwel een eigen Object Library aan te maken. Het toevoegen van het Bean Area item aan de Object Library geschiedt simpelweg door in het objecten bibliotheek onderhoudsform een nieuw item aan te maken (met als naam bijvoorbeeld CGEYSO$MYBEAN), waarbij de Item Type en Implementation Class kenmerken gelijk zijn aan de kenmerken zoals eerder aangegeven.
In Oracle Designer kan vervolgens bij de betreffende module een Unbound item opgenomen worden met een verwijzing naar het gewenste item in de Object Library. In ieder geval dienen de kenmerken Display Type = Bean Area, Unbound Type = Custom en Display ? = Yes ingesteld te worden.

Java speelt een belangrijke rol door de nauwe band die Oracle inmiddels hiermee heeft

Aan het Bean Area item dient vervolgens op item niveau een WHEN-CUSTOM-ITEM-EVENT trigger gehangen te worden, die bijvoorbeeld PL/SQL procedure handleMyAppletEvent anroept (zie volgende alinea voor de code).

PL/SQL procedures

Om vanuit een Oracle WebForm te communiceren met de JavaBean class is een PL/SQL procedure startMyApplet gebruikt. Hierbij is de SET_CUSTOM_PROPERTY built-in gebruikt met de volgende parameters: item, row-number, prop-name en value. Beperking hierbij is dat parameter value alleen de typen VARCHAR2, NUMBER en BOOLEAN aan kan. Met behulp van deze built-in kunnen niet alleen kenmerken van de JavaBean ingesteld worden, maar ook functies/procedures van de JavaBean gestart worden. Dit gebeurt doordat de built-in de functie setProperty in de JavaBean class aanroept. Met behulp van een procedure van de JavaBean kan dan vervolgens bijvoorbeeld een Java Applet gestart worden.

```
procedure startMyApplet()
is
begin
/* Doorgeven gegevens van Oracle Forms naar de JavaBean */
set_custom_property('MYBEAN',ALL_ROWS,'MIJN_KENMERK_1','mijn waarde 1');
set_custom_property('MYBEAN',ALL_ROWS,'MIJN_KENMERK_2','mijn waarde 2');
set_custom_property('MYBEAN',ALL_ROWS,'MIJN_PROCEDURE_1',0);
set_custom_property('MYBEAN',ALL_ROWS,'MIJN_KENMERK_3','mijn waarde 3');
set_custom_property('MYBEAN',ALL_ROWS,'MIJN_PROCEDURE_2',0);
end;
```

Om in een Oracle WebForm te reageren op een gebeurtenis ("event"), voortgebracht vanuit de JavaBean class, is een PL/SQL procedure handleMyAppletEvent gebruikt. Hierbij is de GET_CUSTOM_PROPERTY built-in gebruikt met de volgende parameters: item, row-number en prop-name. Beperking hierbij is dat altijd een VARCHAR2 wordt geretourneerd. De built-in roept de functie getProperty in de JavaBean class aan.

```
procedure handleMyAppletEvent()
is
  eventName varchar2(30);
begin
  eventName := :system.custom_item_event;
  if (eventName = 'MIJN_APPLET_A_GEBEURTENIS') then
  :GLOBAL.APPLETVALUE_A := get_custom_property ('MYBEAN',1,
  'MIJN_KENMERK_4');
  else (eventName = 'MIJN_APPLET_B_GEBEURTENIS') then
  :GLOBAL.APPLETVALUE_B := get_custom_property ('MYBEAN',1,
  'MIJN_KENMERK_5');
  end if;
end;
```

Bovenstaande PL/SQL procedure wordt aangeroepen vanuit de WHEN-CUSTOM-ITEM-EVENT trigger, die op item niveau aan het Bean Area item is gehangen. In dit voorbeeld worden globale variabelen van een waarde voorzien, afkomstig van de externe toepassingen.

JavaBean class

Om interactie met een Oracle Form mogelijk te maken is in de JavaBean class een implementatie van de IView interface nodig. Deze interface bestaat uit de specificaties van functies en procedures benodigd voor de interactie. In de oracle.forms.ui.Vbean class is de implementatie van de IView interface reeds opgenomen. Alle benodigde functies en procedures zijn dus reeds aanwezig. Het is daarom verreweg het gemakkelijkst de JavaBean class als subclass van class Vbean te definiëren (dit scheelt programmeerwerk). Als de JavaBean class reeds een subclass is van een andere class bestaat de meer arbeidsintensieve mogelijkheid om de interface IView zelf te implementeren (en dus alle voor de interactie benodigde functies en procedures zelf te programmeren).

```
package cgey.nl.demo;

import Java.awt.event.*;
import oracle.forms.ui.*;
import oracle.forms.properties.*;
import oracle.forms.handler.*;

public class myJavabean extends VBean
{

  /*
  * Default constructor
  */
  public myJavabean ()
  {
  ..
  }

..
}
```

In de JavaBean class wordt elk kenmerk (“property”), elke functie/procedure (“method”) en ook elke gebeurtenis (“event”) van de JavaBean gerepresenteerd door een ID middels ID.registerProperty.

```
private String sMijnKenmerk_1_waarde;
private String sMijnKenmerk_2_waarde;
private String sMijnKenmerk_3_waarde;
private String sMijnKenmerk_4_waarde;
private String sMijnKenmerk_5_waarde;


private myAppletA maA;
private myAppletB maB;

IHandler mHandler = null;

/*
* Declaratie van property id’s tbv communicatie
* tussen Oracle Forms en de JavaBean
*/
static final ID p_MIJN_KENMERK_1_ID = ID.registerProperty
  (“MIJN_KENMERK_1”);
static final ID p_MIJN_KENMERK_2_ID = ID.registerProperty
  (“MIJN_KENMERK_2”);
static final ID m_MIJN_PROCEDURE_1_ID = ID.registerProperty
  (“MIJN_PROCEDURE_1”);
static final ID p_MIJN_KENMERK_3_ID = ID.registerProperty
  (“MIJN_KENMERK_3”);
static final ID m_MIJN_PROCEDURE_2_ID = ID.registerProperty
  (“MIJN_PROCEDURE_2”);

/*
* Declaratie van property id’s tbv communicatie
* tussen JavaBean en Oracle Forms
*/
static final ID p_MIJN_KENMERK_4_ID = ID.registerProperty
  (“MIJN_KENMERK_4”);
static final ID p_MIJN_KENMERK_5_ID = ID.registerProperty
  (“MIJN_KENMERK_5”);
static final ID e_MIJN_APPLET_A_GEBEURTENIS_ID =
  ID.registerProperty(“MIJN_APPLET_A_GEBEURTENIS”);
static final ID e_MIJN_APPLET_B_GEBEURTENIS_ID =
  ID.registerProperty(“MIJN_APPLET_B_GEBEURTENIS”);
```

In dit voorbeeld heeft de Java component (“JavaBean”) de kenmerken MIJN_KENMERK_1 tot en met MIJN_KENMERK_5 die met een getProperty en setProperty functie respectievelijk opgevraagd en ingesteld kunnen worden. Daarnaast maakt het voorbeeld gebruik van gebeurtenissen MIJN_APPLET_A_GEBEURTENIS en MIJN_APPLET_B_GEBEURTENIS en ook van de procedures MIJN_PROCEDURE_1 en MIJN_PROCEDURE_2 waarmee externe Java Applets gestart worden.

Functie getProperty

Met behulp van de getProperty functie kan Oracle Forms een kenmerk van de JavaBean opvragen. Deze functie verzorgt dus de communicatie van de JavaBean class naar Oracle Forms.

```
public Object getProperty(ID pid)
{

  if( pid == p_MIJN_KENMERK_4_ID )
  {
    return(sMijnKenmerk_4_waarde);
  }
  else if( pid == p_MIJN_KENMERK_5_ID )
  {
    return(sMijnKenmerk_5_waarde);
  }
  // Laat de VBean superclass de andere kenmerken afhandelen
  else
    return(super.getProperty(pid));
}
```

Functie setProperty

Met behulp van de setProperty functie kan een kenmerk van de JavaBean gewijzigd of geïnitialiseerd worden. Ook kan hiermee een functie of procedure gestart worden. Deze functie verzorgt dus de communicatie van Oracle Forms naar de JavaBean Class.

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

  if ( pid == p_MIJN_KENMERK_1_ID )
  {
     sMijnKenmerk_1_waarde = data;
     return true;
  }
  else if ( pid == p_MIJN_KENMERK_2_ID )
  {
     sMijnKenmerk_2_waarde = data;
     return true;
  }
  else if ( pid == m_MIJN_PROCEDURE_1_ID )
  {
     myAppletA maA = new myAppletA( sMijnKenmerk_1_waarde,
     sMijnKenmerk_2_waarde );
     maA.addWindowListener(new WindowAdapter()
     {
       public void windowClosing(WindowEvent e)
       {
         sMijnKenmerk_4_waarde = maA.geefWaarde();
         CustomEvent ceA = new CustomEvent(mHandler,
         e_MIJN_APPLET_A_GEBEURTENIS_ID);
         dispatchCustomEvent(ceA);

        maA.dispose();
         maA = null;
       }

     });
     maA.setSize(1000,500);
     maA.setVisible(true);
     maA.show();

     return true;
  }
  else if ( pid == p_MIJN_KENMERK_3_ID )
  {
     sMijnKenmerk_3_waarde = data;
     return true;
  }
  else if ( pid == m_MIJN_PROCEDURE_2_ID )
  {
     myAppletB maB = new myAppletB( sMijnKenmerk_3_waarde );
     maB.addWindowListener(new WindowAdapter()
     {
       public void windowClosing(WindowEvent e)
       {
         sMijnKenmerk_5_waarde = maB.geefWaarde();
         CustomEvent ceB = new CustomEvent(mHandler,
         e_MIJN_APPLET_B_GEBEURTENIS_ID);
         dispatchCustomEvent(ceB);

         maB.dispose();
         maB = null;
       }

     });
     maB.setSize(1000,500);
     maB.setVisible(true);
     maB.show();

     return true;
  }

  // Laat de VBean superclass de andere kenmerken afhandelen
  return(super.setProperty(pid,value));
}
```

In bovenstaand voorbeeld wordt steeds een gebeurtenis voortgebracht op het moment dat de Java Applet gesloten wordt.

Java Applets

Het is mogelijk om meerdere Applets vanuit de JavaBean class te starten, waarbij iedere Applet zijn eigen argumenten heeft waarmee die gestart kan worden. In myAppletA (zie code) zijn dit bijvoorbeeld doorgegevenWaarde_1 en doorgegevenWaarde_2 en in myAppletB is dit doorgegevenWaarde_1.

```
package cgey.nl.demo;

import Javax.swing.JFrame;

public class myAppletA extends JFrame
{

  private String mijnAppletA_waarde;

  /*
  * Default constructor
  */
  public myAppletA ( String doorgegevenWaarde_1, String
doorgegevenWaarde_2 )
  {
    this.mijnAppletA_waarde = "";
    ..
  }


  public String geefWaarde()
  {
    return this.mijnAppletA_waarde;
  }

  public void zetWaarde( String doorgegevenWaarde)
  {
    this.mijnAppletA_waarde = doorgegevenWaarde;
  }

  ..
}
```

Oracle JInitiator

Onderdeel van een Oracle WebForms applicatie is een applicatie server, waarbij vaak gebruik wordt gemaakt van Oracle JInitiator. Hiermee bestaat de mogelijkheid de "Oracle certified Java Virtual Machine (JVM)" in plaats van de standaard JVM van de webbrowser te gebruiken. In het configuratie bestand van JInitiator kan opgegeven worden dat bepaalde Java archief bestanden ("JAR files") automatisch op de client machine moeten worden neergezet. Door de JavaBean class en andere benodigde Java classes in één of meerdere Java archiefbestanden te zetten en deze bestanden op te geven in het configuratiebestand van JInitiator, is ervoor gezorgd dat de interactie tussen een Oracle

Forms Applet en de Java Applets mogelijk is. Aandachtspunt hierbij is dat de versie van Oracle JInitiator en de daarin ondersteunde JDK versie overeenstemt met de vereiste JDK versie van gebruikte Java tools (zoals bijvoorbeeld Java3D) benodigd voor de externe toepassingen.

Praktijktoepassing

In het zogenaamde BOEG-project, één van de Oracle projecten van Koninklijke Boskalis Westminster NV is bovenstaande techniek toegepast. Koninklijke Boskalis Westminster NV is een wereldwijd opererend baggerbedrijf. Dit BOEG-project behelst de herbouw van een aantal oudere (character-based) Oracle applicaties waarmee de productie en kosten van baggerprojecten kunnen worden ingeschat. Door verouderde technologie en het grote aantal wijzigingen die in de loop der tijd waren ingevoerd was de onderhoudbaarheid sterk afgenomen. Nieuwe wijzigingen konden soms slechts met grote inspanning worden doorgevoerd. Met name ten behoeve van verbeterde onderhoudbaarheid van deze applicaties heeft Koninklijke Boskalis Westminster NV ervoor gekozen alle verouderde applicaties te laten herbouwen. Daarbij is gekozen voor een Oracle WebForms applicatie (three tier architecture).

Een belangrijk onderdeel van het BOEG-project is het tonen van grafieken. Op basis van een aantal eisen is de keuze hierbij gevallen op het Java-tool JClass Chart. Door de grafiek te integreren in een Java Applet en gebruik te maken van de JavaBean is het mogelijk door op een button te klikken in het WebForm, op basis van ingevoerde data en daaruit berekende data, grafieken te tonen (zie afbeelding 4). Het voordeel hierbij is dat zowel het doorgeven van de benodigde data als het starten van de externe toepassing door middel van één en hetzelfde mechanisme geschiedt, zoals besproken.

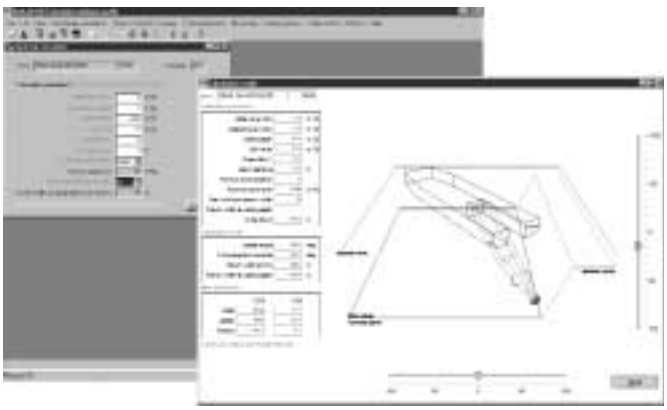
Daarnaast is bovenstaande techniek toegepast om een driedimensionale afbeelding van een snijkopzuiger[1] met behulp van Java 3D te tonen. Door op een knop te klikken wordt op basis van ingevoerde data en daaruit berekende data, een driedimensionale afbeelding van een snijkopzuiger getoond. Hierbij is het tevens mogelijk om de snijkopzuiger te roteren om twee assen (zie afbeelding 5).

Conclusie

De interactie van een externe toepassing (Java Applet) met een Oracle WebForm is, dankzij een techniek gericht op het opnemen van JAVA UI componenten in een Oracle WebForm Applet, een relatief eenvoudige zaak. Voordeel hierbij is dat de ontwikkelaar van het Oracle WebForm, nauwelijks Java kennis nodig heeft. Bij de techniek spelen een drietal onderdelen een belangrijke rol: een Forms item van het type Bean Area, een JavaBean class en een Java Applet. In grote lijnen is de techniek die hierbij gebruikt wordt: het instellen en opvragen van kenmerken ("property's"), het aanroepen van functies/procedures ("methods") en het voortbrengen en afhandelen van gebeurtenissen ("events"). Het voordeel bij het gebruik van deze techniek is dat zowel het doorgeven van de benodigde data als het starten van de externe toepassing via hetzelfde mechanisme plaatsvindt.

[1] baggermolen die de op te baggeren grond stukmaakt met een snijkopschroef

Afbeelding 4. Oracle WebForm Applet Boskalis DD Calculation pump predict



Afbeelding 5. Oracle WebForm Applet Boskalis DD Calculation minimum profile

Referenties

- Using Java Components in Oracle Forms Applications, An Oracle Technical White Paper, January 2000.
- Boskalis DD Calculation pump predict, Koninklijke Boskalis WestMinster NV, Papendrecht.
- Boskalis DD Calculation minimum profile, Koninklijke Boskalis WestMinster NV, Papendrecht.

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