JDAPI voor aanpassen Forms-modules

Waardevolle functionaliteit voor migratietrajecten

Sinds Oracle Forms 9i is het mogelijk om vanuit Java-code Oracle Forms-modules te openen en aan te passen. Met behulp van de Forms Java Design-Time API (JDAPI) is het mogelijk geworden om Oracle Forms-modules batchgewijs aan te passen. Met name in migratietrajecten is dit waardevolle functionaliteit. Dit artikel beschrijft de werking van JDAPI aan de hand van een aantal praktijkvoorbeelden.

De Forms Java Design-time API is de Java-API voor het openen, creëren, aanpassen, opslaan en compileren van Oracle Forms-modules. JDAPI is een uitbreiding van de reeds bestaande C-API. Met JDAPI kan men dezelfde bewerkingen op Oracle Forms-modules uitvoeren als met Forms Builder. Zo kan men vanuit Java-code nieuwe objecten, zoals blocks en items, creëren binnen een Oracle Forms-module en property's van bestaande objecten aanpassen.

Kennis van JDAPI

Om gebruik te maken van JDAPI is zowel Java- als Oracle Forms-kennis nodig. Oracle Forms-ontwikkelaars met basale Java-kennis kunnen vrij snel met JDAPI aan de slag. Alle Java-functies voor het openen en aanpassen van Oracle Forms modules vindt u in de Java-package f90jdapi.jar. Deze package wordt standaard met Oracle Forms meegeleverd. Op basis van de op OTN beschikbare voorbeelden en documentatie in de Forms Builder help-functie kan men snel een programmaatje maken dat bijvoorbeeld de namen van alle blocks en items wegschrijft in een tekstbestandje.

Beperkingen JDAPI

JDAPI werkt probleemloos voor Forms (fmb)- en menu (mmb)-modules. Bij het aanpassen van library's liepen we tegen het probleem aan dat de save-method een foutmelding geeft. Vreemd genoeg werkt de save niet. De workaround voor dit probleem is om zelf een tekstversie van de pll weg te schrijven (een pld) om deze vervolgens via een standaard Oracle Forms batch-script weer te converteren naar een pll. Deze workaround is een beetje omslachtig, maar werkt wel. Overigens wordt deze workaround ook door Oracle zelf gebruikt in hun Oracle-migration assistance.

Tijdrovend

Tijdens migraties van Oracle Forms 4.5 of 6i modules naar Oracle Forms 10g moet men vaak soortgelijke aanpassingen doen in meerdere modules. Met Forms Builder is dit een zeer saaie en tijdrovende klus. Met behulp van een Java-applicatie die gebruik maakt van JDAPI kunnen Oracle Forms modules snel en eenvoudig gemigreerd worden zonder handmatig alle modules te moeten openen in Forms Builder.

> ***Om JDAPI te kunnen gebruiken, is zowel Java- als Oracle Forms-kennis vereist***

Voorbeeldcode

Aan de hand van voorbeelden uit Java-code die wij in de praktijk gebruiken, licht ik hieronder het gebruik van JDAPI toe.

Openen en opslaan van modules

In onderstaande voorbeeldcode wordt onze Java-class getoond voor het openen, saven en migreren van Forms-modules. De onderstaande Java-class bevat een constructor (ComtForm) die de Forms-module opent. Deze class bevat verder een method voor het saven van de Forms-module en een method voor het migreren van een Forms-module. De method `ShowBlocksAndItems` wordt in een volgend voorbeeld verder uitgewerkt.

```
package oreade.jdapitool;

import oracle.forms.jdapi.*;
import java.io.*;

public class ComtForm
{  FormModule   fmb;              // JDAPI object voor forms modules
   File          sourceDir;
   String        fullname;

   /* Deze constructor zorgt ervoor dat bij de aanroep van de ComtForm
class
    * automatisch de form module geopend wordt.
    */
   public ComtForm(String modulename, File inDir, String dbconnection)
   {
     /* Bewaren inDir als sourceDir */
     sourceDir = inDir;
     fullname  = modulename;

     /* Initialiseer connectie met database (t.b.v. het compileren) */
     Jdapi.connectToDatabase(dbconnection);

     // Open file
     try
     {
        fmb = FormModule.open(inDir.toString() + inDir.separatorChar +
modulename);
     }
     catch(JdapiException e)
     {
        System.out.println("ERROR: Probleem bij openen Form Module:
"+e.getMessage());
        System.exit(1);
     }
   }

   /* Method voor het opslaan van de module*/
   public void Save()
   {
      try
      {
         fmb.save(sourceDir.toString() + sourceDir.separatorChar +
fullname);
      }
      catch(JdapiException e)
      {
         System.out.println ("ERROR: Probleem bij opslaan Form Module:
"+e.getMessage());
         System.exit(1);
      }
   }

   /* Methode om alle blocken en items van een form te tonen
    */
   public void ShowBlocksAndItems ()
   {  /* Code van ShowBlocksAndItems volgt in een volgend voorbeeld */
      ..
   }

}
```

Men kan de bovenstaande Java-class op de volgende manier aanroepen:

```
package oreade.jdapitool;

import oracle.forms.jdapi.*;
import java.io.*;

public class ShowObjectsFormsModule
{

   public static void main(String[] args)
 {

     // Plaats command line variabelen in variabelen
     String modulename   = args[0];
     File inDir           = new File(args[1]);
     String dbconnection = args[2];

     ComtForm form = new ComtForm(modulename, inDir, dbconnection);
     form.ShowBlockAndItems();
     form.Save();
   }
}
```

De Java-class ShowObjectsFormsModule bevat een main method die vanaf de commandline aangeroepen kan worden met:

```
Java ShowObjectsFormsModule test.fmb c:\temp test/test@ontw
```

Doorlopen van items

Om door de objecten van een Forms-module heen te kunnen lopen dien je gebruik te maken van de Jdapilterator. De Jdapilterator is een beetje te vergelijken met een PL/SQL-cursor. Als eerste stap dient men de Jdapilterator te initialiseren. Met het commando `JdapiIterator blockForm = fmb.getBlocks();` vul je de iterator met alle blocken van een form. Vervolgens is het mogelijk om door alle blocken heen te lopen met het commando: `while (blockForm.hasNext()) { Block currblok = (Block)blockForm.next(); ..... }`. De methods van de Jdapilterator class leveren objecten op van het type Object. Dit object moet gecast (geconverteerd) worden naar het juiste objecttype. Dit kan door er tussen haakjes het juiste type voor te zetten, bijvoorbeeld: `(Block)blockForm.next();`.
Onderstaande voorbeeldcode komt uit de ComtForm-Class, en toont hoe je door alle blocks en items van een Forms-module kunt lopen. Het voorbeeld schrijft de namen van alle blocks en items weg naar de standaard-output.

```
   private void ShowBlocksAndItems()
   {
      /* Doorloop alle blocken van een forms module */
```

```
    JdapiIterator blockForm = fmb.getBlocks();

    while (blockForm.hasNext())
    {
       Block currblok = (Block)blockForm.next();
       System.out.println ("Block : "+currblok.getName());



       /* Doorloop alle items van block */
       JdapiIterator itemBlock = currblok.getItems();


       while (itemBlock.hasNext())
       {
          Item curritem = (Item)itemBlock.next();
          System.out.println ("Item  : "+curritem.getName());


       }
    }
 }
```

Opvragen en aanpassen

Nu we weten hoe we door alle objecten heen kunnen lopen, willen we natuurlijk ook de property's van objecten kunnen opvragen en kunnen aanpassen. Voor elke property die je in Oracle Forms Builder kunt aanpassen, is een get- en set-method aanwezig. Met de get-methods kun je property's opvragen en met de set-methods kun je property's wijzigen. Om bijvoorbeeld het itemtype van een item op te vragen kan men de method getItemType aanroepen. Om de achtergrondkleur van een item aan te passen wordt de method setBackColor gebruikt. In de JDAPI-documentatie zie je welke methods je kunt gebruiken. In het nu volgende codevoorbeeld wordt getoond hoe deze methods gebruikt worden om de achtergrondkleur van een button aan te passen.

```
    /* ITTY_PB_CTID = Pushbutton */
    if (curritem.getItemType() == JdapiTypes.ITTY_PB_CTID)
    {
       curritem.setBackColor("gray20");
    }
```

Aanmaken nieuwe objecten

Tenslotte wil ik nog een klein voorbeeld geven over het maken van nieuwe objecten. Dit kan men eenvoudig doen door het aanroepen van de default-constructor van de bij het object behorende Java-class. Om een nieuw item aan te maken roep je bijvoorbeeld de constructor van Java-class item aan: `Item myNewItem = new Item(<block object>,<naam nieuw item>);`. Nadat het item is aangemaakt, worden alle property's ingesteld met de methods uit de Java-class Item. Onderstaand codevoorbeeld toont hoe een nieuwe library gekoppeld kan worden aan een scherm.

```
  public void AttachLibrary(String lib)
  {
     // Attach library lib aan form
     try
     {
       AttachedLibrary libr = new AttachedLibrary(fmb,lib.toLower-
Case());
     }
     catch(JdapiException e)
     {
      System.out.println ("ERROR: Kan library niet attachen ("+e.get-
Message()+")");
     }
  }
```

Conclusie

JDAPI is een krachtig hulpmiddel wanneer men in meerdere Forms-modules dezelfde wijzigingen wil doorvoeren. Alles wat je met Forms Builder kunt doen, is ook mogelijk met JDAPI. Sterker nog, je kunt met JDAPI zelfs meer doen dan met Forms Builder. Een voorbeeld hiervan is het converteren van Forms-modules naar een XML-bestand. Zoals genoemd kent JDAPI een aantal beperkingen, maar voor alle beperkingen waar wij tegenaan gelopen zijn, is een workaround beschikbaar. Voor het bouwen van bijvoorbeeld een migratietool is JDAPI zeker geschikt.

Arno van Haren is systeemontwerper en senior Oracle-engineer bij de Caesar Groep. Hij werkt vanaf 1996 met de Oracle-ontwikkelomgeving en is betrokken bij diverse migratietrajecten met JDAPI. E-mail: a.haren@caesar.nl.