BPM Suite 11g und Oracle Forms

Gert Schüßler
Oracle Deutschland B.V. & Co. KG
Hamburg

Schlüsselworte:

BPM, BPM Suite 11g, BPMN, OMG, Java, API, Human Task, Workspace, Forms, JDeveloper, BPM Studio

Einleitung

Oracle Forms ist eine 4GL Entwicklungsumgebung zur Erstellung von Dialogapplikationen, mit denen Informationen in Datenbanken manipuliert werden. Die Historie von Oracle Forms reicht weit in die Vergangenheit zurück. Frühe Versionen von Oracle Forms (2.x,3.x) liefen im Character Mode, spätere Versionen (4.x – 6.x) liefen mit einer grafischen Oberfläche im Client-Server Betrieb. Die Entwicklung des Internets und des Intranets stellten neue Herausforderungen an Applikationen, die zentral auf einem Application Server zur Verfügung gestellt werden. Bereits mit der Version 4.5 liefen Applikationen mit Oracle Forms im Web parallel zum Client-Server Betrieb. Seit der Version 9 laufen Oracle Forms Applikationen nur noch auf einem Application Server ab. Die aktuelle Version 11g von Oracle Forms nutzt als Ablaufumgebung den Oracle WebLogic Server.

Die Web Versionen von Oracle Forms bieten Funktionalitäten für eine Integration von Forms Applikationen mit anderen Systemen, Anwendungen oder Prozessen, die innerhalb oder außerhalb des eigenen Unternehmens laufen. Viele Unternehmen fassen heute unterschiedliche Aktivitäten und Dienste in Prozessen zusammen.
Die Abbildung dieser Prozesse erfolgt bei Oracle in der SOA Suite über BPEL (Business Process Execution Language) oder in der BPM Suite (Business Process Management) über BPMN (Business Process Modeling and Noatation) Prozesse. Neben Aufrufen von Services, z.B. für eine Schufa Abfrage, gibt es immer auch Tätigkeiten und Aufgaben, die von einzelnen Personen oder Gruppen erledigt werden müssen. Beide Suites nutzen dafür Human Tasks und Workflow Services, um Aufgaben an diese Personen zu verteilen.

Das Beispiel

In diesem Vortrag wird beschrieben, wie Oracle Forms Anwendungen auf Prozesse in der BPM Suite 11g zugreifen. Als Grundlage dient ein einfacher Prozess, mit dem ein Angestellter einen Urlaubsantrag stellt. Im Verlauf des Prozesses werden Services aufgerufen, die überprüfen, ob der Antragsteller noch genügend Urlaub hat und wer der Vorgesetzte ist. Diese Services laufen im Hintergrund ab.
Der Antrag wird in verschiedenen Human Tasks bearbeitet, d.h. es sind bestimmte Aufgaben von beteiligten Personen am Prozess zu erledigen. Ein Vorgesetzter muss den Urlaub z.B. genehmigen oder ablehnen.

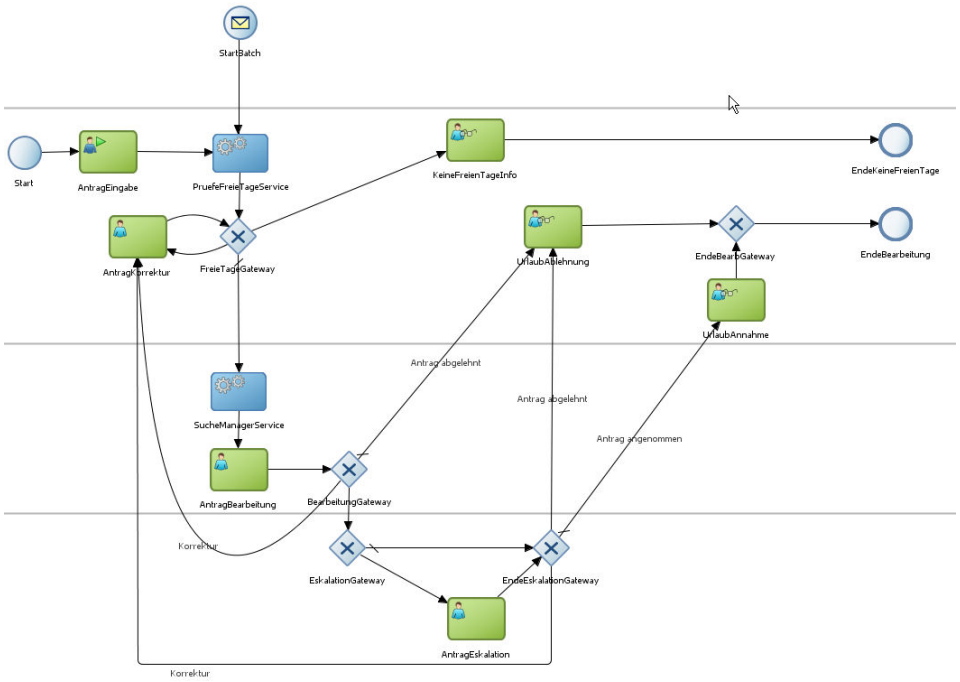

Abb. 1: Beispielprozess Urlaubsantrag

Der Prozess wird entweder im Hintergrund als Service (StartBatch) gestartet oder durch die Eingabe der Daten in eine Maske (AntragEingabe). Wenn der Prozess aus Forms heraus gestartet wird, dann wird er über die Variante im Hintergrund gestartet.
Im Prozess werden Aufgaben verteilt, z.B. AntragBearbeitung oder AntragEskalation. Die Daten für diese Aufgaben werden in Oracle Forms in Data Blocks angezeigt, die nicht auf Datenbank Tabellen basieren. Aus Oracle Forms heraus werden die Aufgaben bearbeitet und geschlossen, so dass der Prozess weiterlaufen kann.
In den folgenden Abschnitten wird kurz erläutert, welche Technologien zur Definition der Prozesse und der Zugriffe aus Forms auf die Prozessinformationen benutzt werden.

Oracle BPM Suite 11g

Mit der BPM Suite 11g (http://www.oracle.com/us/technologies/bpm/index.htm) werden Prozesse auf Basis von BPMN entwickelt. Die BPMN ist eine Notation, die von der Object Management Group (OMG; http://www.bpmn.org) standardisiert wird. An der Entwicklung dieses Standards sind alle großen Softwarehersteller wie IBM, Microsoft, SAP, Oracle etc. beteiligt. Mit dieser grafischen Beschreibungssprache werden sehr flexible und dynamische Prozessabläufe definiert. Deshalb ist sie sehr gut geeignet, um Prozesse abzubilden, in denen viele Aufgaben von Personen erledigt werden müssen und Verzweigungen und Rücksprünge innerhalb der Prozesse erforderlich sind. Die Notation ist leicht verständlich und wird auch von Fachanwendern genutzt, um Prozessabläufe festzulegen. Mit der BPMN gibt es somit eine Basis für die Kommunikation zwischen Fachabteilung und den IT Abteilungen.
Die Oracle BPM Suite 11g enthält die Tools für die Entwicklung und die Laufzeitumgebung für organisations- und systemübergreifende Prozesse. Im BPM Studio im Oracle JDeveloper (IT Abteilung) und im Process Composer (Fachabteilung) werden die Prozesse entwickelt und dann interaktiv oder im Batch in die Laufzeitumgebung verteilt. Die Einbindung von Oracle Business Rules

erlaubt es, komplexe Regeln einfach zu definieren. Das fachliche Monitoring der Prozesse wird mit Oracle Business Activity Monitoring durchgeführt.

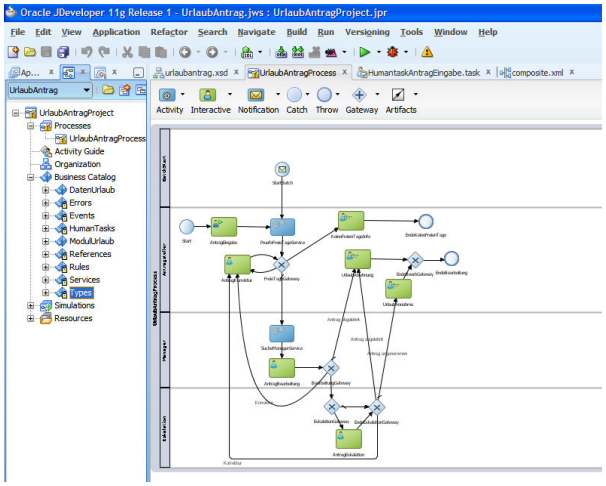

Abb. 2: BPM Studio im JDeveloper

Wie bereits erwähnt werden in vielen Prozessen Aufgaben von Personen bearbeitet werden. Die Workspace Applikation der BPM Suite 11g zeigt die Aufgaben an, die einer Person direkt oder über die Zugehörigkeit zu einer Gruppe zugewiesen werden. Die Workspace Applikation erhält diese Informationen aus dem Metadaten Repository der BPM Suite.

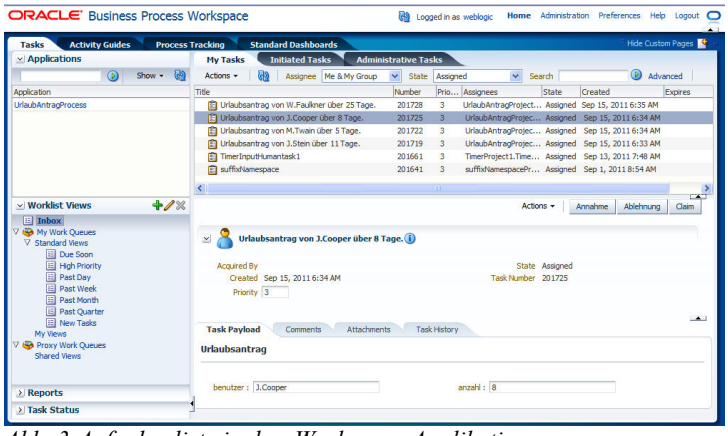

Abb. 3:Aufgabenliste in der Workspace Applikation

Über ein mitgeliefertes Java API werden Daten aus dem Repository ausgelesen und in der Workspace Applikation dargestellt. Dieses API bietet die Möglichkeit, die Aufgabenliste in bestehende Applikationen zu integrieren oder eigene Aufgabenlisten mit anderen Technologien zu entwickeln. Applikationen, die mit Oracle ADF (Application Development Framework), .NET oder dem Spring Framework entwickelt werden, kommunizieren über das Java API mit der Prozessengine.

Das Java API ist auch die Schnittstelle, über die Oracle Forms mit der BPM Suite integriert wird. Dazu müssen Java Klassen erstellt werden, die das Java API nutzen.

Das BPM Java API

Informationen über Prozesse und Aufgaben werden über das BPM API ausgelesen. Das API versetzt Anwendungen in die Lage, mit den Workflow Services über Local und Remote EJBs, SOAP und HTTP zu kommunizieren. Für die Kommunikation von Forms mit den Prozessen in der BPM Suite werden Java Klassen benötigt, die die Einzelheiten über die Aufgaben aus dem Metadaten Repository der BPM Suite auslesen und die Aufgaben bearbeiten, d.h. es wird ein Befehl an den Prozess geschickt, um eine Aufgabe mit einem bestimmten Ergebnis zu beenden. Ein Ergebnis ist z.B. Annahme oder Ablehnung des Antrages.
Für die Integration mit Forms werden folgende Aufgaben in den Klassen implementiert :

- Authentifizierung
- Abfrage der Aufgaben
- Update der Aufgaben

Für jede dieser Aufgaben stellt das Java API die entsprechenden Klassen, Methoden, Interfaces zur Verfügung.

Als erstes erfolgt die Authentifizierung. Diese erfolgt, indem man einen Handle zum Interface **IWorklistServiceClient** von der Klasse **WorkflowServiceClientFactory** erzeugt. Danach wird ein Handle zum Interface **ItaskQueryService** von **IworklistServiceClient** erzeugt.
Zur Authentifizierung werden Benutzername und Kennwort an die Methode authenticate aus **ItaskQueryService** übergeben und es wird ein Handle zu **IworkflowContext** erzeugt .
Vor der Erstellung der Handle werden noch die Properties für die Verbindung zu einem Application Server gesetzt. Alternativ können diese Informationen in der Datei wf_client_config.xml hinterlegt werden.
In diesem Fall wird vor dem Erstellen des Handles auf den Context für die Properties eine Map erstellt, in der es verschiedene Kombinationen von Property und Wert gibt :

```
Map<IWorkflowServiceClientConstants.CONNECTION_PROPERTY, java.lang.String>
  properties = new
               HashMap<IWorkflowServiceClientConstants.CONNECTION_PROPERTY,
               java.lang.String>();
properties.put(IWorkflowServiceClientConstants.CONNECTION_PROPERTY.EJB_PROV
IDER_URL,"t3://myhost:8001");
properties.put(IWorkflowServiceClientConstants.CONNECTION_PROPERTY.EJB_SECU
RITY_CREDENTIALS,"welcome1");
properties.put(IWorkflowServiceClientConstants.CONNECTION_PROPERTY.EJB_SECU
RITY_PRINCIPAL,"weblogic");
properties.put(IWorkflowServiceClientConstants.CONNECTION_PROPERTY.EJB_INIT
IAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFactory");
IworkflowServiceClient client =
WorkflowServiceClientFactory.getWorkflowServiceClient(WorkflowServiceClient
Factory.REMOTE_CLIENT, properties, null);

ITaskQueryService taskQuerySvc = client.getTaskQueryService();

IWorkflowContext ctx = taskQuerySvc.authenticate("jstein",
                                    "welcome1".toCharArray(),"jazn.com");
```

Nach dem Erstellen der Verbindung werden mit dem API Informationen über die Prozesse abgefragt, um den Input für die Forms Anwendung zu liefern.

Die drei wichtigsten Klassen für die Definition der Zugriffe sind die folgenden :

- oracle.bpel.services.workflow.query.ejb.TaskQueryServiceBean und das Interface oracle.bpel.services.workflow.query.ItaskQueryService
- oracle.bpel.services.workflow.task.ejb.TaskServiceBean und das Interface oracle.bpel.services.workflow.task.ItaskService
- oracle.bpel.services.workflow.services.task.model.Task

Die Methode queryTasks() in ITaskQueryService liefert eine Liste (List) mit den Aufgaben :

```
List ITaskQueryService.queryTasks(
  IWorkflowContext ctx,
  java.util.List displayColumns,
  java.util.List optionalInformation,
  ITaskQueryService.AssignmentFilter assignmentFilter,
  java.lang.String keywords,
  Predicate predicate,
  Ordering ordering,
  int startRow,
  int endRow)
```

- Der erste Parameter **ctx** ist der Context, der bei der Authentifizierung erzeugt wird.
- Im zweiten Parameter **displayColumns** wird die Liste der Spalten übergeben, die in der Antwort zurückgegeben werden sollen, z.B. der Titel (TITLE) oder der Status (STATE) der Aufgabe. Die Spaltennamen sind in der Java Dokumentation des **ITaskQueryService** Interface dokumentiert.
- Der Parameter **optionalInformation** wird benötigt um zusätzliche Informationen z.B. über den Payload oder Attachments zu erhalten.
- Der **assignmentFilter** gibt an, welche Aufgaben zurückgeliefert werden sollen. Grundlage ist die Zuweisung einer Aufgabe zu einer Person oder Gruppe. Bei **ITaskQueryService.AssignmentFilter** werden die Konstanten definiert, die benutzt werden können. In diesem Vortrag wird **ITaskQueryService.AssignmentFilter.MY_AND_GROUP** verwendet, d.h. alle Tasks, die ausschliesslich an eine Person oder Gruppe, in der diese Person als Mitglied eingetragen ist, sollen angezeigt werden.
- **keywords** wird zur Einschränkung der Suche benutzt.
- Mit **predicate** wird ein Filter auf weitere Merkmale der vorhandenen Aufgaben erzeugt, in diesem Beispiel für den Status einer Task. In **IworkflowConstants** werden die Konstanten deren Namen mit **TASK_STATE_** beginnen dafür definiert. Als Default wird in diesem Beispiel **IWorkflowConstants.TASK_STATE_ASSIGNED** verwendet.

Vor einer Abfrage werden z.B. die folgenden Spalten für die Rückgabe (**displayColumns**) festgelegt :

```
//Set up list of columns to query
 List queryColumns = new ArrayList();
 queryColumns.add("TASKID");
 queryColumns.add("TASKNUMBER");
 queryColumns.add("TITLE");
 queryColumns.add("OUTCOME");
```

Die Abfrage selber liefert eine Liste zurück :

```
//Query a list of tasks assigned to jstein
 List tasks = querySvc.queryTasks(ctx,
              queryColumns,
              null, //Do not query additional info
              ITaskQueryService.AssignmentFilter.MY_AND_GROUP,
```

```
            null, //No keywords
            null, //No custom predicate
            null, //No special ordering
            0,    //Do not page the query result
            0);
```

Mit der Methode **getTaskDetailsByNumber()** in **ItaskQueryService** erhält man ein Task Objekt mit den Details für die Task, z.B. Attachments, den Payload, Owner etc.. Als Input wird wieder der Context benötigt und die Tasknummer (**taskNumber**). Eine ähnliche Methode arbeitet mit der **taskId**.

Mit den bisher erläuterten Methoden werden Informationen über die Tasks bzw. Aufgaben abgefragt und stehen somit zur Anzeige in Anwendungen zur Verfügung. Jede Aufgabe muss bearbeitet und mit einem bestimmten Ergebnis geschlossen werden, damit der Prozess weiterlaufen kann. Eine Aufgabe wird beispielsweise mit Ja oder Nein geschlossen. Dieser Wert muss an den Prozess zurückgegeben werden.
Das Interface **ITaskService** enthält die Methoden, um eine Task zu bearbeiten :

- **updateTaskOutcome()** erlaubt eine Aktion auf eine Task oder die Ausführung einer Task und das Setzen des Ergebnisses (Outcome). Das Ergebnis einer Task wird in der Definition der Task festgelegt, z.B. Ja/Nein oder Annahme/Ablehnung.
- **escalateTask()** eskaliert eine Task zum Manager.
- **acquireTask()** reserviert (claim) eine Task zur ausschließlichen Bearbeitung für den Benutzer.
- **withdrawTask()** gibt eine Task wieder frei zur Bearbeitung durch andere Personen (Gegenteil von acquireTask()).
- **suspendTask()** legt eine Task quasi schlafen, nimmt sie aus der aktiven Bearbeitung heraus und stoppt die Timer für Zeitüberschreitungen und Nachrichten.
- **resumeTask()** setzt eine Task wieder aktiv (Gegenteil von suspendTask()).
- **purgeTask()** löscht eine Task aus dem System.
- **errorTask()** setzt den Status einer Task auf Error.

Das folgende Beispiel zeigt eine Schleife, bei der alle offenen Tasks mit dem Ergebnis APPROVED geschlossen werden :

```
//Get the task service
 ITaskService taskSvc = wfSvcClient.getTaskService();
 //Loop over the tasks, outputting task information, and approving any
 //tasks whose outcome has not been set...
 for(int i = 0 ; i < tasks.size() ; i ++)
 {
  Task task = (Task)tasks.get(i);
  String taskId = task.getSystemAttributes().getTaskId();
  String outcome = task.getSystemAttributes().getOutcome();
  if(outcome == null)
  {
   outcome = "APPROVED";
   taskSvc.updateTaskOutcome(ctx,taskId,outcome);
  }
}
```

Nach dem Erstellen der Java Klassen mit den Zugriffen auf die BPM Prozesse werden diese Klassen jetzt in eine Oracle Forms Anwendung integriert.

Einbindung externer Services in Oracle Forms mit dem Java Importer

Die Integration von Java ermöglicht es, aus Oracle Forms heraus externe Java Klassen und Methoden aufzurufen, die in einem Java Container auf einem Application Server laufen. Diese Klassen enthalten reinen Java Code oder sie dienen als Schnittstelle zu Webservices. Über den Java Importer werden die Java Klassen aufgerufen. Dazu werden im Oracle Forms Builder PL/SQL Wrapper Prozeduren generiert, die Methoden in den Klassen aufrufen.
Vor dem Aufrufen des Forms Builders muss der FORMS_BUILDER_CLASSPATH gesetzt werden. Dieser Pfad muss vor dem Start des Forms Builders die Klassen oder Verzeichniss mit Klassen enthalten, die importiert werden sollen.
Im Forms Builder wird über das Menu Program→Import Java Classes der Dialog zum Importieren der Klassen gestartet. Alle Packages und Klassen aus dem FORMS_BUILDER_CLASSPATH werden angezeigt. Aus den Klassen wird diejenige ausgewählt, die importiert werden soll, z.B. GetTaskList.

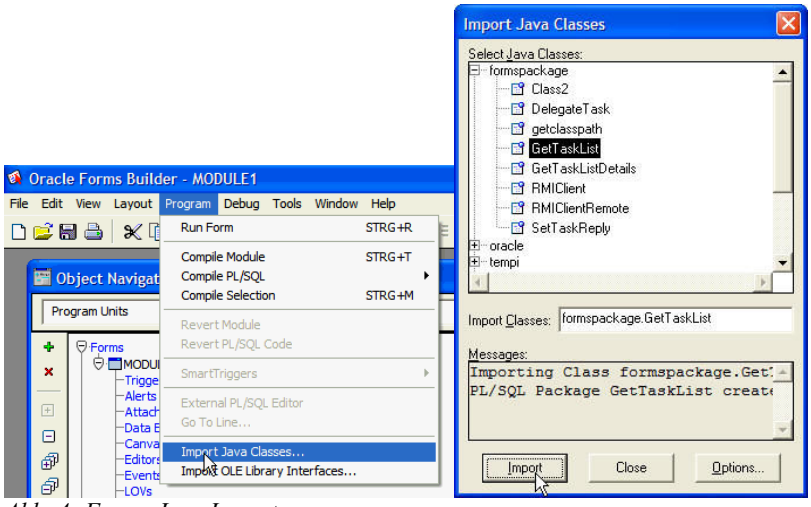

Abb. 4: Forms Java Importer

Durch das Drücken des Buttons Import wird der Import der Klasse gestartet. Beim Import werden PL/SQL Packages mit Funktionen und Prozeduren generiert, die zur Laufzeit die Java Klassen aufrufen.
Das Package bekommt dabei den gleichen Namen wie die hier importierte Java Klasse GetTaskList :

```
PACKAGE GetTaskList /* formspackage.GetTaskList */ IS
--
-- Constructor for signature ()V
FUNCTION new RETURN ORA_JAVA.JOBJECT;

-- Method: main ([Ljava/lang/String;)V
PROCEDURE main(
  a0    ORA_JAVA.JARRAY);

-- Method: AusgabeTasks
-- (Ljava/lang/String;Ljava/lang/String;)[Ljava/lang/String;
FUNCTION AusgabeTasks(
  a0    VARCHAR2,
  a1    VARCHAR2) RETURN ORA_JAVA.JARRAY;
END;
```

Die generierten Prozeduren und Funktionen werden jetzt in Forms Trigger und Prozeduren/Funktionen eingebunden. Der Forms Entwickler arbeitet dabei weiterhin mit PL/SQL, da er nur die generierten PL/SQL Prozeduren und Funktionen einbinden muss. Für eine ähnliche Anzeige der Arbeitsliste wie in der Workspace Anwendung wird ein Data Block in Forms angelegt, der nicht auf einer Datenbank Tabelle basiert. Ein Trigger oder eine Prozedur ruft die Methode AusgabeTasks aus der importierten Java Klasse GetTaskList auf, um die Informationen über die Aufgaben für eine bestimmte Person zu erhalten.

Die Daten werden dafür in einem Java Array in Forms zwischengespeichert :

```
-- New Java Array
 arr := ORA_JAVA.NEW_STRING_ARRAY(100);

 -- Call GetTaskList.AusgabeTasks
   arr := gettasklist.AusgabeTasks(myglobals.antragsteller,'welcome');
```

Nach dem Befüllen des Java Arrays werden die Daten in einer Schleife in den Forms Data Block eingefügt.

Beim Design des Data Blocks wird festgelegt, welche Informationen angezeigt werden sollen. Für einen Überblick über die Aufgaben reicht die Anzeige des Titels der Aufgabe.

Die Details einer Aufgabe werden in einem weiteren Data Block angezeigt. Über Doppelklick auf eine Aufgabe oder über einen Button wird der Detail Data Block aufgerufen. Auch dieser Block basiert nicht auf einer Datenbank Tabelle, sondern er wird ebenfalls über eine importierte Java Klasse und mit PL/SQL Funktionalität befüllt.
Um eine Aufgabe zu schließen, wird über einen WHEN-BUTTON-PRESSED Trigger eine Java Klasse aufgerufen, die die Methode **updateTaskOutcome()** mit dem Ergebnis Annahme oder Ablehnung startet.

Zusammenfassung

Die Integration von Oracle Forms mit Prozessen in der BPM Suite 11g erfolgt über die Einbindung von Java Klassen in Oracle Forms. In diesen Java Klassen sind die lesenden und schreibenden Methoden auf die Prozessinformationen mit Hilfe des BPM Java APIs hinterlegt. Mit dem Java Importer werden diese Klassen in Forms Module importiert und PL/SQL Wrapper Prozeduren generiert. Diese generierten Prozeduren bindet der Forms Entwickler in sein Forms Modul ein, um eine Aufgabenliste im Forms Modul anzuzeigen und die Aufgaben zu bearbeiten.

Kontaktadresse:

Gert Schüßler
Oracle Deutschland B.V. & Co. KG
Kühnehöfe 5
D-22761 Hamburg

Telefon: +49 (0) 40-89091 190
Fax: +49 (0) 40-89091 250
E-Mail gert.schuessler@oracle.com
Internet: www.oracle.de