TopLink in de praktijk

Persistency software eenvoudig in gebruik

Bij de ontwikkeling van iedere applicatie in een J2EE omgeving waarbij persistentie van data noodzakelijk is, is de koppeling tussen de objecten uit de Java wereld en de relationele database een belangrijk issue. TopLink van Oracle is speciaal voor dit onderdeel van de ontwikkeling van een applicatie in de markt gezet. Dit artikel beschrijft het gebruik van TopLink in een project dat uitgevoerd is door Cap Gemini Ernst & Young.

De applicatie is ontwikkeld in een J2EE omgeving met gebruikmaking van JDeveloper. Bij dit project is TopLink 9.0.3 gebruikt met de Oracle 9iAS applicatie server en een Oracle 9.0.2 database. Het product TopLink bestaat uit twee onderdelen:

1. De Mapping Workbench. In deze grafische workbench vindt de daadwerkelijke koppeling van de Java objecten met de relationele database plaats. Het aanmaken van deze koppeling kan volledig onafhankelijk van het ontwikkelen van de Java code gebeuren zodra de Java objecten aangemaakt zijn.
2. Naast de workbench is Java code meegeleverd in de vorm van een jar file. Met behulp van deze code kunnen methodes gebouwd worden die de communicatie met de database verzorgen; er worden geen conventionele SQL-statements geschreven, alle statements worden ontwikkeld met Java code.

In dit artikel worden beide onderdelen uitvoerig besproken aan de hand van voorbeelden uit het project.

Werken met de mapping workbench

Bij de inrichting van de workbench maken we allereerst een nieuw TopLink project aan. Een TopLink project bestaat uit meerdere descriptors (voor elk Java object één); een descriptor is de beschrijving van de koppeling van een object en bestaat uit meerdere mappings (voor elk attribuut één). Na het aanmaken van het project in de workbench, wordt een verwijzing naar de database gemaakt, waarbij de driver, het pad naar de database, de gebruikersnaam en het wachtwoord ingevoerd worden. Na deze stap kunnen de tabellen en views uit de betreffende database in het TopLink project geladen worden. Vervolgens moet het classpath van de applicatie gedefinieerd worden, zodat de te koppelen Java objecten in het project geladen kunnen worden. Na deze stappen is de inrichting van het nieuwe project in de workbench voltooid en kan de mapping gestart worden.

Als voorbeeld van een Java object nemen we het object Persoon. De Java bean Persoon is abstract en heeft twee subclasses, te weten PersoonIntern en PersoonExtern, waarbij het onderscheid wordt gemaakt door de manier waarop een persoon aan het bedrijf verbonden is. Een class wordt gedefinieerd als abstract indien hier geen instantie van gemaakt mag worden; alleen zijn subclasses mogen geïnstantieerd worden.

Aanmaken van mappings

De eerste mapping vindt plaats op classniveau; na het selecteren van de class Persoon, kan de bijbehorende view geselecteerd worden uit de views die in het project geladen zijn. In ons voorbeeld is dit de view V_PERSONEN. Nu deze mapping is gecreëerd, worden de mappings voor de afzonderlijke attributen van Persoon gemaakt. Er zijn verschillende soorten mappings voor de velden. De meest eenvoudige is de direct field mapping, waarbij een attribuut rechtstreeks aan een veld uit de database wordt gekoppeld. Hierbij kan gekozen worden uit de lijst van alle beschikbare velden uit de bijbehorende view. In onze applicatie geldt deze mapping voor bijvoorbeeld de achternaam van een persoon.

In TopLink speelt caching een belangrijke rol; gegevens worden zoveel mogelijk eenmalig ingelezen en bij opvraging uit de cache gehaald

Het attribuut is static gemaakt zodat het als een attribuut van de class zelf gedefinieerd is en niet van een instantie van de class

De meer ingewikkelde mappings zijn de OneToOne en OneToMany mappings, nodig indien het te mappen attribuut zelf een object respectievelijk een vector van objecten is. Om deze typen van een mapping aan te kunnen maken, is het noodzakelijk dat de mapping van het doelobject zelf al aangemaakt is. Vervolgens moet aangegeven worden op welke manier de twee objecten aan elkaar gerelateerd zijn. Hiervoor kan een bestaande foreign key relatie gekozen worden of een nieuwe relatie aangemaakt worden in de Workbench. De OneToMany mapping is in het project toegepast om de gevolgde cursussen van een persoon vast te leggen in de vorm van een vector van cursusobjecten.

Een andere toegepaste mapping is de ObjectTypeMapping. Hierbij kan per mogelijke waarde van een veld in de database aangegeven worden wat de bijbehorende waarde in het object moet zijn. Een mooi voorbeeld hiervan is het attribuut geslacht; in de database heeft dit veld de waarde 'M' of 'V'. Aangezien in de applicatie het geslacht voluit getoond moet worden, wordt het attribuut geslacht als een ObjectTypeMapping mapping aangemaakt met de waarden 'Man' of 'Vrouw'.

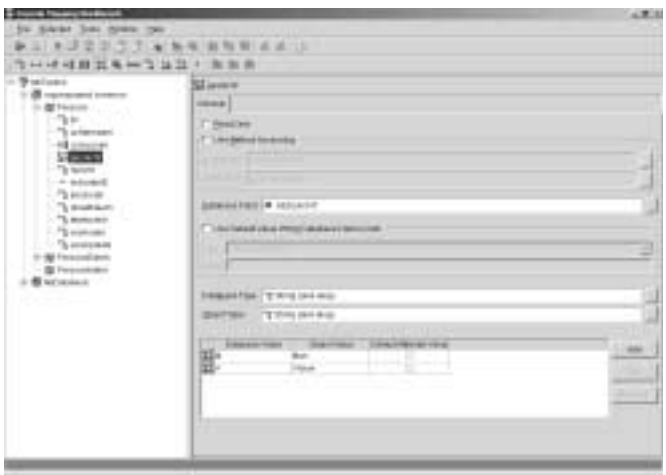
Indien een object gevuld wordt met velden uit meerdere views, wordt bij dit object de property 'MultiTable' aangezet. Dit voegt een extra tabblad aan de workbench toe voor dit object, waarop aangegeven wordt om welke extra tabel c.q. view het gaat en welke foreign key relatie er gebruikt wordt om de tabellen c.q. views te linken. Indien nu een nieuwe mapping voor een attribuut wordt aangemaakt, wordt in de lijst van te kiezen velden de naam van de bijbehorende tabel c.q. view weergegeven.

Overerving

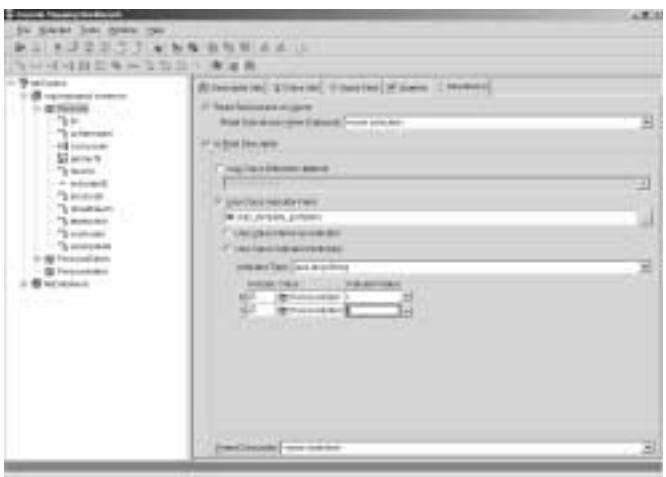
Zoals eerder aangegeven is de class Persoon een abstracte class en kan dus niet geïnstantieerd worden, alleen de twee subclasses kunnen hiervoor gebruikt worden. In TopLink is het zeer eenvoudig om de relatie te leggen tussen een superclass en zijn subclasses, namelijk door middel van een zogenaamde class indicator. In ons project is in de view V_PERSONEN het veld IND_INTERN_EXTERN gedefinieerd welke de waarde I (intern) of E (extern) kan krijgen. Dit veld wordt nu bij de class Persoon, de superclass, gebruikt als de class indicator, waarbij de relatie tussen de waarde van het veld en de bijbehorende subclass wordt gelegd. Op deze manier is het niet nodig om in de code aan te geven of de persoon als een PersoonIntern of PersoonExtern aangemaakt moet worden, door het gebruik van een class indicator wordt dit automatisch geregeld. In de bespreking van Java code voor het ophalen van objecten komen we hier nog op terug.

Performance

Performance is voor elke applicatie van cruciaal belang. In TopLink speelt caching een belangrijke rol, er wordt zoveel mogelijk geprobeerd om gegevens eenmalig in te lezen en vervolgens bij opvraging uit de cache te halen. Hier moet echter voorzichtig mee worden omgegaan, de juistheid ervan hangt af van de volatiliteit van het object. Indien de gegevens van een object statisch zijn en dus niet veranderen, moet de



Afbeelding 1. Voorbeeld van een ObjectTypeMapping



Afbeelding 2. Overerving bij de superclass Persoon

FullIdentityMap als cache mechanisme gekozen worden. Objecten die vaak van waarde veranderen en dus niet eenmalig in de cache ingelezen mogen worden, gebruiken WeakIdentityMap als mechanisme. Hierbij wordt het object, nadat het gebruikt is door de applicatie, aangeboden voor garbage-collection; dit houdt in dat de Java Virtual Machine het object uit de cache verwijdert. De tussenvorm heet de SoftCacheWeakIdentityMap.

Het ophalen van een object uit de database kan, indien er één of meerdere OneToOne of OneToMany mappings in dit object aanwezig zijn, een kostbare operatie zijn en de performance negatief beïnvloeden, aangezien voor elke mapping een aparte query wordt uitgevoerd. Indien alle informatie binnen deze mappings noodzakelijk is, is hier geen verbetering in aan te brengen. Echter, indien slechts een deel van de informatie van het object nodig is, moet er voor gekozen worden om Indirection te gebruiken voor deze mappings. Dit zorgt ervoor dat de gerelateerde objecten pas worden opgehaald op het moment dat hier daadwerkelijk om wordt gevraagd. Voorbeeld: de gevolgde cursussen van een persoon zijn niet noodzakelijk om een overzicht van personen na een zoekactie te tonen. Voor de vector met cursusobjecten is derhalve Indirection gebruikt.

Overgang van workbench naar Java code

Nadat voor alle objecten een mapping is aangemaakt, wordt een export file aangemaakt in de vorm van een XML file of een Java class. Het gebruik van een XML file heeft als voordeel dat in het geval van een aanpassing de Java code niet opnieuw gecompileerd hoeft te worden. De Java class heeft als voordeel dat de Java code na compilatie direct beschikbaar is en niet, zoals een XML file, ingelezen moet worden. In ons project hebben we gekozen voor het exporten van een Java class.

Gebruik van TopLink in Java code

Nadat de Java class met alle mappings is geëxporteerd, kunnen de methodes aangemaakt worden waarmee objecten uit de database gehaald worden. Hierbij worden TopLink sessies gebruikt.

Het gebruik van TopLink sessies

In onze applicatie is voor elk object een Data Access Object (DAO) ontwikkeld. Voor het object Persoon hebben we de class Personae aangemaakt; deze class bevat alle methodes die operaties op de database uitvoeren voor het Persoon object.

Voor de communicatie met de database is een zogenaamde ServerSession nodig; deze vormt de laag tussen een ClientSession in de Java applicatie en de database. Voor elke database operatie is zo'n ClientSession nodig; deze wordt aangevraagd bij de ServerSession. Let op: de genoemde sessions zijn instanties van TopLink classes en niet gerelateerd aan de HttpSession (een instantie van de class HttpSession wordt in internet applicaties gebruikt om data tussen de browser en de applicatie heen en weer te zenden). Voor de configuratie van de ServerSession is een XML file aanwezig (sessions.xml); hierin worden configuratie gegevens als username, password en de locatie van de database gedefinieerd, of indien er gebruikt van wordt gemaakt, een data source. De configuratie wordt buiten de Java applicatie gehouden; bij deployment van de applicatie op verschillende omgevingen met ieder een eigen database, is dus slechts een aanpassing in de sessions.xml nodig.

In onze applicatie beschikken we over een class ProjectServerSession met daarin een static attribuut myServerSession van het type ServerSession. Het attribuut is static gemaakt zodat het als een attribuut van de class zelf gedefinieerd is en niet van een instantie van de class. De ServerSession wordt eenmalig geïnstantieerd en is vervolgens beschikbaar om aangeroepen te worden. Dit is een gebruikelijke werkwijze voor objecten die als een soort server binnen de applicatie dienen. De methode getClientSession() wordt gebruikt voor de aanvraag van een ClientSession bij myServerSession. Onderstaand is een vereenvoudigde versie van onze class ProjectServerSession weergegeven:

```
public class ProjectServerSession
{
   public static ServerSession myServerSession;

   public static ClientSession getClientSession()
   {
                                return
myServerSession.acquireClientSession();
   }

  ....
}
```

Een ServerSession geeft dus op verzoek een ClientSession uit, waarop een database operatie uitgevoerd kan worden. In onze voorbeeldcode zal het gebruik van de ClientSession verduidelijkt worden. Een kenmerk van het gebruik van TopLink is dat er geen SQL-statement geschreven worden, maar dat er Query objecten gebruikt worden die deze statements genereren. Dit kan op twee manieren gebeuren, namelijk via de ExpressionBuilder of het QueryByExample object.

Gebruik van de ExpressionBuilder

Als voorbeeld voor het gebruik van de ExpressionBuilder nemen we de methode findPersoonByID() uit de class PersoonDAO welke aan de hand van een unieke identifier een persoon ophaalt uit de database. Na het aanmaken van een client session, wordt een Expression aangemaakt waarin wordt gedefinieerd waaraan het op te halen object moet voldoen.

```
ExpressionBuilder bldr = new ExpressionBuilder();
Expression exp = bldr.get("persoonId").equal(persoonID);
```

Als argument van de get() methode van de ExpressionBuilder wordt de naam van het attribuut in de Java class gebruikt. Hier is dus geen kennis over de database naamgeving nodig. Vervolgens wordt een Query object aangemaakt; aangezien we hier slechts één object ophalen, wordt hiervoor de subclass ReadObjectQuery gekozen. We gebruiken hiervoor de constructor met twee argumenten, te weten het type van de te retourneren class en de eerder opgebouwde expressie. Merk op dat we hier als class type de superclass Persoon gebruiken, aangezien we hier niet weten of we een intern of extern persoon ophalen. Doordat we gebruik maken van de eerder beschreven class indicator, bepaalt TopLink zelf welke subclass teruggegeven wordt. Tenslotte voeren we de query uit door op de client session de executeQuery() methode aan te roepen met als argument het ReadObjectQuery object. De volledige code van deze methode ziet er als volgt uit:

```
public Persoon find PersoonByID(String persoonID) throws MyException{

    Persoon persoon = null;
    try
    {
      ClientSession cSession =
ProjectServerSession.getClientSession();
      ExpressionBuilder bldr = new ExpressionBuilder();
      Expression exp = bldr.get("persoon Id").equal(persoonID);
      ReadObjectQuery pQuery = new ReadObjectQuery(Persoon.class,exp);
      persoon = (Persoon)cSession.executeQuery(pQuery);
    }
    catch (TopLinkException tle)
    {
    MyException me = new MyException(MyException.ERROR_DATABASE_PROBLEM);
      throw me;
    }
    return persoon;
}
```

Gebruik van QueryByExample

In onze applicatie hebben we het QueryByExample principe toegepast in de methode selectPersonen(), waarin aan de hand van een aantal zoekcriteria een vector van personen wordt opgehaald. Een vector is een verzameling van meerdere objecten. In deze methode wordt een Persoon object aangemaakt als een 'voorbeeld object', vandaar de naam QueryByExample; dit object wordt vervolgens gevuld met de waarden waarop gezocht wordt. In onderstaande code wordt het attribuut achternaam gevuld met de waarde uit het object pZoekCriteria, dat als argument aan de methode is meegegeven. Indien dit attribuut de waarde 'null' bevat, wordt dit door TopLink zelf afgehandeld; het is dus niet nodig om hier zelf op te controleren. Omdat in de methode naar meerdere objecten wordt gezocht, wordt als Query object een ReadAllQuery object aangemaakt. Aan dit object wordt het voorbeeld object toegevoegd via de methode setExampleObject. Dit resulteert in onderstaande code. Voor de eenvoud van het voorbeeld is de vulling van het voorbeeld object beperkt tot één attribuut:

```
public Vector selectPersonen(PZoekCriteria pZoekCriteria) throws
MyException{

    ClientSession cSession = ProjectServerSession.getClientSession();
    Vector pVector = null;
    try
    {
      if (pZoekCriteria != null) {
        Persoon qPersoon = new Persoon();
        qPersoon.setAchternaam(pZoekCriteria.getAchternaam());

        ReadAllQuery pQuery = new ReadAllQuery(Persoon.class);
        pQuery.setExampleObject(qPersoon);
        pVector = (Vector)cSession.executeQuery(pQuery);
      }
    }
    catch (TopLinkException tle)
    {
      MyException me = new MyException(MyException.ERROR_DATABASE_PRO-
BLEM);
      throw me;
    }

    return pVector;
  }
```

Conclusie

Het gebruik van TopLink voor de koppeling van Java objecten naar een relationele database is zeer aan te bevelen. De grafische Workbench voor het aanmaken van deze koppeling is snel te installeren en eenvoudig in het gebruik. Gemaakte fouten in de koppeling worden direct gemeld, zodat niet achteraf nog naar foutmeldingen gezocht behoeft te worden. Daarnaast kan het maken van de koppeling parallel lopen aan het ontwikkelen van de Java methoden, mits de gebruikte Java objecten aangemaakt en gecompileerd zijn. Tenslotte merken we op dat voor het werken met TopLink geen kennis van SQL nodig is, omdat hiervoor de Java code van TopLink wordt gebruikt. Het principe hierachter is echter dusdanig vergelijkbaar met het schrijven van SQL statements, dat TopLink voor zowel ontwikkelaars met een Java als een Oracle achtergrond eenvoudig te gebruiken is.

Jeroen Eversen
is werkzaam als Senior Consultant bij Cap Gemini Ernst & Young (e-mail: jeroen.eversen@cgey.nl). Bij vragen over dit artikel kan contact met de schrijver worden opgenomen.