Codegenerierung

Entwicklung von Modell-zu-Code Transformationen zur Abbildung von Modellen auf eine Webapplikationsplattform (Backend)

Plattformmodell

Für die Abbildung von 3-Schichten Modellen wird eine Einteilung des zu erstellenden Softwaresystems in ein Frontend-System und in ein Backend-System getroffen. Für die Übung ist nur die Generierung des Backend-Systems gefordert. Das Backend-System besteht aus einem Datenbankmanagementsystem für die persistente Speicherung der Objekte, Hibernate Mapping Files für das Objekt-Relationale Mapping, Java Beans für die Klassenbeschreibungen und Data Access Objects (DAO) um Objekte zu suchen, zu ändern und zu löschen. Abbildung 1 zeigt den groben Architekturaufbau.

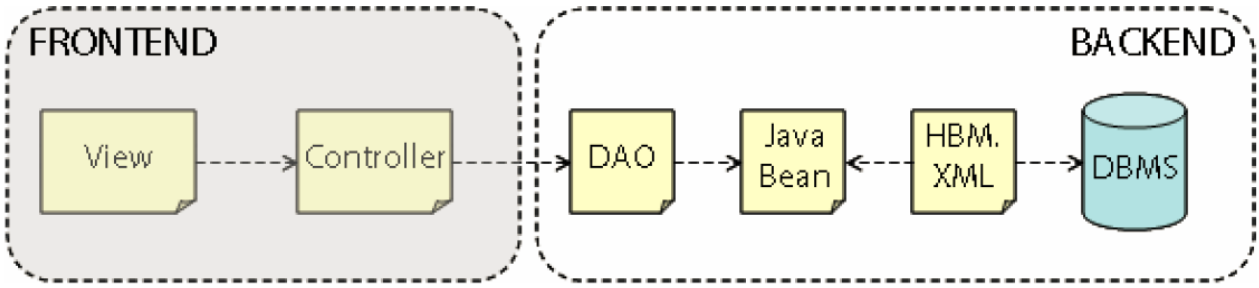


Abbildung 1 Architektur

Gegeben sind das Metamodell aus Übung 1 bzw die dazugehörige XText Grammatik (sollten möglichst ident sein).

Ziel dieser Übung ist es mit Xtend Codegenerierungsregeln zu schreiben, um die entsprechenden Files der Pakete **beans, daos, daoImpl** sowie das **hibernate.cfg.xml** File aus der Implementierung erzeugen. Für die Ausarbeitung der Codegenerierungsregeln wird das vorher erstellte XText Projekt verwendet und im File SWMLGenerator.xtend müssen die entsprechenden Regeln ergänzt werden.

Aufgabenstellung

Gesucht sind Xtend Templates, die beliebige Beispielmodelle konform zum SWML-Metamodell in implementierungsspezifische Artefakte transformieren.

Aufgabe a)

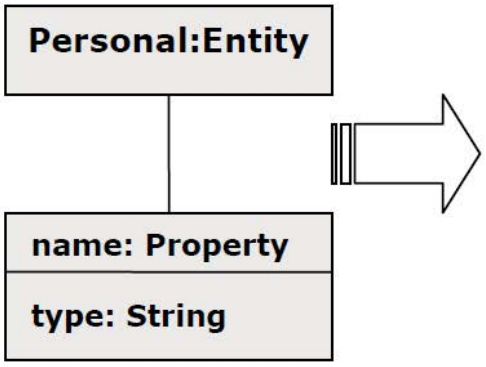
Beschreiben Sie in dem Xtend Template die im Folgenden beschriebenen Abbildungen.

1) DataAccessLayer: Für DataAccessLayer Elemente wird nur ein Hibernate Configuration File benötigt in dem für jede Entity definiert werden muss wo sich das entsprechende Hibernate Mapping File befindet. Im Hibernate Configuration File ist des Weiteren angegeben, dass das Datenbankschema automatisch erstellt wird, d.h. Sie müssen keine DDLs generieren, dieser Schritt wird von Hibernate automatisch übernommen.

2) Entity: Eine Entity wird auf eine Java Bean und ein Hibernate Mapping File abgebildet. Beide Artefakte sollen automatisch im Paket beans generiert werden.

- Entities werden als Java Beans abgebildet, wobei die Properties als Attribute mit entsprechenden Setter- und Getter-Methoden umgesetzt werden. Für mehrwertige Properties soll der Datentyp Set (definiert in java.util.Set) verwendet werden.

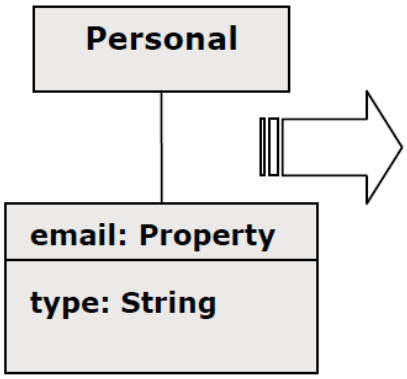
  Beispiel: Entity Personal -> Personal.java




```
package beans;

import java.util.Set;

public class Personal {
    private String name;

    public Personal() {
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

- Für jede Entity soll zusätzlich zur Java Bean Klasse ein Hibernate Mapping File (z.B.: Personal.hbm.xml) erzeugt werden. In diesem File müssen vor allem die Properties entsprechend abgebildet werden. Speziell für Properties mit gesetztem inverseProperty müssen eigene Fallunterscheidungen getroffen werden, da es sich um Rollen von Assoziationsenden handelt. Studieren Sie dazu genau die Referenzimplementierung, es reicht aber aus die Fälle des Beispielmodells abzudecken, d.h., one-to-many Beziehungen. Zusätzlich muss in dem Hibernate Mapping File ein identifizierendes Property gesetzt werden.

  Beispiel: Entity Personal -> Personal.hbm.xml




```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC …">
<hibernate-mapping package="beans">
  <class name="Personal" table="Personal">
    <id name="name" column="name_ID">
    </id>
      ...
    <property name="email" type="string"/>
  </class>
</hibernate-mapping>
```

3) DataAccessObject: Für jede Entity wird einerseits ein Interface mit vier vorgegeben Methoden benötigt (im Paket daos) und andererseits eine Implementierungsklasse (im Paket daosImpl) welche die in den Interfaces definierten Methoden implementiert.

Beispiel: DataAccessObject PersonalDAO -> daos.PersonalDAO.java und daosImpl.PersonalDAOHibernate.java


```java
package daos;

import beans.*;
import java.util.List;

public interface PersonalDAO {

    public List getPersonals();

    public Personal getPersonal
      (String id);

    public void savePersonal
      (Personal personal);

    public void removePersonal
     (Stringid);
}
```

```java
package daosImpl;

import beans.*;
import daos.*;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import java.util.List;


public class PersonalDAOHibernate implements
PersonalDAO {

    public PersonalDAOHibernate() {
    }

    public List getPersonals() {
    }

    public Personal getPersonal(Long id) {
    }

    public void removePersonal(Long id) {
    }

    public void savePersonal(Personal personal) {
    }
}
```