Curso de Java

Acesso a banco de dados através do Hibernate

Motivação

Java é uma linguagem orientada a objetos:

- Numa aplicação, o uso de objetos é um recurso importante ao programador.
- Bancos de dados relacionais são orientados à manipulação de tabelas.

- O Hibernate se propõe a manter objetos num banco de dados relacional através do ‘mapeamento objeto relacional’ (ORM). Outras características:
  - Isolamento quanto ao SGBD utilizado.
  - Linguagem de consultas orientada aobjetos: HQL.

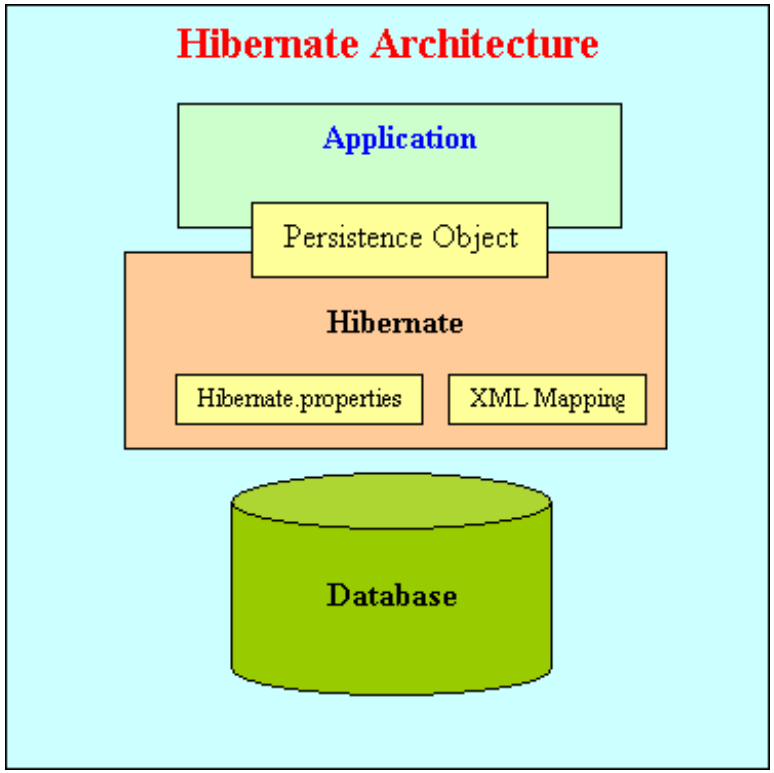
Mapeamento Objeto Relacional

Como tanto Java como SQL ('modelo relacional') não prevêem o mapeamento 'objeto-relacional' este é obtido por 'meios externos':

- Um arquivo <xml> que associa uma classe a uma tabela no SGBD e cada um dos seus campos às colunas da tabela, ou
- Através de anotações no código Java da classe.

A partir do Mapeamento Objeto Relacional, o Hibernate gera para cada classe mapeada uma classe derivada que inclui os métodos associados ao acesso ao SGBD.

Arquitetura



- Hibernate.properties: configuração do Hibernate com relação ao SGBD (url, senha, driver JDBC, etc.).
- XML mapping: associa cada classe a uma tabela no SGBD e seus campos às colunas.
- Persistence Object: classe derivada da classe mapeada da aplicação, que contém os métodos para acesso ao SGBD.

Construção de uma aplicação

Para a construir uma aplicação baseada no Hibernate deve-se seguir os seguintes passos:

- criar as tabelas no SGBD onde os objetos vão *persistir*.
- criar os objetos cujo estado será *persistido*.
- criar um XML, que relaciona as propriedades dos objetos às tabelas e seus campos às colunas.
- criar o arquivo *hibernate.properties* que contém as propriedades para que o Hibernate se conecte ao bando de dados.
- criar as classes para acesso aos objetos através do Hibernate (classes DAO - *Data Access Object*).

Criação das Tabelas no SGBD

Um exemplo:

```
CREATE TABLE aluno
(
  aluno_matricula character varying(10) NOT NULL
                  DEFAULT ''::character varying,
  aluno_nome character varying(40) NOT NULL
                  DEFAULT ''::character varying,
  aluno_rg character varying(10) NOT NULL
                  DEFAULT ''::character varying,
  CONSTRAINT aluno_pkey PRIMARY KEY (aluno_matricula)
)
WITHOUT OIDS;
ALTER TABLE aluno OWNER TO postgres;
```

Definição dos objetos persistidos

```
public class Aluno {
   String matricula;  // matrícula - chave primária
   String nome;       // nome do aluno
   String rg;         // rg do aluno

   /** Construtor padrão **/
   public Aluno() {}

   /** setters padrão **/
   public void setMatricula(String m) { matricula = m; }
   public void setNome(String n)      { nome = n;      }
   public void setRg(String r)        { rg = r;        }

   /** getters padrão **/
   public String getMatricula() { return matricula; }
   public String getNome()      { return nome;      }
   public String getRg()        { return rg;        }

}
```

XML associando tabelas às classes

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
   PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
   http://hibernate.sourceforge.net/hibernate-mapping.dtd
>
<hibernate-mapping>
  <class name="exemplo1.Aluno" table="aluno">
     <id name="matricula" column="aluno_matricula" type="string">
         <generator class="assigned"/>
     </id>
     <property name="nome" column = "aluno_nome" type="string"/>
     <property name="rg"   column="aluno_rg" type="string"/>
  </class>
</hibernate-mapping>
```

Hibernate.properties

```
## PostgreSQL

hibernate.dialect org.hibernate.dialect.PostgreSQLDialect
hibernate.connection.driver_class org.postgresql.Driver
hibernate.connection.url jdbc:postgresql://localhost/teste2
hibernate.connection.username postgres
hibernate.connection.password postgres
hibernate.query.substitutions yes 'Y', no 'N`

## MS SQL Server

#hibernate.dialect org.hibernate.dialect.SQLServerDialect

## JSQL Driver
#hibernate.connection.driver_class com.jnetdirect.jsql.JSQLDriver
#hibernate.connection.url jdbc:JSQLConnect://1E1/test
```

Classe de Acesso aos Objetos

```
public class DaoAlunos{

    private SessionFactory factory;

    public DaoAlunos() throws Exception{
        factory = new Configuration()
                        .addClass(Aluno.class)
                        .buildSessionFactory();
    }

    public void insert(Aluno aluno) throws Exception{
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();
        session.persist(aluno);
        tx.commit();
        session.close();
    }
```

Classe de Acesso aos Objetos (cont)

```
public class DaoAlunos{
   ...
   public Aluno retrieve(String pk) throws Exception{
       Session session = factory.openSession();
       Aluno aluno = (Aluno)session.get(Aluno.class, pk);
       session.flush();
       session.close();
       return aluno;
    }

    public void delete(Aluno aluno) throws Exception{
       Session session = factory.openSession();
       Transaction tx = session.beginTransaction();
       session.delete(aluno);
       tx.commit();
       session.close();
    }
}
```

Objetos e chaves estrangeiras

Na figura acima:

- Um objeto Alocação associa um Professor a um Curso.
- Ao se criar as tabelas corresponentes essas associações correspondem a chaves estrangeiras na tabela professor.
- Na aplicação baseada em Hibernate, ao se recuperar um objeto Alocacao, os objetos Curso e Professor associados são recuperados automaticamente.

Consultas

O Hibernate permite as seguintes formas de consultas:

- Linguagem de consultas Hibernate (**HQL**): linguagem voltada à consulta de objetos, similar a SQL.
- Consultas por critérios (**QBC**) e consultas por exemplos
- Consultas via **SQL** nativo do SGBD

Anotações

Anotações são diretivas colocadas no código Java para associar as classes a tabelas no SGBD, substituindo o arquivo XML. Um exemplo:

```
@Entity
@Table(name="aluno")
public class Aluno {
   @Id
   @GeneratedValue(strategy=GenerationType.ASSIGNED)
   String matricula;  // matrícula - chave primária

   @Column(name="aluno_nome")
   String nome;       // nome do aluno

   public Aluno() {}

   public void setMatricula(String m) { matricula = m; }
   public void setNome(String n)      { nome = n;      }

   public String getMatricula() { return matricula; }
   public String getNome()      { return nome;      }
}
```

Curso de Java

Acesso a banco de dados através do Hibernate

Motivação

Java é uma linguagem orientada a objetos:

- Numa aplicação, o uso de objetos é um recurso importante ao programador.
- Bancos de dados relacionais são orientados à manipulação de tabelas.

- O Hibernate se propõe a manter objetos num banco de dados relacional através do ‘mapeamento objeto relacional’ (ORM). Outras características:
  - Isolamento quanto ao SGBD utilizado.
  - Linguagem de consultas orientada aobjetos: HQL.

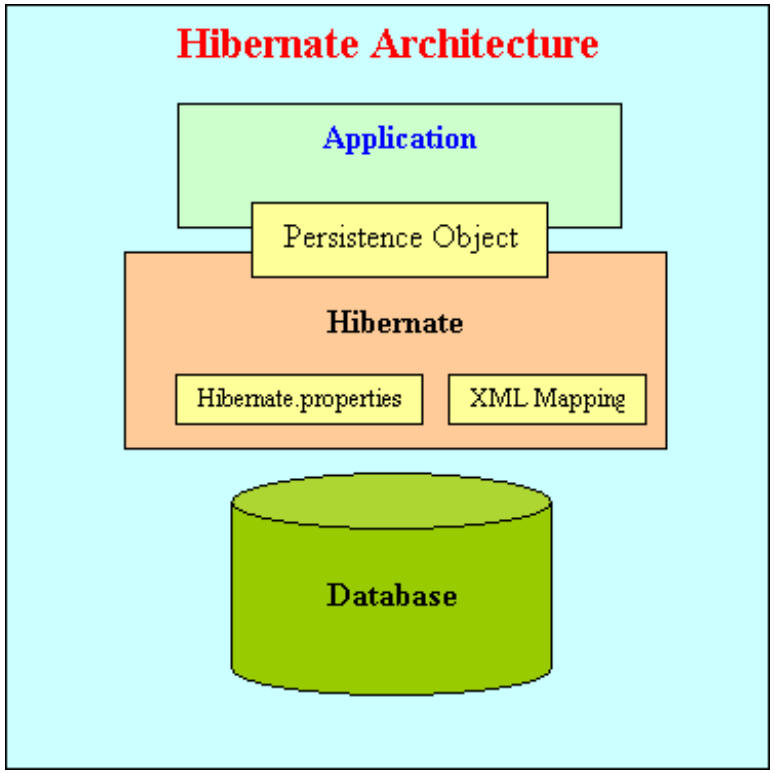
Mapeamento Objeto Relacional

Como tanto Java como SQL ('modelo relacional') não prevêem o mapeamento 'objeto-relacional' este é obtido por 'meios externos':

- Um arquivo <xml> que associa uma classe a uma tabela no SGBD e cada um dos seus campos às colunas da tabela, ou
- Através de anotações no código Java da classe.

A partir do Mapeamento Objeto Relacional, o Hibernate gera para cada classe mapeada uma classe derivada que inclui os métodos associados ao acesso ao SGBD.

Arquitetura



- Hibernate.properties: configuração do Hibernate com relação ao SGBD (url, senha, driver JDBC, etc.).
- XML mapping: associa cada classe a uma tabela no SGBD e seus campos às colunas.
- Persistence Object: classe derivada da classe mapeada da aplicação, que contém os métodos para acesso ao SGBD.

Construção de uma aplicação

Para a construir uma aplicação baseada no Hibernate deve-se seguir os seguintes passos:

- criar as tabelas no SGBD onde os objetos vão *persistir*.
- criar os objetos cujo estado será *persistido*.
- criar um XML, que relaciona as propriedades dos objetos às tabelas e seus campos às colunas.
- criar o arquivo *hibernate.properties* que contém as propriedades para que o Hibernate se conecte ao bando de dados.
- criar as classes para acesso aos objetos através do Hibernate (classes DAO - *Data Access Object*).

Criação das Tabelas no SGBD

Um exemplo:

```
CREATE TABLE aluno
(
  aluno_matricula character varying(10) NOT NULL
                  DEFAULT ''::character varying,
  aluno_nome character varying(40) NOT NULL
                  DEFAULT ''::character varying,
  aluno_rg character varying(10) NOT NULL
                  DEFAULT ''::character varying,
  CONSTRAINT aluno_pkey PRIMARY KEY (aluno_matricula)
)
WITHOUT OIDS;
ALTER TABLE aluno OWNER TO postgres;
```

Definição dos objetos persistidos

```
public class Aluno {
   String matricula;  // matrícula - chave primária
   String nome;       // nome do aluno
   String rg;         // rg do aluno

   /** Construtor padrão **/
   public Aluno() {}

   /** setters padrão **/
   public void setMatricula(String m) { matricula = m; }
   public void setNome(String n)      { nome = n;      }
   public void setRg(String r)        { rg = r;        }

   /** getters padrão **/
   public String getMatricula() { return matricula; }
   public String getNome()      { return nome;      }
   public String getRg()        { return rg;        }

}
```

XML associando tabelas às classes

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping
   PUBLIC "-//Hibernate/Hibernate Mapping DTD//EN"
   http://hibernate.sourceforge.net/hibernate-mapping.dtd
>
<hibernate-mapping>
  <class name="exemplo1.Aluno" table="aluno">
     <id name="matricula" column="aluno_matricula" type="string">
         <generator class="assigned"/>
     </id>
     <property name="nome" column = "aluno_nome" type="string"/>
     <property name="rg"   column="aluno_rg" type="string"/>
  </class>
</hibernate-mapping>
```

Hibernate.properties

```
## PostgreSQL

hibernate.dialect org.hibernate.dialect.PostgreSQLDialect
hibernate.connection.driver_class org.postgresql.Driver
hibernate.connection.url jdbc:postgresql://localhost/teste2
hibernate.connection.username postgres
hibernate.connection.password postgres
hibernate.query.substitutions yes 'Y', no 'N`

## MS SQL Server

#hibernate.dialect org.hibernate.dialect.SQLServerDialect

## JSQL Driver
#hibernate.connection.driver_class com.jnetdirect.jsql.JSQLDriver
#hibernate.connection.url jdbc:JSQLConnect://1E1/test
```

Classe de Acesso aos Objetos

```
public class DaoAlunos{

    private SessionFactory factory;

    public DaoAlunos() throws Exception{
        factory = new Configuration()
                        .addClass(Aluno.class)
                        .buildSessionFactory();
    }

    public void insert(Aluno aluno) throws Exception{
        Session session = factory.openSession();
        Transaction tx = session.beginTransaction();
        session.persist(aluno);
        tx.commit();
        session.close();
    }
```

Classe de Acesso aos Objetos (cont)

```
public class DaoAlunos{
   ...
   public Aluno retrieve(String pk) throws Exception{
       Session session = factory.openSession();
       Aluno aluno = (Aluno)session.get(Aluno.class, pk);
       session.flush();
       session.close();
       return aluno;
    }

    public void delete(Aluno aluno) throws Exception{
       Session session = factory.openSession();
       Transaction tx = session.beginTransaction();
       session.delete(aluno);
       tx.commit();
       session.close();
    }
}
```

Objetos e chaves estrangeiras

Na figura acima:

- Um objeto Alocação associa um Professor a um Curso.
- Ao se criar as tabelas corresponentes essas associações correspondem a chaves estrangeiras na tabela professor.
- Na aplicação baseada em Hibernate, ao se recuperar um objeto Alocacao, os objetos Curso e Professor associados são recuperados automaticamente.

Consultas

O Hibernate permite as seguintes formas de consultas:

- Linguagem de consultas Hibernate (**HQL**): linguagem voltada à consulta de objetos, similar a SQL.
- Consultas por critérios (**QBC**) e consultas por exemplos
- Consultas via **SQL** nativo do SGBD

Anotações

Anotações são diretivas colocadas no código Java para associar as classes a tabelas no SGBD, substituindo o arquivo XML. Um exemplo:

```
@Entity
@Table(name="aluno")
public class Aluno {
   @Id
   @GeneratedValue(strategy=GenerationType.ASSIGNED)
   String matricula;  // matrícula - chave primária

   @Column(name="aluno_nome")
   String nome;       // nome do aluno

   public Aluno() {}

   public void setMatricula(String m) { matricula = m; }
   public void setNome(String n)      { nome = n;      }

   public String getMatricula() { return matricula; }
   public String getNome()      { return nome;      }
}
```