Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе №2**

по дисциплине «Базы данных»

«Клиентское приложение»

Работу выполнила

студентка гр. 43501/3

К.В. Ардашова

Проверил преподаватель

А.В. Мяснов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016г

Санкт - Петербург

2016

**Цель работы**

Ознакомление с концепцией объектно-реляционного отображения.

**Программа работы**

1. Ознакомиться с технологией Hibernate.
2. Создать приложение и реализовать объектно-реляционное отображение.
3. Реализовать всю функциональность приложения посредством ORM.
4. Проанализировать достоинства и недостатки изученной технологии.

**Ход работы**

Hibernate — open sourse библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения (object-relational mapping — ORM). Предоставляет собой легкий в использовании фреймворк для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных.

Файл настройки библиотеки Hibernate (hibernate.cfg.xml) содержит информацию о подключении к базе данных.

Текст файла hibernate.cfg.xml:

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">org.hibernate.dialect.FirebirdDialect</property>

<property name="hibernate.connection.driver\_class">org.firebirdsql.jdbc.FBDriver</property>

<property name="hibernate.connection.url">jdbc:firebirdsql://localhost:3050/D:/IT/LABS/films.fdb</property>

<property name="hibernate.connection.username">SYSDBA</property>

<property name="hibernate.connection.password">masterkey</property>

<property name="hibernate.show\_sql">true</property>

<property name="connection.characterEncoding">cp1251</property>

<mapping resource="hibernate.hbm.xml"/>

<mapping resource="tables/tickets\_sold.hbm.xml"/>

<mapping resource="tables/award.hbm.xml"/>

<mapping resource="tables/session.hbm.xml"/>

<mapping resource="tables/film.hbm.xml"/>

<mapping resource="tables/actor.hbm.xml"/>

<mapping resource="tables/stage\_director.hbm.xml"/>

<mapping resource="tables/link\_film\_actor.hbm.xml"/>

<mapping resource="tables/link\_role\_award.hbm.xml"/>

<mapping resource="tables/link\_award\_film.hbm.xml"/>

</session-factory>

</hibernate-configuration>

Для использования библиотеки Hibernate необходимо создать вспомогательный класс для обработки запуска и обращения к SessionFactory библиотеки Hibernate для получения объекта "Session". Класс обеспечивает вызов метода configure() библиотеки Hibernate, загрузку файла настройки hibernate.cfg.xml и последующую сборку SessionFactory для получения "Session".

Класс HibernateUtil:

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

// Create the SessionFactory from standard (hibernate.cfg.xml)

// config file.

sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();

} catch (Throwable ex) {

// Log the exception.

System.err.println("Initial SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

Файл обратного проектирования (hibernate.reveng.xml) является файлом XML, который можно использовать для изменения настроек по умолчанию, примененных при создании файлов Hibernate из метаданных базы данных, определенной в hibernate.cfg.xml. Мастер создает файл с основными настройками по умолчанию. Можно изменить файл и явным образом указать используемую схему базы данных, отфильтровать таблицы, которые не следует использовать, а также указать, как типы JDBC сопоставляются с типами Hibernate.

Файл hibernate.reveng.xml:

<hibernate-reverse-engineering>

<schema-selection/>

<table-filter match-name="FILM"/>

<table-filter match-name="ACTOR"/>

<table-filter match-name="LINK\_FILM\_ACTOR"/>

<table-filter match-name="STAGE\_DIRECTOR"/>

<table-filter match-name="AWARD"/>

<table-filter match-name="LINK\_AWARD\_FILM"/>

<table-filter match-name="LINK\_ROLE\_AWARD"/>

<table-filter match-name="TICKETS\_SOLD"/>

<table-filter match-name="SESSION1"/>

</hibernate-reverse-engineering>

Файлы сопоставления представляют собой пары файлов:

* 1. Java-класс вида:

public class Actor implements java.io.Serializable {

private Integer id;

private String name;

private Date birthdate;

public Actor() {

}

public Actor(String name, Date birthdate) {

this.name = name;

this.birthdate = birthdate;

}

public Integer getId() {

return this.id;

}

public void setId(Integer newValue) {

this.id = id;

}

public String getName() {

return this.name;

}

public void setName(String newValue) {

this.name = newValue;

}

public Date getBirthdate(){

return this.birthdate;

}

public void setBirthdate(Date newValue){

this.birthdate = newValue;

}

}

* 1. Файл сопоставления в формате xml:

<hibernate-mapping>

<class name="tables.Actor" table="actor">

<id name="id" type="integer">

<column name="id\_actor"/>

<generator class="identity"/>

</id>

<property name="name" type="string">

<column length="255" name="name" not-null="true"/>

</property>

<property name="birthdate" type="date">

<column name="birthdate" not-null="true"/>

</property>

</class>

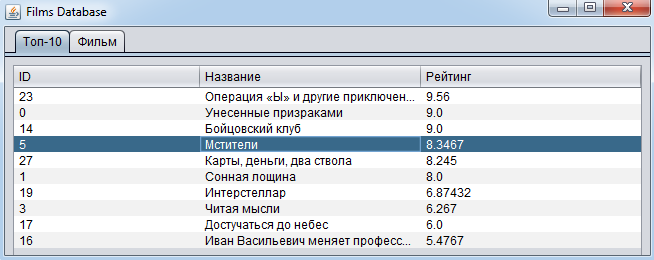
</hibernate-mapping>

**Описание приложения**

В данной работе с помощью вышеописанной технологии реализовано графическое приложение, осуществляющее отображение списка из десяти фильмов с наибольшим рейтингом и возможностью просмотра данных о выбранном фильме, а также ввода данных о приобретенных билетах.

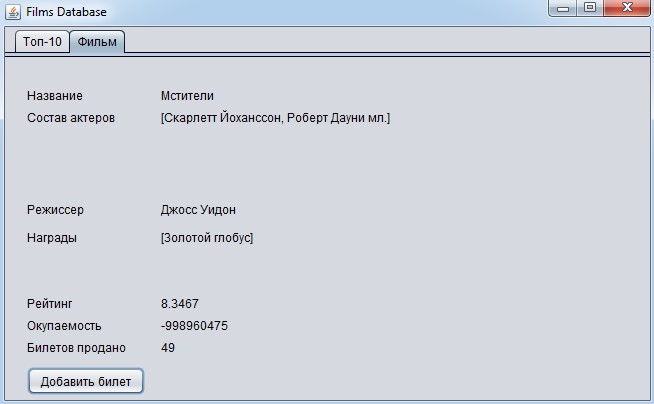
Основное окно приложения состоит из двух вкладок:

1. Отображение топ-10 фильмов (загружается по умолчанию).

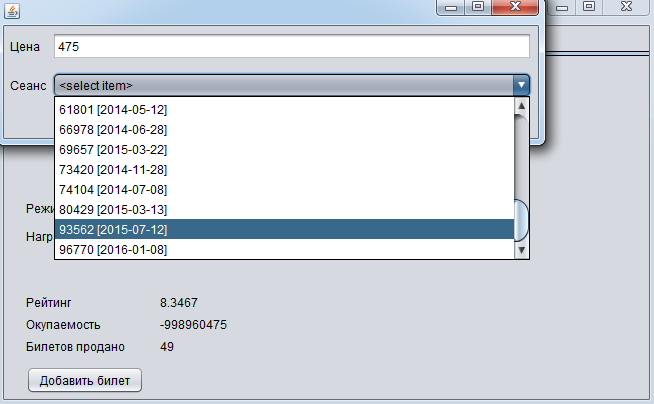


Если нажать на строку с фильмом, открывается вторая вкладка.

1. Отображение информации о фильме

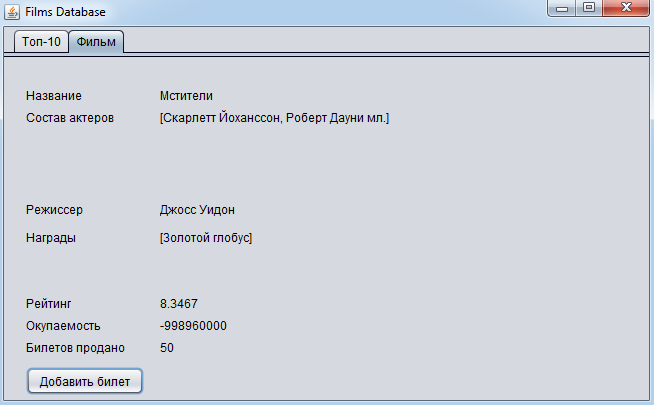


1. Форма добавления билета:



В данном случае для добавления записи используются SQL-запросы, т.к. в HQL возможен только запрос вида insert into .. select..

После добавления билета количество билетов и окупаемость пересчитываются:



**Выводы**

Hibernate значительно уменьшает время разработки приложений, работающих с базами данных.

Существенным в плане сокращения времени разработки является отсутствие необходимости составлять парсеры для результатов запросов.

Ещё одно преимущество технологии заключается в том, что при необходимости изменить названия полей в базе данных в случае, если код уже написан, в приложении название можно изменить только в одной строке, а не во всем приложении.

Неудобством является отсутствие в HQL запросов вида insert, а также конструкции вроде «select first …» - для них требуется использовать SQL.