

课程名称: JavaEE 技术	年级: XXXX	实验日期:
姓名: XXX	学号: XXXXX	班级: XXXX
实验名称: Hibernate 持久层编程	实验序号: 实验五	成员人数: 1

一、实验目的及要求

实验目的:

- 1、理解 orm 框架;
- 2、掌握 Hibernate 的本质 (对 JDBC 进行了轻量级的封装)

实验原理:

- 1、Hibernate 可以按照不同数据库, 使用最优化的 SQL 语句处理不同的操作;
- 2、引入 Hibernate 可以是工作人员角色细化; 让程序员更关系业务流程;
- 3、Hibernate 使得分层更加清晰, 耦合性更小, 通用性更强。

二、实验环境(仪器与材料)

Windows 7/10, MySQL/MariaDB, Eclipse, Tomcat

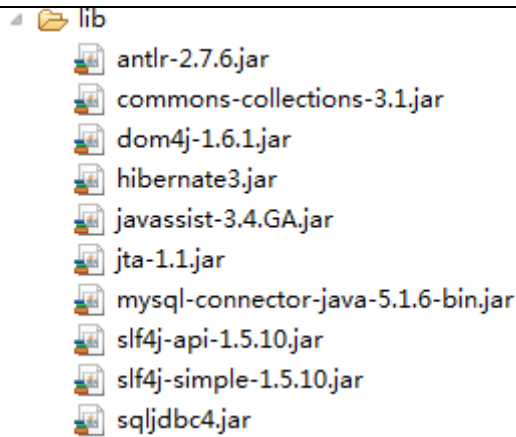
三、实验内容及完成情况

1、创建雇员数据库

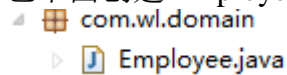
```
create database Employee --库名为Employee
on
( name='Employee_data',
  filename='d:\Employee_data.mdf',
  size=10mb
)
log on
( name='Employee_log',
  filename='d:\Employee_log.ldf',
  size=10mb
)
go
use Employee
go
create table Employee --创建雇员表
(
  id int primary key, --标签编号, 主码
  empname nvarchar(20) not null,
  --雇员姓名可变字符串类型, 长度
  email nvarchar(50),
  hiredate date not null
)
```

2、引入 sql 驱动包 sqljdbc4.jar。

3、引入 hibernate 包。



4、创建 com.wl.domain 包。在这个包下面创建 Employee 类。



5、编写 Employee 类代码。

```
package com.wl.domain;
public class Employee {
    private Integer id;
    private String name;
    private String email;
    private java.util.Date hiredate;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public java.util.Date getHiredate() {
        return hiredate;
    }
    public void setHiredate(java.util.Date hiredate) {
        this.hiredate = hiredate;
    }
}
```

6、在 com.wl.domain 包下面手工创建 Employee.hbm.xml 文档。(实际开发项目时，是自动生成的 xml 配置文件，但是建议初学者第一次手工编写 xml 文件，以便加深印象。)

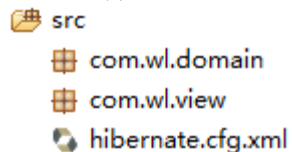


7、Employee.hbm.xml 文档内容如下：

```

<?xml version="1.0" encoding="utf-8"?>
<!-- 加入DTD约束文件，可以从驱动包里找到 -->
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.wl.domain">
<class name="Employee" table="Employee">
    <!-- id用于指定主码 -->
    <id name="id" column="id" type="java.lang.Integer">
        <!-- generator 用于指定主码的生成策略 -->
        <generator class="increment">
            <param name="id"></param>
        </generator>
    </id>
    <!-- 配置除了主码之外的其他属性 -->
    <property name="name" type="java.lang.String">
        <column name="empname" not-null="false"/>
    </property>
    <property name="email" type="java.lang.String">
        <column name="email" not-null="true"/>
    </property>
    <property name="hiredate" type="java.util.Date">
        <column name="hiredate" not-null="false"/>
    </property>
</class>
</hibernate-mapping>
  
```

8、在 src 下面编写 hibernate.cfg.xml 配置文件。



9、hibernate.cfg.xml 文档内容如下：

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/
hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
    <property name="connection.driver_class">
        com.microsoft.sqlserver.jdbc.SQLServerDriver
    </property>
    <property name="hibernate.connection.username">
        sa
    </property>
    <property name="hibernate.connection.password">
  
```

```

123456
</property>
<property name="hibernate.connection.url">
    jdbc:sqlserver://localhost:1433;DatabaseName=Employee
</property>
<property name="hibernate.dialect">
    org.hibernate.dialect.SQLServerDialect</property>
<property name="hibernate.show_sql">true</property>
<mapping resource="com/wl/domain/Employee.hbm.xml"/>
</session-factory>
</hibernate-configuration>

```

10、创建 com.wl.view 包，在这个包下面编写测试主函数 EmployeeTest.java。

```

package com.wl.view;
import java.util.Date;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.*;
import com.wl.domain.Employee;
public class EmployeeTest {
    public static void main(String[] args) {
        //使用hibernate 完成数据库的CRUD操作。
        //1.创建configuration,该对象用于读取hibernate.cfg.xml,
        //并完成初始化工作
        Configuration configuration=new
Configuration().configure("hibernate.cfg.xml");
        //2.创建SessionFactory,会话工厂,是一个重量级的对象。一个项目中只
        //能加载一次,最好使用单态模式。
        SessionFactory sessionFactory
=configuration.buildSessionFactory();
        //3.创建会话session
        Session session=sessionFactory.openSession();
        //4.添加一条雇员信息
        Employee employee=new Employee();
        //employee.setId(111);
        //此处要注意:创建数据库时,如果指定identity(1,1)自增长策略,则配置
        //Employee.hbm.xml文件时,必须配置主码策略为native,或者identity.
        //<!-- generator 用于指定主码的生成策略 -->
        //      <generator class="native">
        //          <param name="id"></param>
        //      </generator>
        employee.setName("wl");
        employee.setEmail("wl@126.com");
        employee.setHiredDate(new Date());
        //在hibernate中执行增加、删除和修改语句,必须使用事务。查询不需要。
        //5.创建一个事务
        Transaction ransaction=session.beginTransaction();
        //6.事务开始执行
        transaction.begin();
        session.save(employee);
        //7.提交事务
    }
}

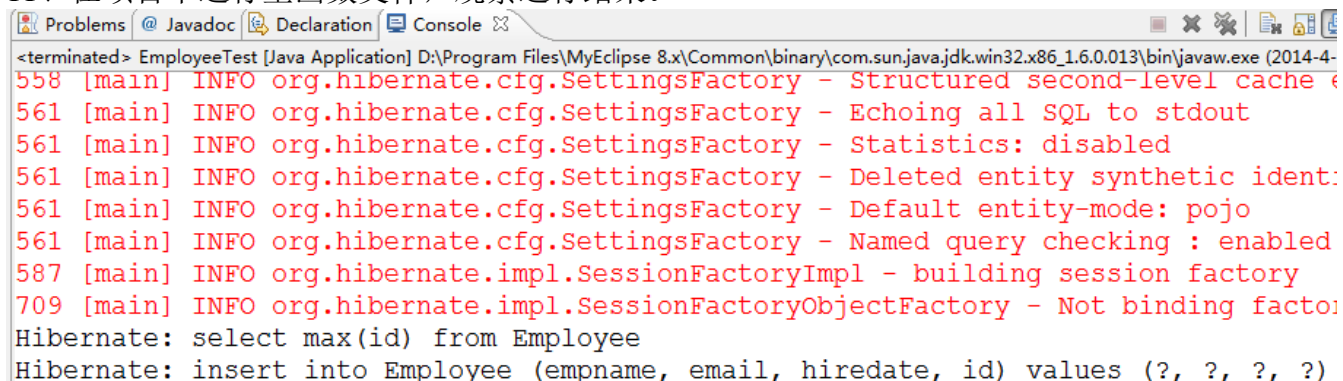
```

```

        transaction.commit();
        //8.一定要及时释放资源。
        session.close();
    }
}

```

11、在项目中运行主函数文件，观察运行结果。



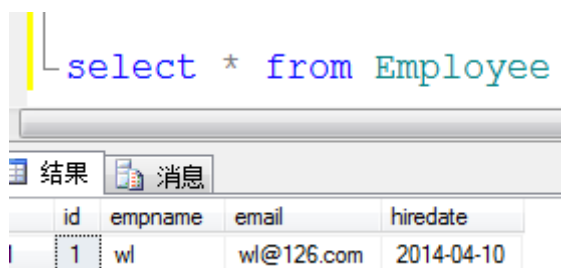
The screenshot shows the Eclipse IDE console with the following output:

```

<terminated> EmployeeTest [Java Application] D:\Program Files\MyEclipse 8.x\Common\binary\com.sun.java.jdk.win32.x86_1.6.0.013\bin\javaw.exe (2014-4-
558 [main] INFO org.hibernate.cfg.SettingsFactory - Structured second-level cache e
561 [main] INFO org.hibernate.cfg.SettingsFactory - Echoing all SQL to stdout
561 [main] INFO org.hibernate.cfg.SettingsFactory - Statistics: disabled
561 [main] INFO org.hibernate.cfg.SettingsFactory - Deleted entity synthetic ident:
561 [main] INFO org.hibernate.cfg.SettingsFactory - Default entity-mode: pojo
561 [main] INFO org.hibernate.cfg.SettingsFactory - Named query checking : enabled
587 [main] INFO org.hibernate.impl.SessionFactoryImpl - building session factory
709 [main] INFO org.hibernate.impl.SessionFactoryObjectFactory - Not binding facto
Hibernate: select max(id) from Employee
Hibernate: insert into Employee (empname, email, hiredate, id) values (?, ?, ?, ?)

```

12、观察后台数据库变化。



The screenshot shows a SQL query result in a database client. The query is `select * from Employee`. The result is displayed in a table with the following columns: id, empname, email, and hiredate. The first row of data shows id=1, empname=wl, email=wl@126.com, and hiredate=2014-04-10.

id	empname	email	hiredate
1	wl	wl@126.com	2014-04-10

四、 出现的问题及解决方案

1、主码生成策略常见的有：

(1) 自动增长 identity

适用于 MySQL、DB2、MS SQL Server，采用数据库生成的主键，用于为 long、short、int 类型生成唯一标识。

(2) sequence

DB2、Oracle 均支持的序列，用于为 long、short 或 int 生成唯一标识。

(3) 3、hilo

使用一个高/低位算法生成的 long、short 或 int 类型的标识符，给定一个表和字段作为高位值的来源，默认的表是 `hibernate_unique_key`，默认为的字段是 `next_hi`。它将 id 的产生源分成两部分，DB+内存，然后按照算法结合在一起产生 id 值，可以在很少的连接次数内产生多条记录，提高效率。

(4) native

会根据底层数据库的能力，从 identity、sequence、hilo 中选择一个，灵活性更强，但此时，如果选择 sequence 或者 hilo，则所有的表的主键都会从 Hibernate 默认的 sequence 或者 hilo 表中取。并且，有的数据库对于默认情况主键生成测试的支持，效率并不是很高，对于 oracle 采用 Sequence 方式，对于 MySQL 和 SQL Server 采用 identity（自增主键生成机制），native 就是将主键的生成工作交由数据库完成，hibernate 不管（很常用）。

(5) seqhilo

sequence 和 hilo 的结合，hilo 的高位由 sequence 产生，所以也需要底层数据库的支持，通过 hilo 算

法实现，但是主键历史保存在 Sequence 中，适用于支持 Sequence 的数据库，如 Oracle（比较少用）。

(6) increment

这个是由 Hibernate 在内存中生成主键，每次增量为 1，不依赖于底层的数据库，因此所有的数据库都可以使用，但问题也随之而来，由于是 Hibernate 生成的，所以只能有一个 Hibernate 应用进程访问数据库，否则就会产生主键冲突，不能在集群情况下使用，插入数据的时候 hibernate 会给主键添加一个自增的主键，但是一个 hibernate 实例就维护一个计数器，所以在多个实例运行的时候不能使用这个方法。

(7) uuid.hex

使用一个 128-bit 的 UUID 算法生成字符串类型的标识符，UUID 被编码成一个 32 位 16 进制数字的字符串。UUID 包含：IP 地址、JVM 启动时间、系统时间（精确到 1/4 秒）和一个计数器值（JVM 中唯一），hibernate 会算出一个 128 位的唯一值插入。

2、hibernate 执行增加、删除或者修改数据库记录时，必须使用事务进行处理。