struts2: 架构最简单
Hibernate: 架构最复杂,使用以及思想都很简单.

hibernate是一个持久层的orm框架（orm对象关系映射）

我们在Hibernate中需要使用面向对象思想操作数据库.
我们需要告诉Hibernate 我们的对象与 数据库中表的关系.

ORM　　Object Relational Mapping　对象关系映射

O: 对象
R:关系(表)
M:映射(配置文件)

orm元数据===>配置文件

hibernate.cfg.xml配置文件：参考hibernate包下project/etc/hibernate.cfg.xml文件中的内容

如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  ~ Hibernate, Relational Persistence for Idiomatic Java
  ~
  ~ License: GNU Lesser General Public License (LGPL), version 2.1 or later.
  ~ See the lgpl.txt file in the root directory or
<http://www.gnu.org/licenses/lgpl-2.1.html>.
  -->
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
 <hibernate-configuration>
        <session-factory >
<!--    数据库基本信息 -->
连接数据库的驱动
                <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
连接数据库的用户名
                <property name="hibernate.connection.username">root</property>
```

连接数据库的密码

```
                <property
name="hibernate.connection.password">AQL271422</property>
```

连接数据库的路径

```
                <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hb1</property>
                <!-- show_sql操作数据库时，会向控制台打印SQL语句 -->
                <property name="show_sql">true</property>
                <!-- format_sql操作数据库时，会将SQL语句先格式化 -->
                <property name="format_sql">true</property>
                 <!-- hbm2ddl.auto是否自动生成表结构（后面介绍） -->
                <property name="hbm2ddl.auto">update</property>
                <!-- 事务自动提交 -->
                <property
name="hibernate.connection.autocommit">true</property>
```

引入orm映射文件，注意填写src之后的路径

```
                    <mapping resource="hibernateday1/user.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

hbm.xml配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<!--表示要将哪个文件的对象映射到哪张表上去-->
```

class配置实体与表的关系，其中name填写实体的完整类名，table填写与实体对应的表的名称

```
        <class name="hibernateday1.user" table="users">
```

id用于配置实体与表中id对应，name是user对象中标识主键的属性名称，column标识表中的列名

```
                <id name="id" column="id">
```

generator表示主键的生成策略：native表示由数据库来维护（数据库中的配置是主键自增，其它取值后期再说

```
                        <generator class="native"></generator>
                </id>
//注意：列名column的值要和数据库中表的名字一致
property实体中属性与表中的列对应，name表示实体中属性名称，column表示表中列的名称
                <property name="name" column="name"></property>
                <property name="passworld" column="password"></property>
        </class>
</hibernate-mapping>
```

重要：给.cfg.xml和.hbm.xml配置文件配置自动提示功能：

①在referenced libraries下找到hibernate的核心包，在核心包下的org.hibernate下找到hibernate-configuration...和hibernate.mapping...文件。

②打开相应的文件，分别复制

```
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
```

和

```
   <!DOCTYPE hibernate-mapping PUBLIC
       "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
       "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
```

到相应的配置文件的第二行代码处

property：配置文件中的这个元素用于配置hibernate中的属性，是以键值对的形式存在；

<property>值</property>

mapping：引入映射文件

测试：

```
@Test
public void run(){
```

//Configuration用于加载配置文件

//1.读取配置文件

//1.1调用configure（）方法=>加载src下面名为hibernate.cfg.xml

Configuration configuration = new Configuration().configure();

//1.2如果配置文件不符合默认的加载规则，我们可以调用

//new Configuration().configure(file)通过file加载或 new

//Configuration().configure(path)通过path加载;

或1.3通过Configuration对象推荐映射文件（不推荐），推荐hibernate.cfg.xml使用mapping属性引入配置文件。若没有，则可以用conf.addClass(实体类名.class);来手动加载hbm文件

规范：orm映射文件名称与实体的简单类名一致；orm映射文件需要与实体的类在同一个包下

//2，根据配置创建factory

sessionFactory存放的是session，用来创建session，然后session在取访问数据库

1，如何创建sessionFactory：根据Configuration配置信息创建sessionFactory

SessionFactory sessionFactory = configuration.buildSessionFactory();

//3，根据factory获得操作数据库的session对象

2，如何从sessionFactory中取出session对象：session对象.openSession（）；

（openSession获得一个全新的session对象（测试阶段使用）；getCurrentSession获得当前的session对象（上线后使用），调用getCurrentSession需要加上一个配置<property name="hibernate.current_session_context_class">thread</property>写在.cfg.xml文件中，将session与县城绑定，只有配置了该配置，才能使用getCurrentSession）

Session session = sessionFactory.openSession();

//4，操作数据库

相当于jdbc中的连接池，session对象，用于操作数据库，

1，增：操作数据库的增加：创建一个实体对象，并给每个属性设置相应的值，调用session的save方法保存到数据库中，若SQL语句在控制台打印的话，表明数据插入成功。

user user = new user();

user.setName("anqili");

user.setPassworld("12321435446");

session.save(user);

2，改：在查询结果的结果上修改对象

//打开事务

```
        Transaction ts = session.beginTransaction();
        user user2 = session.get(user.class, 2);
        user2.setName("linmingjun");
        session.update(user2);
        //提交事务
        ts.commit();
```

3，查：user user2 = session.get(user.class, 2);

实体类 实体名 = session,get（实体名.class，查询的值）

```
        user user2 = session.get(user.class, 2);

System.out.println(user2.getId()+","+user2.getName()+","+user2.getPassworld());
```

4，删：//删除

```
        user user3 = (user)session.get(user.class, 2);
        session.delete(user3);
```

操作完之后需要关闭session和sessionfactory对象

```
        //5，关闭资源
        session.close();
}
```

hibernate中不允许没有主键

hibernate07（看到）