



燕山大学
YANSHAN UNIVERSITY

《JavaWeb 开发》三级项目任务说明书

题目： Javaweb 开发授课

学院（系）： 信息科学与工程学院

年级专业： 软件工程六班

学生学号： 201811040809

学生姓名： 乔翱

校内教师： 梁顺攀

教师职称：

企业教师： 田大强

教师职称： 高级开发工程师

1. 课程说明:

课程时间	2020.6.20~2020.6.21	实施地点	线上直播讲授	课程学时	16
课程目的	使学生通过本课程进一步巩固所学的基础知识。了解企业对前后端开发的技术需求;掌握基于 Java Web 技术项目开发所需知识,具备进行 Java Web 项目开发的能力。学习掌握 Java Web 项目开发的集成开发工具的使用,能对调试程序,完成相关的测试,解决相关问题。				
课程内容	1. Struts2 Hibernate 实现用户登录功能 2. Spring 整合 Struts2、Hibernate 后端开发技术				
工作要求	1. 按时上线进行知识学习保障出勤; 2. 按照要求完成课堂案例编写并提交; 3. 按照要求完成本课程规定的课程案例编写并提交; 4. 按照要求完成本课程报告编写并提交。				
开发环境	前端 IDE: HBuilder 后端 IDE: Eclipse\IDEA JAVA 工具: JDK1.8 数据库: MySQL5.7 数据库管理: Navicat 项目管理: Maven3.6 Web 容器: Tomcat9				
工作计划	第 1 天: 明确课程要求及计划;掌握基于 Maven 的 Java Web 项目开发流程;Struts2 开发环境搭建,应用 Struts2 实现模拟用户登录操作;Hibernate 开发环境搭建,Struts2 Hibernate 实现用户登录功能;完成前端课程任务。 第 2 天: 了解后端开发技术构成;Spring 开发环境搭建、Spring IOC 与 AOP 快速讲解, Spring 整合 Struts2、Hibernate 实现用户注册的核心开发案例;完成后端课程任务。				
课程成果	1. 后端: 基于 Struts 的用户登录系统,基于 Hibernate 的持久层实现,基于 Spring 的案例。 2. 基于 SSH 完成学生信息管理操作				

2. 任务列表

2.1 任务一

任务名称	基于 Struts2 实现用户登录功能模块的设计实现
开发环境	IDEA，Chrome 版本 83.0.4103.61 Tomcat v9.0 JDK1.8
完成功能	利用 Struts2 框架实现用户登录功能
任务内容	<p>1. 主要技术点</p> <p>Struts2</p> <p>利用 Struts2 框架，对前端 jsp 页面中用户输入的用户名和密码进行一个检验，实现用户登录模块</p> <p>2. 运行效果图</p> <p>登录界面截图：</p>  <p>登录成功后跳转的页面的截图：</p>  <p>控制台输出截图：</p> <p>我是Helloaction</p> <p>UserInfo{id=0, name='admin', password='123456', slary=0.0}</p> <p>3. 核心代码</p> <p>在 pom.xml 文件中引入依赖</p> <pre><!-- https://mvnrepository.com/artifact/org.apache.struts/struts2-core --> <dependency> <groupId>org.apache.struts</groupId> <artifactId>struts2-core</artifactId> <version>2.5.16</version> </dependency></pre> <p>修改 web.xml 文件引入 struts2</p> <pre><!DOCTYPE web-app PUBLIC</pre>

```

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >
<web-app>
  <display-name>Archetype Created Web
Application</display-name>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.filter.StrutsPr
epareAndExecuteFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>

```

创建实体类

```

package com.chinasofti.struts2.entity;
import java.io.Serializable;
/**
 * Created by qa on 2020/6/20
 */
public class UserInfo implements Serializable {
    public UserInfo() {
    }
    public UserInfo(int id, String name, String password,
double salary) {
        this.id = id;
        this.name = name;
        this.password = password;
        this.salary = salary;
    }
    @Override
    public String toString() {
        return "UserInfo{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", password='" + password + '\'' +
            ", salary=" + salary +
            '}';
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {

```

```

        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public double getSlary() {
        return slary;
    }
    public void setSlary(double slary) {
        this.slary = slary;
    }
    private int id;
    private String name;
    private String password;
    private double slary;
}

```

前端登录主页面 Index.jsp

```

<%--
    Created by IntelliJ IDEA.
    User: qa
    Date: 2020/6/20
    Time: 11:41
    To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8"
    language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<form action="user/helloAction" method="post">
    用户名<input type="text" name="user.name" placeholder="请
输入账号"><br>
    密码<input type="password" name="user.password"

```

```
placeholder="请输入密码"><br>
    <input type="submit" value="登录">
</form>
</body>
</html>
```

HelloAction（校验用户名密码，设置只有用户名是“admin”，密码是 123456 的时候登录成功跳转到 welcome.jsp，否则登陆失败）

```
package com.chinasofti.struts2.action;
import com.chinasofti.struts2.entity.UserInfo;
import com.opensymphony.xwork2.ActionSupport;
/**
 * Created by qa on 2020/6/20
 */
public class HelloAction extends ActionSupport {
    //访问 action 调用的方法
    private UserInfo user;
    public UserInfo getUser() {
        return user;
    }
    public void setUser(UserInfo user) {
        this.user = user;
    }
    @Override
    public String execute() throws Exception {
        System.out.println("我是 Helloaction");
        System.out.println(user);
        if
("admin".equals(this.user.getName()) && "123456".equals(this.
user.getPassword()))
            return "welcome";
        else {
            return "index";
        }
    }
}
```

登录成功时的欢迎界面 welcome.jsp

```
<%--
    Created by IntelliJ IDEA.
    User: qa
    Date: 2020/6/20
    Time: 11:41
    To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html; charset=UTF-8"
```

	<pre>language="java" %> <html> <head> <title>Title</title> </head> <body> 欢迎哈哈 </body> </html></pre> <p>Struts 配置文件 struts.xml, 配置 action</p> <pre><?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN" "http://struts.apache.org/dtds/struts-2.5.dtd"> <struts> <!-- 创建命名空间 访问路径--> <package name="user" namespace="/user" extends="struts-default"> <!--配置 action name 访问路径名称 class 指向哪一个--> <action name="helloAction" class="com.chinasofti.struts2.action.HelloAction"> <result name="welcome">/welcome.jsp</result> <result name="index">/index.jsp</result> </action> </package> </struts></pre>
任务总结	<p>1. 通过此任务的收获</p> <p>通过本次任务，了解和掌握了使用 struts2 框架开发的一个基本的流程，对 struts2 框架有了一个简答的入门。</p> <p>Struts2 是一个基于 MVC 设计模式的 web 应用框架，作为控制器来建立模型与视图的数据交互，利用 struts2 可以方便程序员实现模型与视图的数据交互，相对于传统的 servlet 获取前端数据的方法更加简便。</p> <p>在此次任务中，只是简单了解了 struts2 框架，对于这种通过配置文件来编码实现数据交互的方法是第一次接触，对于 struts2 的深层次的使用还有待学习以及进一步的研究</p> <p>2. 此任务的可扩展性</p> <p>目前，本任务只是完成了一个简单的利用 struts2 框架实现登录的功能，只是简单实现了模型与视图的数据交互，没有对数据库进行操作，接下来可以利用 jdbc 或者是 hibernate 进行持久化层的编写，就行对数据库的操作，来校验用户名密码是否正确实现一套整体的用户登录的 MVC 模式完整系统。并且在后期可以对本登录页面进行扩展，例如可以利用 bootstrap 和 jQuery 进行前端的编写，还可以对功能进行完善例如实现先注册再登录，登录页面进行验证码校验等功能。</p>

2.2 任务二

任务名称	基于 hibernate 实现数据库单表 CURD 设计																				
开发环境	IDEA, Chrome 版本 83.0.4103.61, JDK1.8, MySQL5.7, Navicat																				
完成功能	使用 hibernate 实现对数据库 ssh 中的 userinfo 表的增删改查功能																				
任务内容	<div>1. 主要技术点</div> <div>Hibernate MySQL</div> <div>利用 hibernate 框架, 实现对 mysql 数据库的增删改查操作</div> <div>2. 运行效果图</div> <div>插入一个用户:</div> <div><div>Hibernate:</div><div>insert</div><div>into</div><div>userinfo</div><div>(name, password, salary)</div><div>values</div><div>(?, ?, ?)</div></div> <div><table><tr><th>id</th><th>name</th><th>password</th><th>salary</th></tr><tr><td>1</td><td>小hong</td><td>123547</td><td>1250</td></tr><tr><td>2</td><td>小明</td><td>123456</td><td>5550</td></tr></table></div> <div>删除一个用户:</div> <div><div>Hibernate:</div><div>delete</div><div>from</div><div>userinfo</div><div>where</div><div>id=?</div></div> <div>更新一个用户的信息:</div> <div><div>Hibernate:</div><div>update</div><div>userinfo</div><div>set</div><div>name=?,</div><div>password=?,</div><div>salary=?</div><div>where</div><div>id=?</div></div> <div><table><tr><th>id</th><th>name</th><th>password</th><th>salary</th></tr><tr><td>2</td><td>小刚</td><td>123547</td><td>1250</td></tr></table></div>	id	name	password	salary	1	小hong	123547	1250	2	小明	123456	5550	id	name	password	salary	2	小刚	123547	1250
id	name	password	salary																		
1	小hong	123547	1250																		
2	小明	123456	5550																		
id	name	password	salary																		
2	小刚	123547	1250																		

查询所有用户:

```
Hibernate:
select
    userinfo0_.id as id1_0_,
    userinfo0_.name as name2_0_,
    userinfo0_.password as password3_0_,
    userinfo0_.salary as salary4_0_
from
    userinfo userinfo0_
UserInfo{id=2, name='小刚', password='123547', salary=1250.0}
UserInfo{id=3, name='小明', password='123456', salary=5550.0}
UserInfo{id=4, name='小红', password='1ss26', salary=8720.0}
```

根据 id 查询某个用户:

```
Hibernate:
select
    userinfo0_.id as id1_0_0_,
    userinfo0_.name as name2_0_0_,
    userinfo0_.password as password3_0_0_,
    userinfo0_.salary as salary4_0_0_
from
    userinfo userinfo0_
where
    userinfo0_.id=?
UserInfo{id=3, name='小明', password='123456', salary=5550.0}
```

3. 核心代码

在 pom.xml 文件中导入依赖

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.2.Final</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.15</version>
</dependency>
```

数据库创建 userinfo 表

```
SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

DROP TABLE IF EXISTS `userinfo`;
CREATE TABLE `userinfo` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NULL
    DEFAULT NULL,
    `password` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin
```

```
NULL DEFAULT NULL,  
    `salary` double(255, 0) NULL DEFAULT NULL,  
    PRIMARY KEY (`id`) USING BTREE  
) ENGINE = InnoDB AUTO_INCREMENT = 5 CHARACTER SET = utf8  
COLLATE = utf8_bin ROW_FORMAT = Dynamic;  
  
SET FOREIGN_KEY_CHECKS = 1;
```

创建 UserInfo 实体类

```
package com.chinasofti.hibernate.entity;  
import java.io.Serializable;  
  
/**  
 * Created by qa on 2020/6/20  
 */  
public class UserInfo implements Serializable {  
    private int id;  
    private String name;  
    private String password;  
    private double salary;  
    public UserInfo() {  
        //System.out.println("调用了无参的构造方法");  
    }  
    public UserInfo(int id, String name, String password,  
double salary) {  
        this.id = id;  
        this.name = name;  
        this.password = password;  
        this.salary = salary;  
        System.out.println("调用了有参的构造方法");  
    }  
    @Override  
    public String toString() {  
        return "UserInfo{" +  
            "id=" + id +  
            ", name='" + name + '\'' +  
            ", password='" + password + '\'' +  
            ", salary=" + salary +  
            '}';  
    }  
    public int getId() {  
        return id;  
    }  
    public void setId(int id) {  
        this.id = id;  
    }  
}
```

```

public String getName() {
    return name;
}

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

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}
}

```

创建实体类对应的映射文件 UserInfo.hbm.xml

```

<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.chinasofti.hibernate.entity.UserInfo"
table="userinfo">
        <id name="id" column="id">
            <generator class="native"></generator>
        </id>
        <property name="name" column="name"
type="string"></property>
        <property name="password" column="password"
type="string"></property>
        <property name="salary" column="salary"
type="double"></property>
    </class>
</hibernate-mapping>

```

创建 hibernate 配置文件 hibernate.cfg.xml

```

<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.d
td">

```

```

<hibernate-configuration>
    <session-factory>
        <!--表面当前 hibernate 连接什么数据库-->
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect
        </property>
        <!--jdbc 连接数据    mysql    URL driver username password-->
        <property
name="connection.driver_class">com.mysql.cj.jdbc.Driver</pr
operty>
        <property
name="connection.url">jdbc:mysql://localhost:3306/ssh?serve
rTimezone=UTC&useUnicode=true&characterEncoding=utf
-8</property>
        <property name="connection.username">root</property>
        <property
name="connection.password">12297804qa</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <mapping
resource="com.chinasofti.hibernate.entity/UserInfo.hbm.xml"
></mapping>
    </session-factory>
</hibernate-configuration>

```

测试文件

```

package com.chinasofti.hibernate.test;
import com.chinasofti.hibernate.entity.UserInfo;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.junit.Test;
import java.util.List;

/**
 * Created by qa on 2020/6/20
 */
public class Mytest {
    // @Test
    // public void test(){
    //     // 获取配置对象
    //     Configuration conf=new Configuration().configure();
    //     // 通过配置对象获取 sessionFactory
    //     SessionFactory
    sessionFactory=conf.buildSessionFactory();
}

```


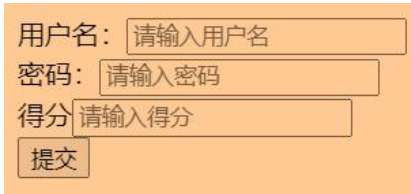
```

//    //获取 session 对象
//    Session session=sessionFactory.openSession();
//    //获取事物控制对象
//    Transaction tx=session.beginTransaction();
//    //执行 sql 语句
//    UserInfo userInfo=new UserInfo();
//    userInfo.setName("小红");
//    userInfo.setPassword("lss26");
//    userInfo.setSalary(8720.5);
//    session.save(userInfo);
//    //事物控制提交
//    tx.commit();
//    //关闭资源
//    session.close();
//    sessionFactory.close();
//}
@Test
//    public void test(){
//        Configuration conf=new Configuration().configure();
//        //通过配置对象获取 sessionFactory
//        SessionFactory
sessionFactory=conf.buildSessionFactory();
//        //获取 session 对象
//        Session session=sessionFactory.openSession();
//        //获取事物控制对象
//        Transaction tx=session.beginTransaction();
//        //执行 sql 语句
//        List<UserInfo> list=session.createQuery("from UserInfo
").list();
//        for (UserInfo us:list)
//            System.out.println(us);
//        //事物控制提交
//        tx.commit();
//        //关闭资源
//        session.close();
//        sessionFactory.close();
//}
//
@Test
public void test(){
    Configuration conf=new Configuration().configure();
    //通过配置对象获取 sessionFactory
    SessionFactory
sessionFactory=conf.buildSessionFactory();

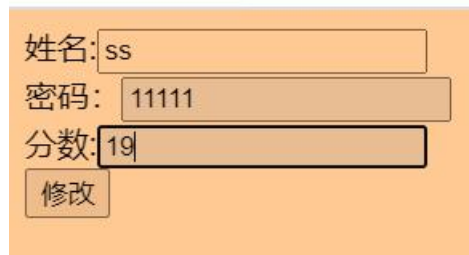
```

	<pre> //获取 session 对象 Session session=sessionFactory.openSession(); //获取事物控制对象 Transaction tx=session.beginTransaction(); //执行 sql 语句 //参数一 Class 类型,根据 id 查询 UserInfo load=session.load(UserInfo.class,3); System.out.println(load); // session.delete(load);//对象类型,自动调用 getId 的 () 方法; //事物控制提交 //UserInfo userInfo=new UserInfo(2,"小刚", "123547",1250.2); //session.update(userInfo); //tx.commit(); //关闭资源 session.close(); sessionFactory.close(); } } </pre>
任务总结	<p>1. 通过此任务的收获</p> <p>通过此任务了解并且掌握了如何利用 hibernate 框架实现对数据库的操作,掌握了利用 hibernate 框架对数据库进行增删改查操作的基本的一个流程。</p> <p>Hibernate 对 jdbc 进行了非常轻量级的对象封装,是有一个全自动的 orm 框架, hibernate 可以自动生成 sql 语句,自动执行。利用 hibernate 可以非常方便的操纵数据库,程序员可以利用 hibernate 非常方便地实现网站的持久层搭建。</p> <p>通过这次任务的学习,很好的入门了 hibernate,并且通过课下代码的练习熟练掌握了基本的使用 hibernate 框架的一套流程。</p> <p>2. 此任务的可扩展性</p> <p>对于已经完成的利用 hibernate 实现的对于数据库的增删改查,还有部分功能没有实现,例如对于数据库的复杂查询等,以及接下来可以对 hibernate 对数据库的操作进行一个封装,减少代码的冗余度,以及接下来进行前端页面的开发,利用 struts 框架整合 hibernate 框架,实现两个框架的整合,实现从前端接收传递给后端的参数,来对前端的 request 请求进行后端数据库中的操作,并且返回给前端界面响应。</p> <p>此外,对于 hibernate 中其他的一些函数 API 以及 hibernate 的面向对象思想操纵对象的特性还有待进一步的学习以及研究。</p>

2.3 任务三

任务名称	基于 Java Web 的学生信息管理系统设计实现
开发环境	IDEA, Chrome 版本 83.0.4103.61, JDK1.8, MySQL5.7, Navicat
任务内容	<p>1.主要技术点</p> <p>Spring Struts2 Hibernate Mysql</p> <p>对 Spring Struts2 Hibernate 进行整合, 利用 ssh 框架实现简单的学生信息管理系统。</p> <p>2.运行效果图</p>  <p>控制台输出截图:</p> <pre>Hibernate: select student0_.id as id1_0_, student0_.name as name2_0_, student0_.password as password3_0_, student0_.score as score4_0_ from student student0_ Student{id=1, name='小花', password='111111', score=86.0} Student{id=2, name='ss', password='1234456', score=85.0} Student{id=3, name='111', password='2222', score=85.0} Student{id=5, name='qw', password='123456', score=56.0} Student{id=6, name='www', password='12222', score=46.0} Student{id=7, name='hqy', password='1111111111', score=85.0} Student{id=8, name='eeee', password='ddddd', score=85.0} Student{id=14, name='dddd', password='22ddf', score=45.0} </pre> <p>添加学生的截图</p>  <pre>Hibernate: insert into student (name, password, score) values (?, ?, ?)</pre>

修改学生的信息:



姓名: ss
密码: 11111
分数: 19
修改

Hibernate:

```
update
    student
set
    name=?,
    password=?,
    score=?
where
    id=?
```

删除学生:

Hibernate:

```
delete
from
    student
where
    id=?
```

3.核心代码

在 pom.xml 文件中引入依赖

```
<dependencies>
    <!--
    https://mvnrepository.com/artifact/org.apache.struts/struts
    2-core -->
    <dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
        <version>2.5.16</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13</version>
        <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
```



```

</dependency>

<!--
https://mvnrepository.com/artifact/org.apache.tomcat.embed/
tomcat-embed-jasper -->
<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-jasper</artifactId>
  <version>9.0.33</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.0-b01</version>
</dependency>
<!-- https://mvnrepository.com/artifact/taglibs/standard
-->
<dependency>
  <groupId>taglibs</groupId>
  <artifactId>standard</artifactId>
  <version>1.1.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.mchange/c3p0
-->
<dependency>
  <groupId>com.mchange</groupId>
  <artifactId>c3p0</artifactId>
  <version>0.9.5.2</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-
core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.2.Final</version>
</dependency>
<!--
https://mvnrepository.com/artifact/mysql/mysql-connector-ja
va -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.15</version>
</dependency>

```

```

<!--
https://mvnrepository.com/artifact/org.springframework/spri
ng-core -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>5.2.1.RELEASE</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spri
ng-context -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.2.1.RELEASE</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spri
ng-aspects -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aspects</artifactId>
  <version>5.2.1.RELEASE</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spri
ng-orm -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-orm</artifactId>
  <version>5.2.1.RELEASE</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spri
ng-test -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>5.2.1.RELEASE</version>
  <scope>test</scope>
</dependency>
<!--
https://mvnrepository.com/artifact/org.springframework/spri
ng-web -->

```

```

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>5.2.1.RELEASE</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.apache.struts/struts
2-spring-plugin -->
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-spring-plugin</artifactId>
  <version>2.5.20</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.13.3</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.apache.logging.log4j
/log4j-api -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-api</artifactId>
  <version>2.13.3</version>
</dependency>
</dependencies>

```

在 MySQL 中国创建 student 表

```

SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS `student`;
CREATE TABLE `student` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NULL
  DEFAULT NULL,
  `password` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin
  NULL DEFAULT NULL,
  `score` double(255, 0) NULL DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 17 CHARACTER SET = utf8
  COLLATE = utf8_bin ROW_FORMAT = Dynamic;

SET FOREIGN_KEY_CHECKS = 1;

```

编写 jdbc.properties 用来放与数据库有关的信息

```
jdbc.driverClass =com.mysql.cj.jdbc.Driver
jdbc.url =
jdbc:mysql://127.0.0.1:3306/ssh?useUnicode=true&characterEn
coding=utf-8&useSSL=true&serverTimezone=UTC
jdbc.username = root
jdbc.password =12297804qa
```

编写实体类 Student

```
package com.chinasofti.ssh.entity;
import java.io.Serializable;
public class Student implements Serializable {
    private int id;
    private String name;
    private String password;
    private double score;
    public Student() {
    }
    public Student(int id, String name, String password, double
score) {
        this.id = id;
        this.name = name;
        this.password = password;
        this.score = score;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public double getScore() {
        return score;
    }
}
```

```

    }

    public void setScore(double score) {
        this.score = score;
    }

    @Override
    public String toString() {
        return "Student{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", password='" + password + '\'' +
            ", score=" + score +
            '}';
    }
}

```

配置数据库与实体类的映射文件 Student.hbm.xml

```

<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.chinasofti.ssh.entity.Student"
        table="student">
        <!-- 设置主键-->
        <id name="id" column="id">
            <!-- 按照数据库规则设置-->
            <generator class="native"></generator>
        </id>
        <!-- 设置其他数据 java 实体属性名 数据库列名 以及类型-->
        <property name="name" column="name" type="string"/>
        <property name="password" column="password"
            type="string"/>
        <property name="score" column="score" type="double"/>
    </class>
</hibernate-mapping>

```

修改 web.xml 配置 spring 加载和 struts 解析器以及过滤器

```

<!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application
    2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd" >
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    >

```

```

<!--配置监听器-->
<listener>
<listener-class>org.springframework.web.context.ContextLoad
erListener</listener-class>
</listener>
<!--配置 spring 加载-->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
</context-param>
<!--配置 struts 解析器-->
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.filter.StrutsPr
epareAndExecuteFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
</web-app>

```

配置 applicationContext.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/b
eans
http://www.springframework.org/schema/beans/spring-beans-2.
5.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-contex
t-2.5.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xs
d
    http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.xsd"
>
    <!--加载 jdbc.properties 资源-->
    <context:property-placeholder

```

```

location="classpath:jdbc.properties"/>

<!-- 配置 c3p0 连接池 -->
<bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass"
value="${jdbc.driverClass}"/>
    <property name="jdbcUrl" value="${jdbc.url}"/>
    <property name="user" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>
</bean>

<!-- 获取配置对象-》 sessionFactory-》 session-》 事物 交给 spring
创建 sessionFactory-->
<bean name="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
    <!-- 注入连接池 -->
    <property name="dataSource" ref="dataSource"/>
    <!-- 配置连接属性-->
    <property name="hibernateProperties">
        <props>
            <prop
key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect
</prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.format_sql">true</prop>
            <prop
key="hibernate.hbm2ddl.auto">update</prop>
        </props>
    </property>
    <!-- 加载映射资源，之前 hibernate.cfg.xml mapping 标记加载资源-->
    <property name="mappingResources">
        <list>
            <value>Student.hbm.xml</value><!-- 可以添加多个映射对象-->
        </list>
    </property>
</bean>

<!--
    事务管理器
-->
<bean id="transactionManager"
class="org.springframework.orm.hibernate5.HibernateTransactionManager">

```

```

        <property name="sessionFactory">
            <ref bean="sessionFactory"/>
        </property>
    </bean>
    <tx:advice id="tx"
transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="insert*" read-only="false"/>
            <tx:method name="delete*" read-only="false"/>
            <tx:method name="update*" read-only="false"/>
        </tx:attributes>
    </tx:advice>
    <aop:config>
        <aop:pointcut
            expression="execution(*
com.chinasofti.ssh.dao.impl.*(..))"
            id="perform"/>
        <aop:advisor advice-ref="tx" pointcut-ref="perform"/>
    </aop:config>

<!--配置 dao 层 javaBean-->
    <bean id="studentDao"
class="com.chinasofti.ssh.dao.impl.StudentDaoImpl">
        <property name="sessionFactory">
            <ref bean="sessionFactory"></ref>
        </property>
    </bean>
    <bean id="studentService"
class="com.chinasofti.ssh.service.impl.StudentServiceImpl">
        <property name="studentDao">
            <ref bean="studentDao"></ref>
        </property>
    </bean>
    <!--配置 action-->
    <bean id="studentAction"
class="com.chinasofti.ssh.action.StudentAction">
        <property name="studentService">
            <ref bean="studentService"/>
        </property>
    </bean>
</beans>

```

配置 struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```



```

<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts
Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>
    <constant name="struts.i18n.encoding" value="UTF-8" />

    <package name="student" namespace="/student"
extends="struts-default">
        <!--ssh 整合 class 指向 spring 配置 action 就可以了-->
        <action name="del" class="studentAction"
method="del">
            <result name="toList"
type="redirect">getall</result>
        </action>
        <action name="update" class="studentAction"
method="update">
            <result name="toList"
type="redirect">getall</result>
        </action>
        <action name="getById" class="studentAction"
method="getById">
            <result name="update">/update.jsp</result>
        </action>
        <action name="addStudent" class="studentAction"
method="addStudent">
            <result name="toList"
type="redirect">getall</result>
        </action>
        <action name="getall" class="studentAction"
method="getall">
            <result name="list">/list.jsp</result>
        </action>
    </package>
</struts>

```

编写数据库层代码 StudentDaoImpl 采取面向接口编程的方法

```

package com.chinasofti.ssh.dao.impl;

import com.chinasofti.ssh.dao.StudentDao;
import com.chinasofti.ssh.entity.Student;
import org.hibernate.query.Query;
import
org.springframework.orm.hibernate5.support.HibernateDaoSupp

```

```

    ort;
    import java.io.Serializable;
    import java.sql.SQLException;
    import java.util.List;
    /*要去使用 hibernate 进行数据库操作 */
    public class StudentDaoImpl extends HibernateDaoSupport
    implements StudentDao {
        public void insertStudent(Student stu) {
            //进行数据库添加
            this.getHibernateTemplate().save(stu);
        }
        public List<Student> list() {
            Query
            query=getSessionFactory().openSession().createQuery("from
            Student");
            return query.list();
        }
        public void deleteStudent(int id){

            Student stu= queryById(id);
            this.getHibernateTemplate().delete(stu);
        }
        public Student queryById(int id){
            return
            this.getHibernateTemplate().get(Student.class,id);
        }
        public void updateStudent(Student stu){
            this.getHibernateTemplate().update(stu);
        }
    }

```

编写业务逻辑层代码 StudentServiceImpl，同样采用面向接口编程

```

package com.chinasofti.ssh.service.impl;
import com.chinasofti.ssh.dao.StudentDao;
import com.chinasofti.ssh.entity.Student;
import com.chinasofti.ssh.service.StudentService;
import java.util.List;
public class StudentServiceImpl implements StudentService {
    //设置一个属性 dao 层
    private StudentDao studentDao;
    public void setStudentDao(StudentDao studentDao) {
        this.studentDao = studentDao;
    }
    @Override
    public boolean isExit(int id) {

```

```

        Student stu=studentDao.queryById(id);
        if(stu==null)
            return false;
        else
            return true;
    }

    @Override
    public boolean addStudent(Student stu) {
        //判断当前这个人是否已经添加了?
        this.studentDao.insertStudent(stu);
        return true;
    }

    @Override
    public boolean deleteStudent(int id) {
        this.studentDao.deleteStudent(id);
        return true;
    }

    @Override
    public boolean updateStudent(Student stu) {
        if(!isExit(stu.getId()))
            return false;
        else {
            this.studentDao.updateStudent(stu);
            return true;
        }
    }

    @Override
    public List<Student> queryAllStudent() {
        return this.studentDao.list();
    }

    @Override
    public Student queryStudentById(int id) {
        return this.studentDao.queryById(id);
    }
}

```

编写 action 的代码 StudentAction

```

package com.chinasofti.ssh.action;
import com.chinasofti.ssh.entity.Student;
import com.chinasofti.ssh.service.StudentService;
import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;
import jdk.nashorn.internal.ir.ReturnNode;
import org.apache.struts2.ServletActionContext;

```

```

import java.util.List;

public class StudentAction extends ActionSupport {
    private Student student;
    private StudentService studentService;
    public void setStudentService(StudentService
studentService) {
        this.studentService = studentService;
    }
    public Student getStudent() {
        return student;
    }
    public void setStudent(Student student) {
        this.student = student;
    }
    /*设置多个方法进行绑定*/
    public String addStudent(){
        //调用 service 进行数据添加
        this.studentService.addStudent(this.student);
        return "toList";
    }
    public String getall(){
        List<Student> list=studentService.queryAllStudent();
        for(Student stu:list){
            System.out.println(stu);
        }

        ServletActionContext.getRequest().setAttribute("list",list)
;

        return "list";
    }
    public String getById(){
        int id=student.getId();
        student=studentService.queryStudentById(id);
        return "toList";
    }
    public String update(){
        System.out.println("update"+student);
        studentService.updateStudent(student);
        return "success";
    }
    public String del(){
        System.out.println("del"+student);
        studentService.deleteStudent(student.getId());
        return "toList";
    }
}

```

```
}  
}
```

编写前端页面 list.jsp

```
<%--  
    Created by IntelliJ IDEA.  
    User: qa  
    Date: 2020/6/25  
    Time: 21:41  
    To change this template use File | Settings | File Templates.  
--%>  
<%@ page language="java" contentType="text/html;  
    charset=UTF-8"  
    pageEncoding="UTF-8"%>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"  
    prefix="c" %>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
    <base href="${pageContext.request.contextPath}/">  
    <meta http-equiv="Content-Type" content="text/html;  
    charset=UTF-8">  
    <title>Insert title here</title>  
    <script type="text/javascript">  
        function update(id) {  
            location.href="/student/getById?id="+id;  
        }  
        function del(id) {  
            alert(id);  
            //location.href="/student/del?id="+id;  
        }  
    </script>  
</head>  
<body>  
<table>  
    <tr>  
        <th>序号</th>  
        <th>姓名</th>  
        <th>分数</th>  
        <th>操作</th>  
    </tr>  
    <c:forEach items="${list}" var="s">  
        <tr>  
            <td>${s.id}</td>
```

```

        <td>${s.name}</td>
        <td>${s.score}</td>
        <td>
            <button onclick="update(${s.id})">修改</button>
            <button onclick="del(${s.id})">删除</button>
        </td>
    </tr>
</c:forEach>
</table>
</body>
</html>

```

update.jsp

```

<%@ page language="java" contentType="text/html;
charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<base href="${pageContext.request.contextPath}/">
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="student/update" method="post">
<input type="text" name="id" value="8"/>
    姓名:<input type="text" name="name"
value="${student.name}"/><br/>
    密码:<input type="text" name="password"
value="${student.password}"/><br/>
    分数:<input type="text" name="age"
value="${student.score}"/><br/>
    <input type="submit" value="修改"/><br/>
</form>
</body>
</html>

```

addStudent.jsp

```

<%--
    Created by IntelliJ IDEA.
    User: qa
    Date: 2020/6/21
    Time: 15:01
--%>

```

	<div>To change this template use File Settings File Templates.</div> <div>--></div> <div><%@ page contentType="text/html; charset=UTF-8"</div> <div>language="java" %></div> <div><html></div> <div><head></div> <div><title>Title</title></div> <div></head></div> <div><body></div> <div><form action="student/addStudent" method="post"></div> <div>用户名: <input type="text" name="student.name"</div> <div>placeholder="请输入用户名">
</div> <div>密码: <input type="password" name="student.password"</div> <div>placeholder="请输入密码">
</div> <div>得分<input type="text" name="student.score" placeholder="</div> <div>请输入得分">
</div> <div><input type="submit" value="提交"></div> <div></form></div> <div></body></div> <div></html></div>
任务总结	<div>1.通过此任务的收获</div> <div>首先通过此次任务了解并且掌握了如何使用 spring 框架来进行开发，对 spring 框架有了一个简单的入门，学会了如何利用 spring 框架进行编程，利用 spring 框架可以不需要创建对象，而是通过在配置文件中配置产生对象或者是通过工厂模式来产生对象，相对于以往的手动 new 对象十分方便简单，减少了很多代码的编写。Spring 是一个轻量级框架，对于 spring 中重要的两部分内容面向切面编程以及控制反转有了一个简单的理解，但是在后期还有待学习和研究才能透彻理解其中的内涵。</div> <div>在学习完 struts2、hibernate、spring 后，对这三部分进行了一个整合，在利用 ssh 开发学生管理系统后，收获颇多。掌握了如何利用 ssh 框架进行开发的一个基本的流程，掌握了每一部分的使用法。利用 ssh 框架进行开发，相较于以前利用 jdbc，创建对象，获取参数的方式更加的简便。通过此次任务深刻感受到了利用框架进行开发的便利，利用框架可以极大的减少程序员开发的代码量，并且利用框架会使得整个程序结构更加的清晰。</div> <div>2.此任务的可扩展性</div> <div>对于此任务还有很多可以扩展的地方，首先是前端的扩展，例如对界面的优化，还有就是可以使用 bootstrap 框架或者 vue 框架进行前端界面的 UI 设计，把界面设计的更加美观以及用户友好性更强。</div> <div>对于本项目的功能还有很多可以完善的地方，例如增加注册功能，增加验证码功能，增加分页查询功能等。</div> <div>对于 ssh 框架的使用目前只是一个简单的入门，其中的某些用法还有待接下来的进一步学习与研究。</div>

3. 课程总结

通过这两天和田老师的学习，感触很多。虽然只和田老师学习了两天，但是收获满满。由于之前没接触过框架，之前开发 web 应用的时候只是用一些基础的 jdbc、jsp、servlet 来开发。在这两天学习了 ssh 框架后，觉得利用框架开发是多么的方便快捷，深刻感受到框架的优势，使用框架使得开发 web 应用变得简单起来。使用 hibernate 对数据库进行操作相对于以往的编写 jdbc 的一系列流程简单了很多，只需要写好配置文件就可以很方便的对数据库进行操作。使用 struts 则是方便了前后端的数据交互，控制业务跳转。而 spring 则是用来管理 struts 和 hibernate，不需要像以往一样 new 对象。

虽然只是对框架有了一个简单的入门，但是还是学到了很多。ssh 是 struts+spring+hibernate 的一个集成框架。使用 ssh 框架可以帮助开发人员快速搭建出结构清晰的 web 应用程序，并且使用框架开发的 web 程序维护起来也是十分的方便。Struts 作为系统的整体基础架构，负责 MVC 的分离，在 Struts 框架的模型部分，控制业务跳转，利用 Hibernate 框架对持久层提供支持，Spring 做管理，管理 struts 和 hibernate。

最后，在此感谢田老师的授课，虽然只有两天时间，但是田老师讲课风格幽默，授课内容清晰，使得我学到了很多，感谢老师。