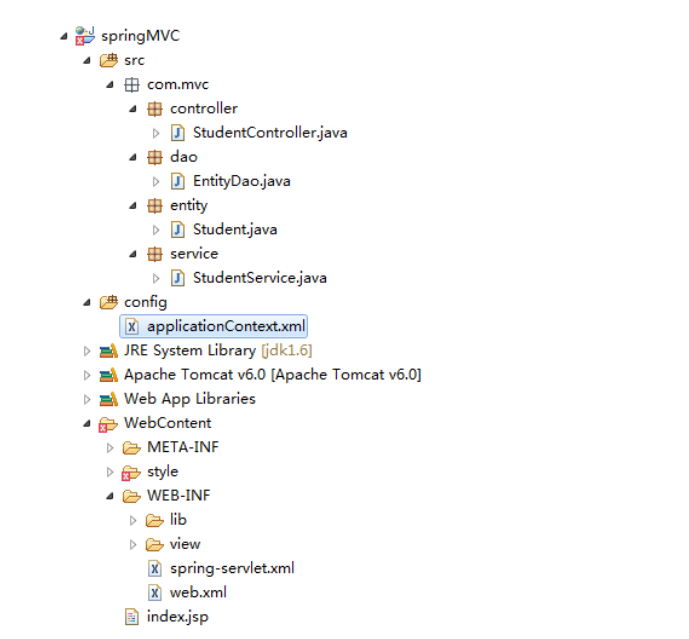
SpringMVC的搭建

一直以来接触到的都是SSH的框架，形成了MVC模式，本来并没有想着去弄另一个MVC模式，但现在springMVC模式很热，所以我也学习一下，首先我声明一下，这个框架我也是在网上找了一些资料才完成的，源文件等也是利用的网上的现成的，但是有对其进行修改。下面来详细的说一说这个模式的搭建。首先在spring中是以controller来作为控制器（相当于SSH中的action），其他的和SSH框架没有区别。因为Spring是基于注解的，所以在整个的模式中都是采用注解的方式来处理，这个项目是用springMVC+hibernate一起来搭建的。这个项目的搭建我花了很久的时间，也弄了蛮久才成功，希望日后能更加完善！理解更加的深入。

一：整体框架的结构图以及所需的jar包。这里spring是3.0.1，hibernate是用的3.6，数据库是用的mysql 5.6 ，前提工作是要建立好一个数据库，我这里是名为springmvc的数据库来进行操作，这里是采用的hibernate自动更新的方式，所以可以不需要建表只需要建立起数据库就好。



二：开始搭建环境。

1， 首先把上面所需的包添加进来后，我们要在/WEB-INF目录下的

web.xml里面添加spring的监听器，以及相关的配置。源码如下： <?xml version="1.0" encoding="UTF-8"?>

<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">

<display-name>s3h3</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext\*.xml</param-value> </context-param>

xmlns:context="http://www.springframework.org/schema/context" xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:p="http://www.springframework.org/schema/p" xmlns:tx="http://www.springframework.org/schema/tx" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-3.0.xsd http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">

<context:annotation-config />

<context:component-scan base-package="com.mvc" /> <!-- 自动扫描所有注解该路径 -->

<mvc:annotation-driven></mvc:annotation-driven> <aop:aspectj-autoproxy/> <bean id="sessionFactory"

class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">

<property name="dataSource" ref="dataSource" /> <property name="hibernateProperties"> <props> <prop

key="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</prop>

<prop key="hibernate.hbm2ddl.auto">update</prop> <prop key="show\_sql">true</prop> <prop key="format\_sql">true</prop> </props> </property>

<property name="packagesToScan"> <list>

<value>com.mvc.entity</value><!-- 扫描实体类，也就是平时所说的model -->

</list> </property> </bean>

<bean id="transactionManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory" ref="sessionFactory" /> </bean>

<bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSource"> <property name="driverClassName" value="com.mysql.jdbc.Driver" /> <property name="url"

value="jdbc:mysql://localhost:3306/springmvc" /> <property name="username" value="root" /> <property name="password" value="8425" /> </bean>

<tx:annotation-driven transaction-manager="transactionManager" />

</beans>

3， 完成上面的工作后，我们可以来进行下面的spring-servlet.xml文

件的编写了，这个文件是放在/WEB-INF目录下，应该他也可以放到config目录里，这里我就不做这一步工作了，具体的源码如下： <?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-3.0.xsd http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-3.0.xsd http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:annotation-config />

<!-- 把标记了@Controller注解的类转换为bean -->

<context:component-scan base-package="com.mvc.controller" /> <!-- 启动Spring MVC的注解功能，完成请求和注解POJO的映射 --> <bean

class="org.springframework.web.servlet.mvc.annotation.AnnotationMeth

odHandlerAdapter" />

<!-- 对模型视图名称的解析，即在模型视图名称添加前后缀 --> <bean

class="org.springframework.web.servlet.view.InternalResourceViewResolver"

p:prefix="/WEB-INF/view/" p:suffix=".jsp" />

<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver"

p:defaultEncoding="utf-8" />

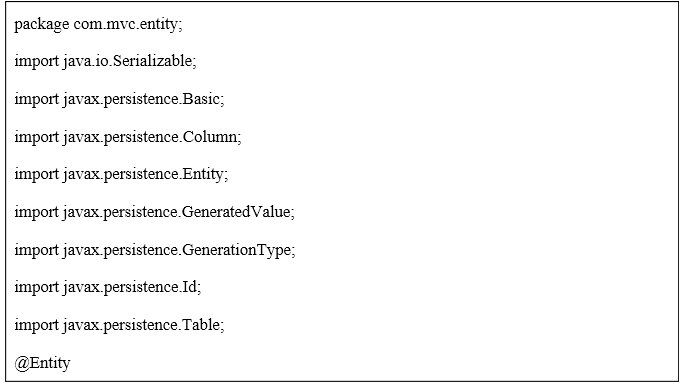
</beans>

4， 配置文件完成后，最好进行一下测试，这样子保证环境没有出错。

如果测试通过了，就可以进行下面的工作啦！ 三：开始编写代码。

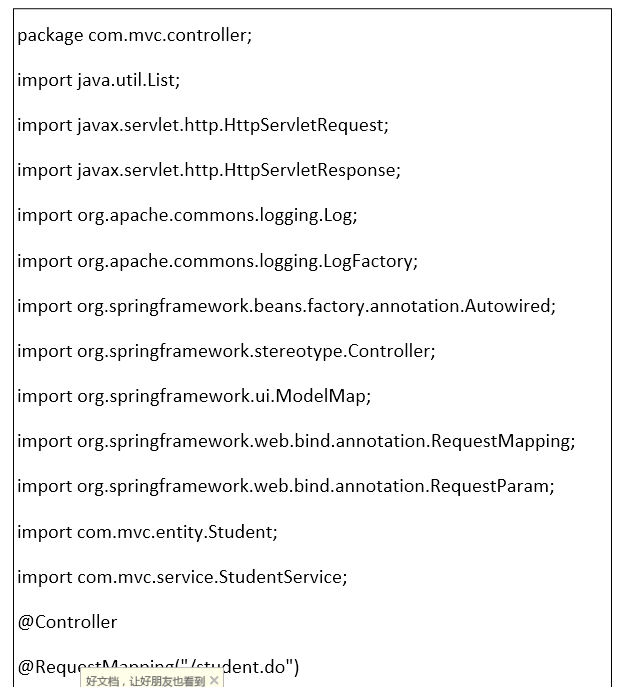
1， 首先要建立需要用到的实体，这里我是采用的一个对学生的简单

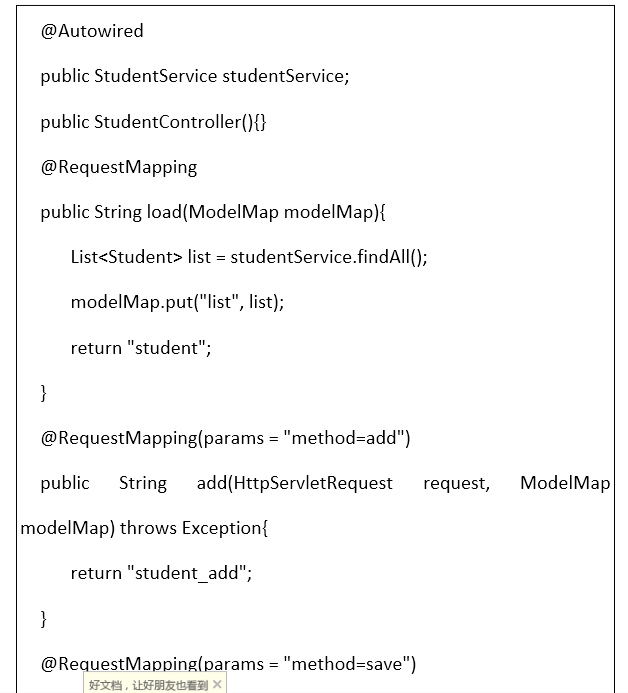
增删改查的操作来进行测试案例的。项目中采用注解的方式来进行映射。实体是放在com.mvc.entity包里面，名为Student.java，源码如下：





2、来添加控制层代码即controller层代码，这里是放在了com.mvc.controller包里，我这里建立的是StudentController.java文件，源码如下：





3. 接下来就可以来按照IDE工具里面所提示的进行业务层的编写，这里只用了两层，即dao层和service层来实现业务的控制。用dao层来操作数据库。具体的是EntityDao.java，这里的命名不太规范，最规范的命名应当为StudentDao.java，这里我就不做修改了，他的具体源码如下：

package com.mvc.dao;

import java.util.List;

import org.hibernate.SessionFactory;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Repository; import com.mvc.entity.Student;

@Repository("entityDao") public class EntityDao {

@Autowired

private SessionFactory sessionFactory;

public void save(Student s){

sessionFactory.getCurrentSession().save(s); }

@SuppressWarnings("unchecked") public List<Student> findAll(){ return

sessionFactory.getCurrentSession().createQuery("from Student").list();

4.service层来调用dao层的方法来实现业务。这里主要是放在com.mvc.service包里，建立了StudentService.java方法。具体的源码如下：

package com.mvc.service; import java.util.List;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional; import com.mvc.dao.EntityDao; import com.mvc.entity.Student; @Service("studentService") @Transactional

public class StudentService { @Autowired

public EntityDao entityDao; public void save(Student st){ entityDao.save(st);

}

public void delete(Student obj){ entityDao.delete(obj); }

public List<Student> findAll() {

return entityDao.findAll();

}

public void update(Student model){ entityDao.update(model);

} }

四：最后我们只需要进行表现层的编写，这里页面的代码我是放在了/WEB-INF/view目录下，主要有student\_add.jsp和student.jsp页面，我还没有完成编辑页面的编写，可以由大家自己去实现。