**实验九 SpringMVC项目搭建**

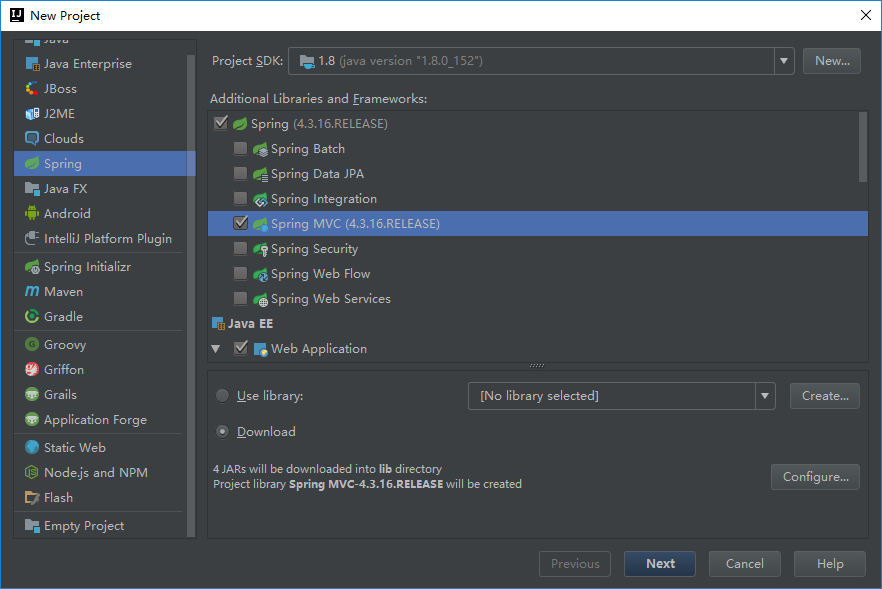
### 一、实验目的

使用SpringMVC搭建项目框架。

### 二、实验项目内容

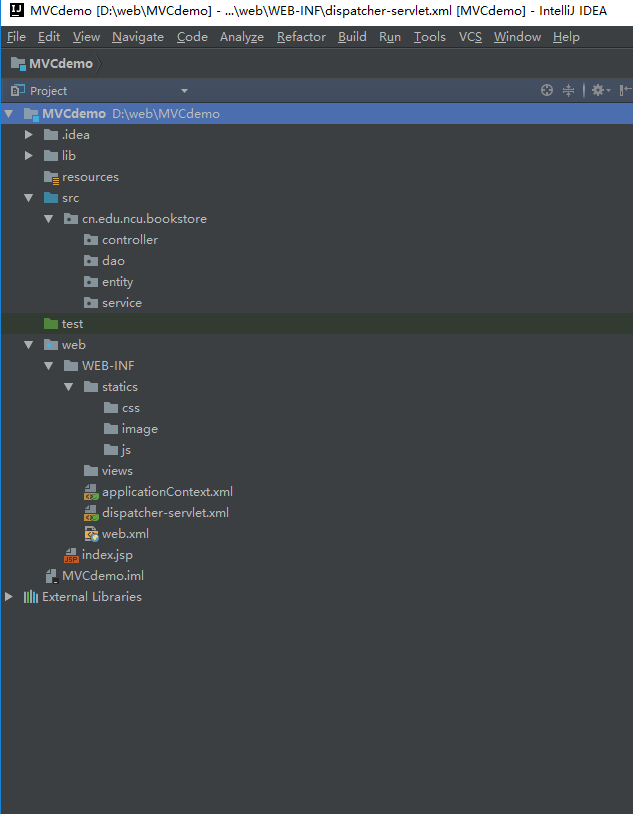
* **1、SpringMVC安装**

以idea为例



选择Spring以及SpringMVC

完善项目目录结构，创建controller、dao、entity、service等目录



还需要完善项目的目录结构

1、src下的项目分层结构

2、WEB-INF中view的文件夹

3、资源目录

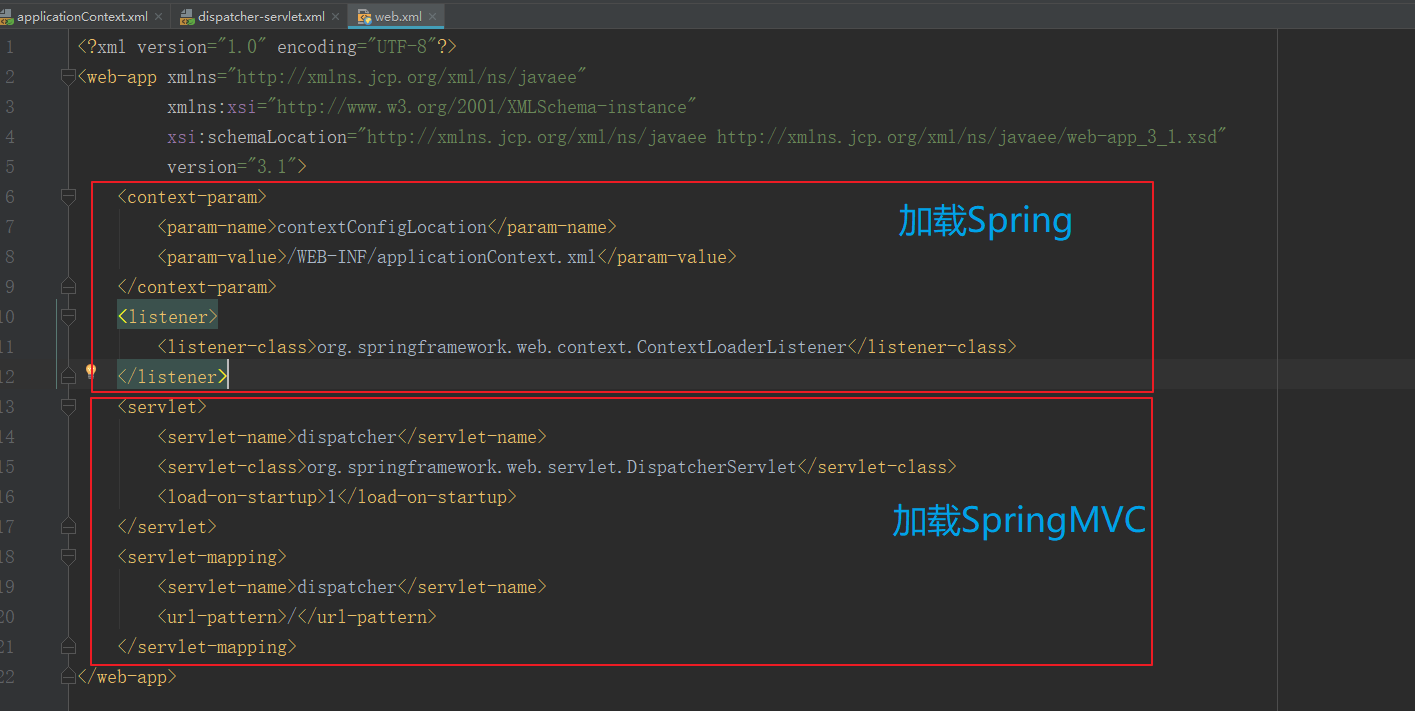
4、测试代码目录

5、静态资源(css、js、image)

* 2、修改配置文件

web.xml

加载**Spring**和**SpringMVC**



Springmvc 没有指定xml路径，默认会去查找xxx-servlet.xml的文件，如果需要指定可以添加<init-param>进行相应的指定。

同时可以设置多个dispathcher来处理不同url的请求，来区分不同的应用场景。

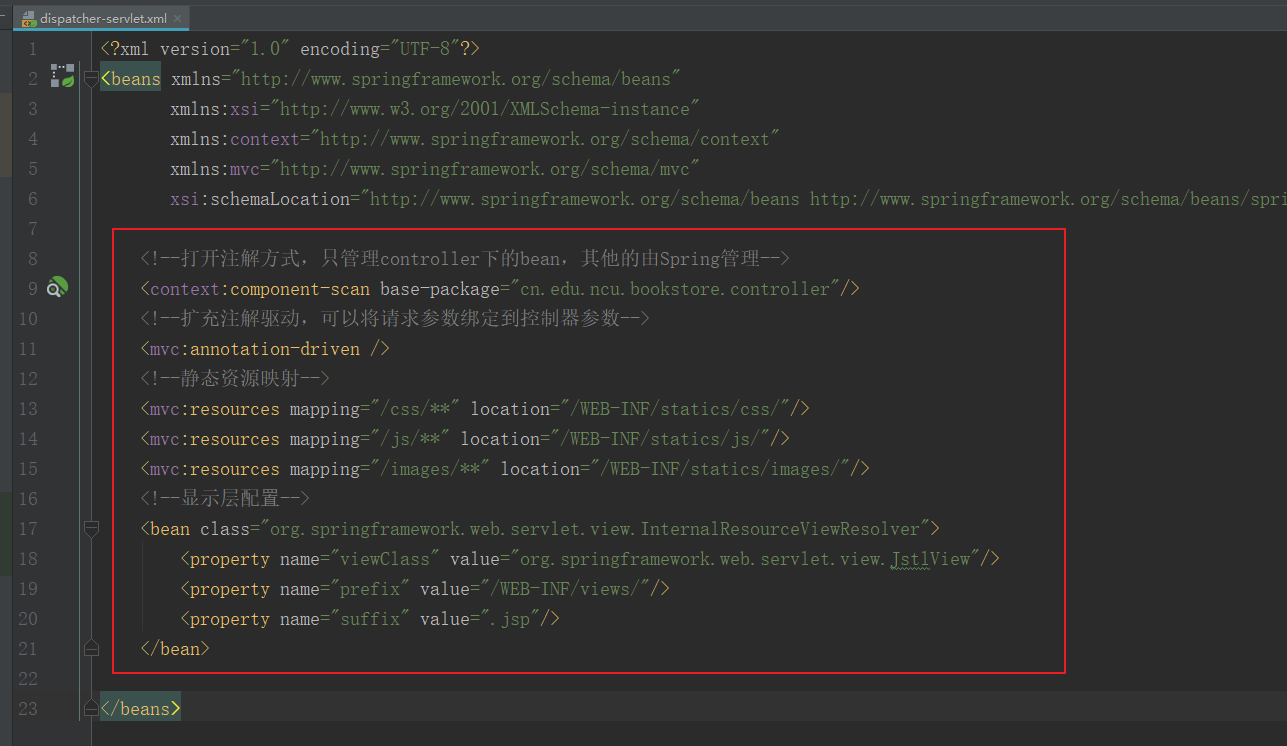
dispatcher-servlet.xml

添加依赖注入的注解方式（通过指定目录实现）

扩充注解驱动，可以将请求参数绑定到控制器参数

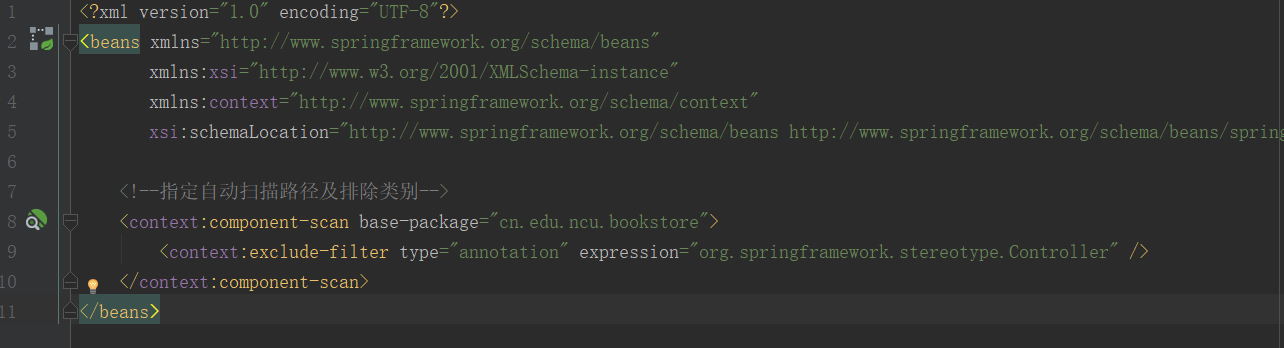
静态资源配置

显示层配置



applicationContext.xml

设置自动扫描路径

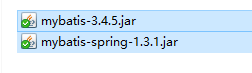


* 3、配置MyBatis环境

首先需要添加jar，可以通过官网下载，也可以通过maven配置；

需要两个jar包

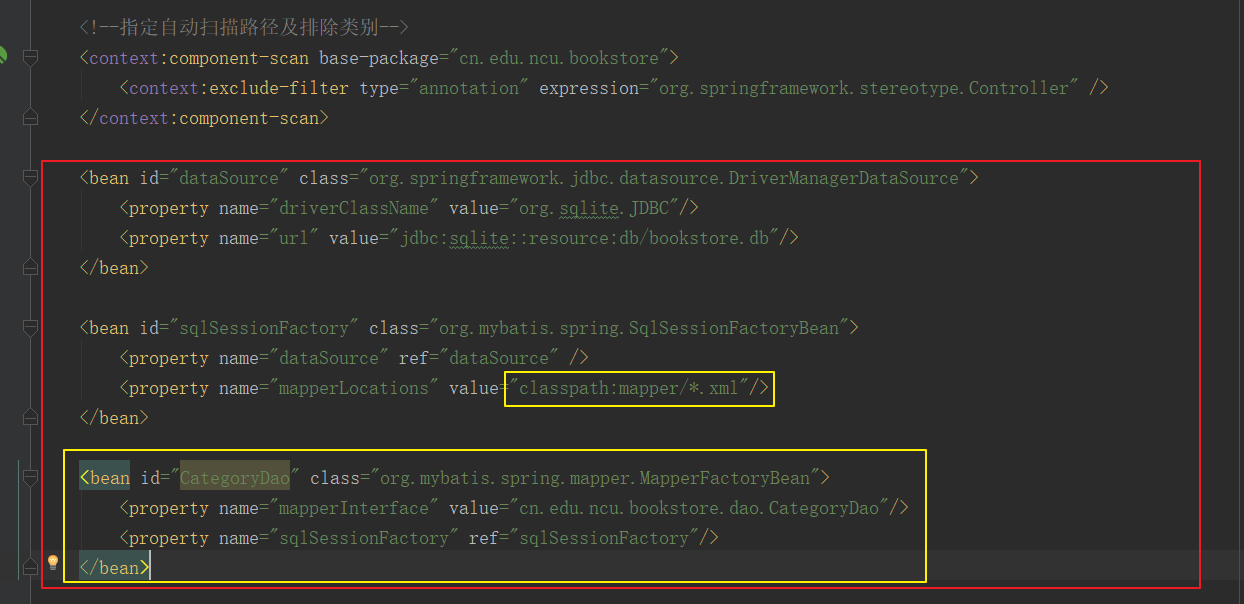
* + mybatis.jar
  + mybatis-spring.jar



配置SqlSessionFactory

* + 整合后，**不需要**单独的mybatis配置文件，全部的配置内容都可以在spring的上下文当中进行。
  + myBatis框架主要是围绕SqlSessionFactory进行的
    - 定义一个configuration对象，其中包括数据源、mapper文件、事务和影响数据库行为属性设置。
    - 通过配置对象，创建一个SqlSessionFactoryBuilder对象
    - 通过SqlSessionFactoryBuilder 获得SqlSessionFactory实例

SqlSessionFactoryBuilder实例可以获得操作数据的SqlSession实例，通过该实例对数据库进行操作



**其中 “dataSource”中dirverClassName及url的值设置为自己的环境；**

黄色部分需要先建立相关的目录和文件，否则配置文件会有错误提示。

在定义SqlSessionFactoryBean的时候，dataSource属性是必须指定的，它表示用于连接数据库的数据源。

最下面的框中，设置Mapper（CategoryDao）对象对应的MapperFactoryBean。

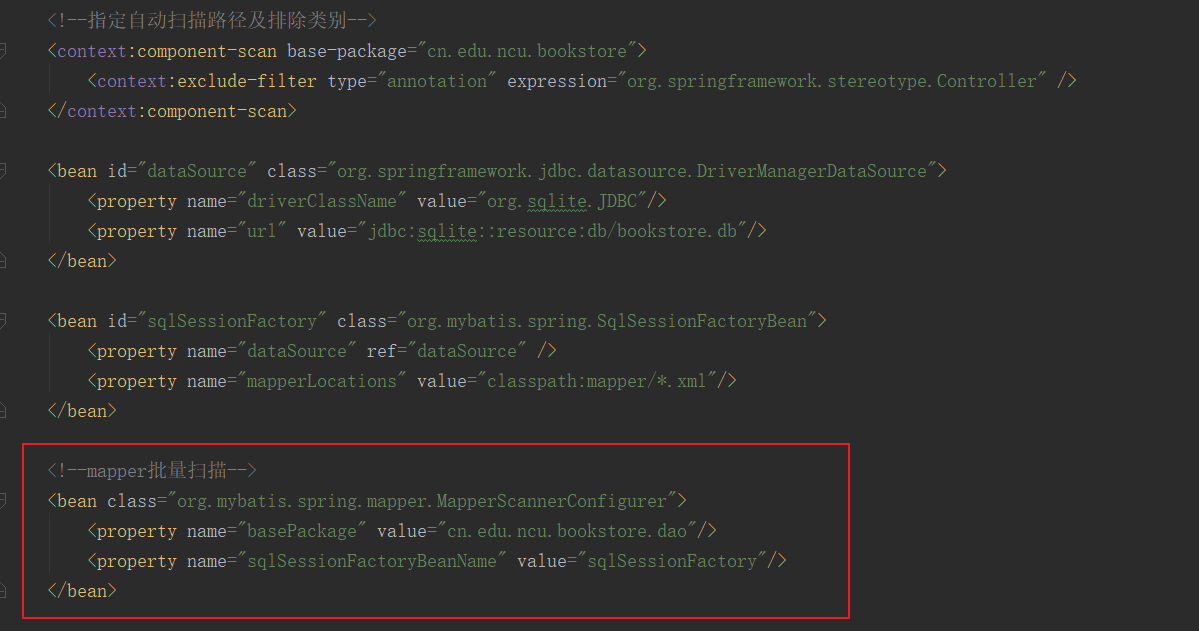
通过MapperFactoryBean可以获取到我们想要的Mapper对象。

(MapperFactoryBean实现了Spring的FactoryBean接口，所以MapperFactoryBean是通过FactoryBean接口中定义的getObject方法来获取对应的Mapper对象的。)

这部分需要两个参数，一个是mapper接口，一个是sqlSessionFactory

【**批量扫描方式**】

之前的方法每一个dao都需要写一段映射，太繁琐当dao很多的时候相当麻烦，所以可以考虑使用批量扫描的办法

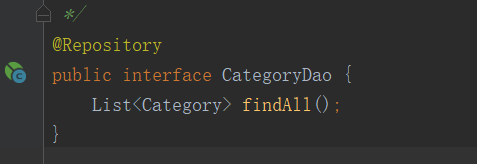


basePackage是用来指定Mapper接口文件所在的包，在这个包或其所有子包下面的Mapper接口都将被搜索到。多个包之间可以使用逗号或者分号进行分隔。

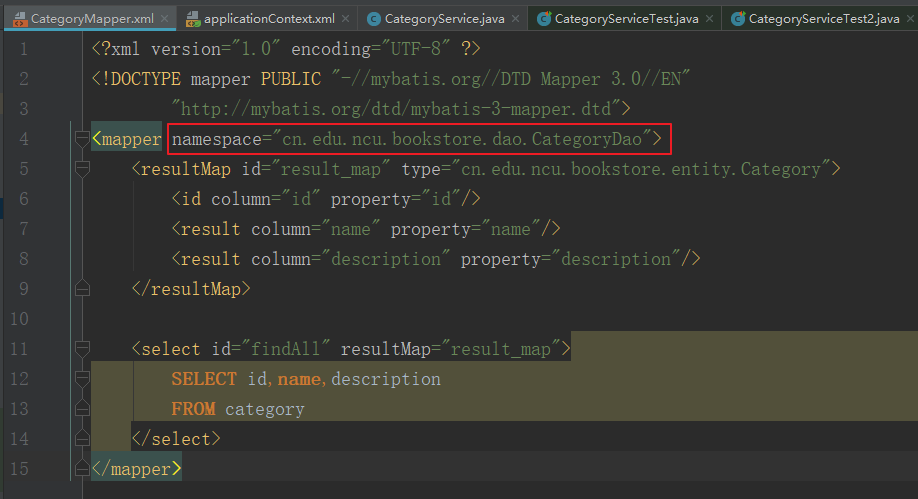
定义好相应Mapper接口对应的MapperFactoryBean之后，就可以把对应的Mapper接口注入到由Spring管理的bean对象中了，比如Service bean对象

* 4、Mapper（Dao）开发

添加Dao接口



使用myBatis不需要实现接口，需要编写相应的xml文件



1、namespace要正确，否则无法对应到相关的mapper

2、resultMap为返回的对象，其中property的值需要和指定的entity一致

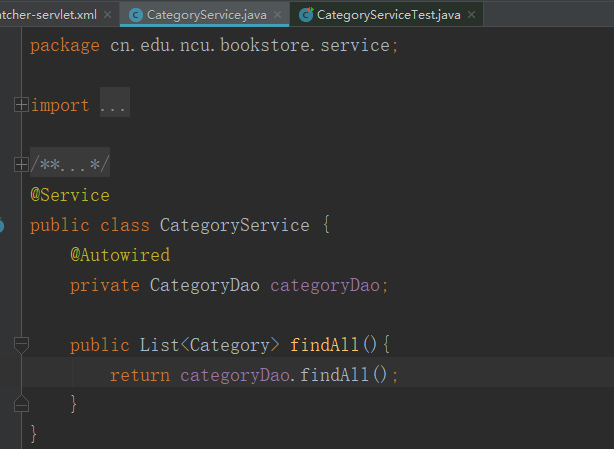
3、如需要传入变量则使用#{parm}的方式

SELECT id,name,description  
FROM category

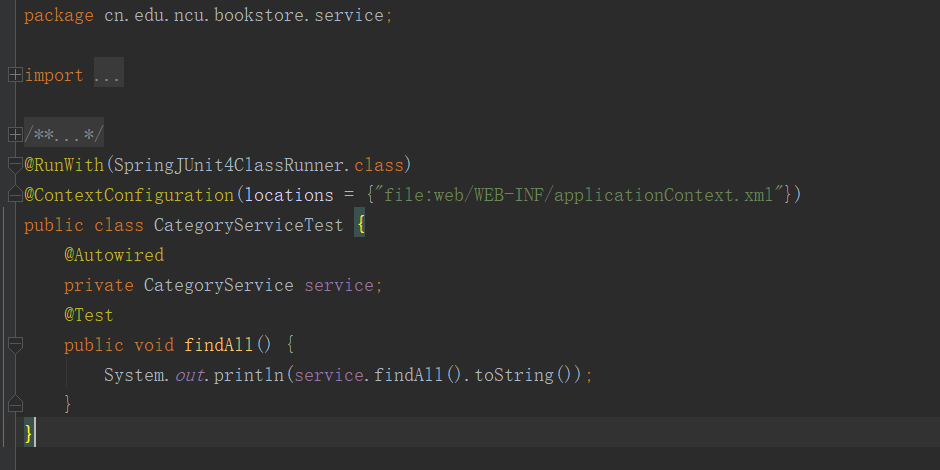
WHERE id = #{id}

* 5、Service开发

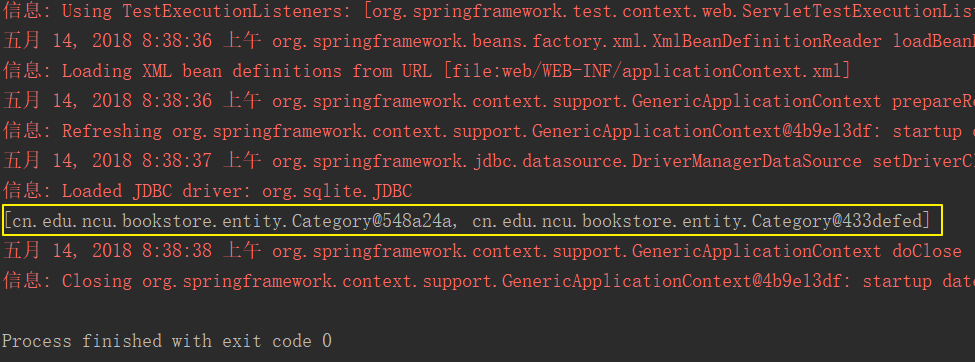
新建CategoryService类，将Mapper(CategoryDao）接口注入到由Spring管理的bean对象中



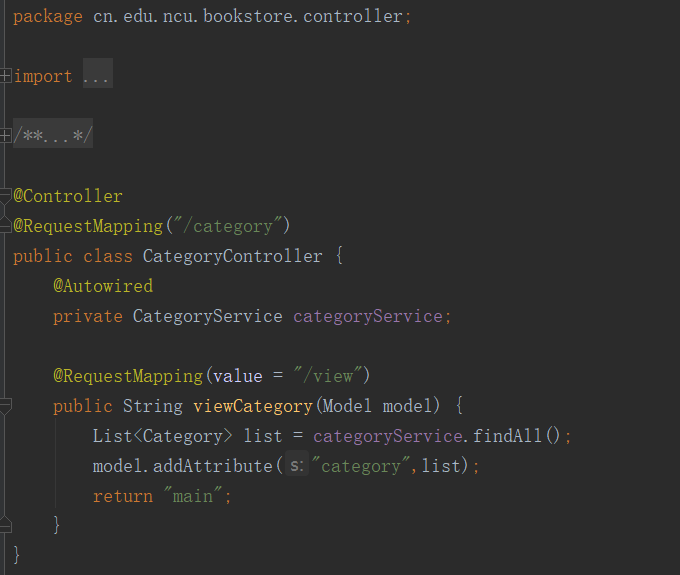
* 6、Service单元测试



因为applicationContext.xml放在了WEB-INF目录下，所有不能使用classpath的方式来定位，需通过file的方式定位。



* 7、Controller开发



增加注解**@Controller**交由SpringMVC管理

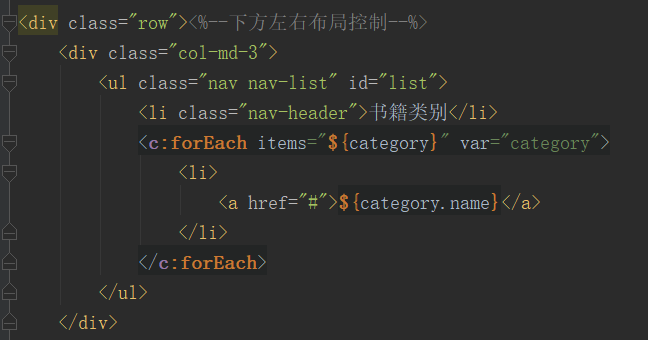
@RequestMapping注解在类上，表明该路径下所有访问由此类处理

注解在方法上的和注解在类上的路径组合**/category/view**

注：return “main” 在前面的MVC配置中设置了前缀和后缀。

* 8、view开发

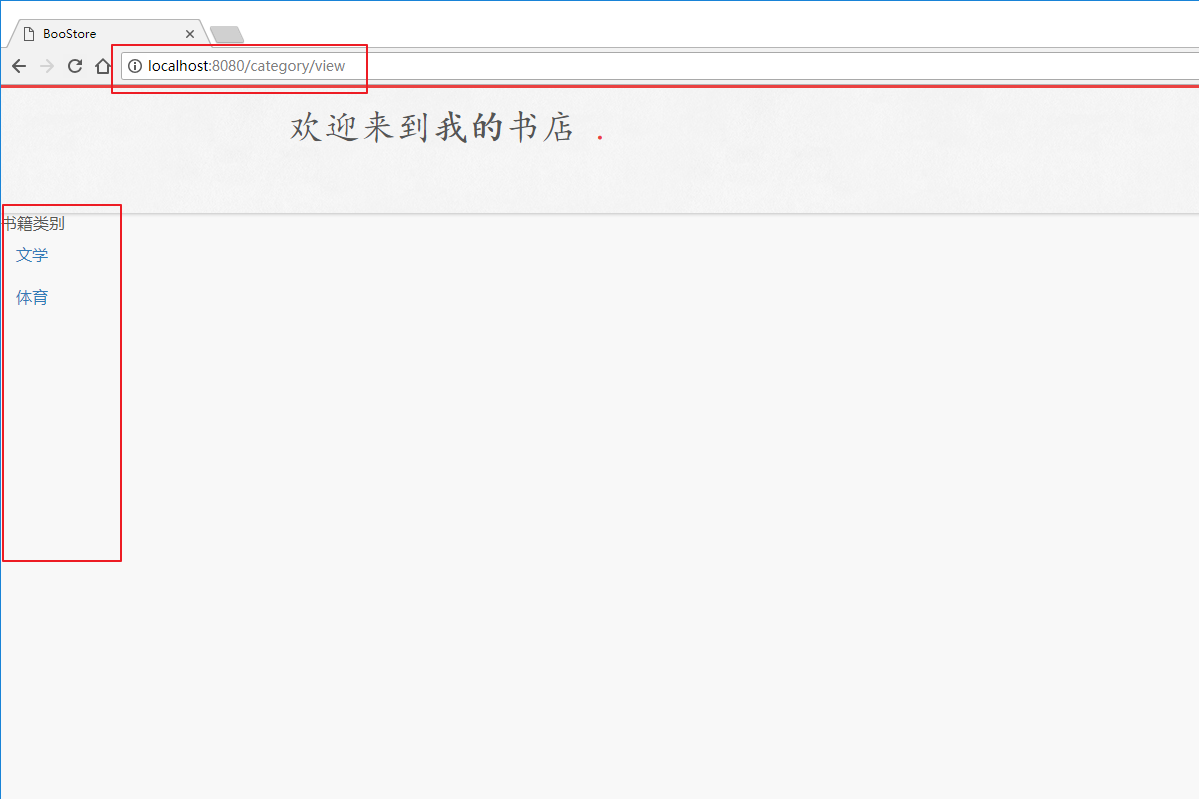
修改（或新建）main.jsp，修改类别显示部分代码



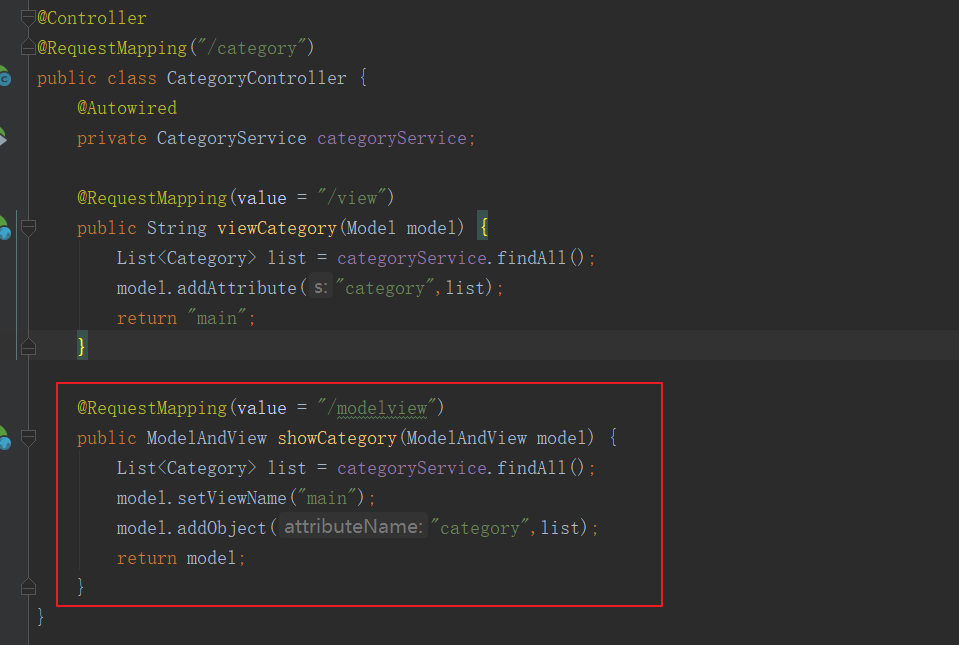
使用JSTL处理，呈现数据显示。注意在jsp文件中引入相应的包

<%@ **taglib prefix**="**c**" **uri**="**http://java.sun.com/jsp/jstl/core**" %>  
<%@ **taglib prefix**="**fn**" **uri**="**http://java.sun.com/jsp/jstl/functions**" %>

访问相应路径：



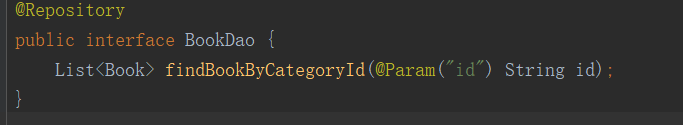
* 9、修改Controller，使用modelandview



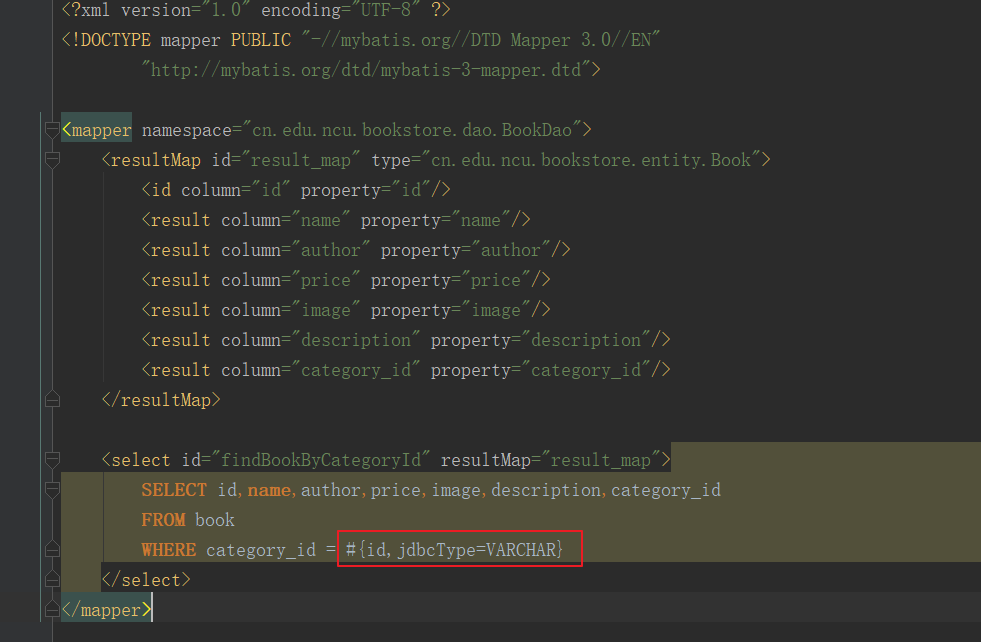
访问相应路径

* 10、使用参数的查询

1. 新建Book实体
2. 新建Mapper(BookDao的接口)
   1. 使用@Param注解



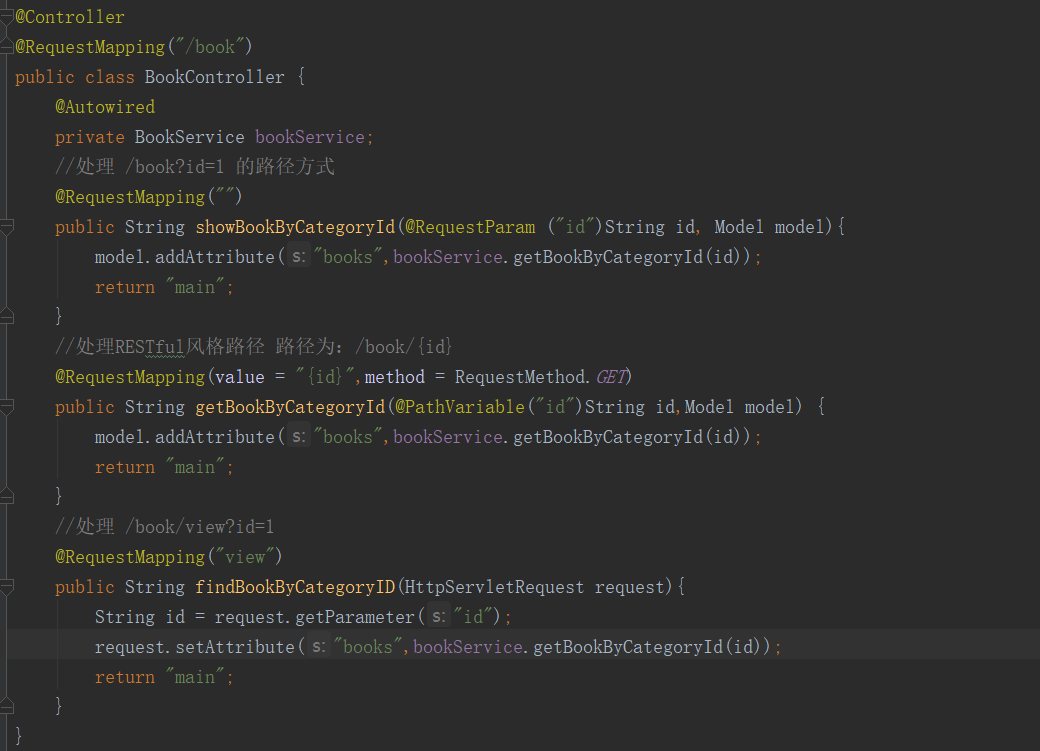
1. 新建Mapper XML文件



1. 新建BookService

自行完成

1. 新建BookController



前面两个是不同路径风格的处理方式

最后一个是通过传统的ServletRequest方式获取传入值。

http://localhost:8080/book?id=1

http://localhost:8080/book/1

http://localhost:8080/book/view?id=1

* 11、使用json格式传递数据

使用@ ResponseBody处理数据

作用：

该注解用于将Controller的方法返回的对象，通过适当的HttpMessageConverter转换为指定格式后，写入到Response对象的body数据区。

使用时机：

返回的数据不是html标签的页面，而是其他某种格式的数据时（如json、xml等）使用

注：HttpMessageConverter 早期的SpringMVC需要通过配置SpringMVC的xml文件进行相应的映射

本例中的SpringMVC可以自动转JSON格式，无需配置，只需导入SpringMVC支持的json工具包

完成后访问相应路径：



### 三、思考题

1、修改实验7和实验8中的项目使用springMVC框架。