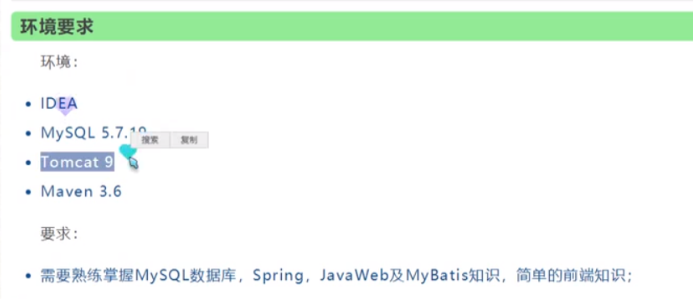
**SpringMVC: 整合SSM**



创建数据库

|  |
| --- |
| CREATE TABLE `books`(  `bookID` INT(10) NOT NULL AUTO\_INCREMENT COMMENT '书籍号码',  `bookName` VARCHAR(100) NOT NULL COMMENT '书名',  `bookCounts` INT(11) NOT NULL COMMENT '书籍数量',  `detail` VARCHAR(200) NOT NULL COMMENT '描述',  PRIMARY KEY `bookID` (`bookID`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8  INSERT INTO `books` (bookID,bookName,bookCounts,detail) VALUES  (1,'Java',1,'o\_o'),  (2,'MySQL',10,'^v^'),  (3,'Linux',6,'0v0'); |

1. 新建一个Maven项目，ssmbuild，添加web的支持
2. 导入相关的pom依赖

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>org.example</groupId>  <artifactId>ssmbuild</artifactId>  <version>1.0-SNAPSHOT</version>   *<!--导入依赖：junit，数据库驱动，数据库连接池，servlet，jsp，mybatis，mytatis-spring，spring-->* <dependencies>  *<!--Junit-->* <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  </dependency>   *<!--数据库驱动-->* <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.47</version>  </dependency>  *<!--数据库连接池-->* <dependency>  <groupId>com.mchange</groupId>  <artifactId>c3p0</artifactId>  <version>0.9.5.5</version>  </dependency>   *<!--Servlet - JSP-->* <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <version>2.5</version>  </dependency>  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>jsp-api</artifactId>  <version>2.1</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>   *<!--Mybatis-->* <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.5.2</version>  </dependency>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>2.0.6</version>  </dependency>   *<!--Spring-->* <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>5.3.10</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>5.3.9</version>  </dependency>   <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>1.18.22</version>  </dependency>  </dependencies>   *<!--静态资源导出问题-->* <build>  <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  <resource>  <directory>src/main/resources</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  </resources>  </build>  </project> |

创建controller、dao、pojo、service包，且在resource包下创建mybatis-config.xml、aplicationContext.xml文件

创建并编写数据库配置文件database.properties

|  |
| --- |
| jdbc.driver=com.mysql.jdbc.Driver *# 如果使用的是MySQL8.0+需要增加一个时区的配&serverTimezone=Asia/Shanghai* jdbc.url=jdbc:mysql://localhost:3306/ssmbuild?useSSL=true&useUnicode=true&characterEncoding=utf8 jdbc.username=root jdbc.password=123456 |

Mybatis.xml:

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8" *?>* <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>* <configuration>    <typeAliases>  <package name="com.xiang.pojo"/>  </typeAliases>   *<!--配置数据源，交给Spring去做了-->* <mappers>  <mapper class="com.xiang.dao.BookMapper"/>  </mappers>  </configuration> |

在pojo包中创建并编写实体类Books.java

|  |
| --- |
| package com.xiang.pojo;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @NoArgsConstructor @AllArgsConstructor public class Books {   private int bookID;  private String bookName;  private int bookCounts;  private String detail;  } |

在dao包下创建并编写接口BookMapper

|  |
| --- |
| package com.xiang.dao;  import com.xiang.pojo.Books; import org.apache.ibatis.annotations.Param;  import java.util.List;  public interface BookMapper {   *// 增加一本书* int addBook(Books books);   *// 删除一本书* int deleteBookbyId(@Param("bookID") int id);   *// 更新一本书* int updateBook(Books books);   *// 查询一本书* Books queryBookById(@Param("bookId") int id);   *// 查询全部的书* List<Books> queryAllBook();  Books queryBookByName(@Param("bookName") String bookName);  } |

在dao包下创建并编写BookMapper.xml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8" *?>* <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>* <mapper namespace="com.xiang.dao.BookMapper">   <insert id="addBook" parameterType="Books">  insert into ssmbuild.books (bookName,bookCounts,detail)  values (#{bookName},#{bookCounts},#{detail});  </insert>   <delete id="deleteBookById" parameterType="int">  delete from ssmbuild.books where bookID = #{bookId}  </delete>   <update id="updateBook" parameterType="Books">  update ssmbuild.books  set bookName=#{bookName},bookCounts=#{bookCounts},detail=#{detail}  where bookID=#{bookID}  </update>   <select id="queryBookById" resultType="Books">  select \* from ssmbuild.books  where bookID=#{bookId}  </select>   <select id="queryAllBook" resultType="Books">  select \* from ssmbuild.books  </select>  <select id="queryBookByName" resultType="Books">  select \*  from ssmbuild.books  where bookName = #{bookName};  </select>  </mapper> |

在service包下创建并编写接口BookService

|  |
| --- |
| package com.xiang.service;  import com.xiang.pojo.Books; import org.apache.ibatis.annotations.Param;  import java.util.List;  public interface BookService {   *// 增加一本书* int addBook(Books books);   *// 删除一本书* int deleteBookbyId(int id);   *// 更新一本书* int updateBook(Books books);   *// 查询一本书* Books queryBookById(int id);   *// 查询全部的书* List<Books> queryAllBook();  Books queryBookByName(String bookName); } |

在service包下创建并编写实现类BookServiceImpl

|  |
| --- |
| package com.xiang.service;  import com.xiang.dao.BookMapper; import com.xiang.pojo.Books;  import java.util.List;  public class BookServiceImpl implements BookService{   *// service调dao层，组合Dao* private BookMapper bookMapper;  public void setBookMapper(BookMapper bookMapper){  this.bookMapper = bookMapper;  }   public int addBook(Books books) {  return bookMapper.addBook(books);  }   public int deleteBookbyId(int id) {  return bookMapper.deleteBookbyId(id);  }   public int updateBook(Books books) {  return bookMapper.updateBook(books);  }   public Books queryBookById(int id) {  return bookMapper.queryBookById(id);  }   public List<Books> queryAllBook() {  return bookMapper.queryAllBook();  }  public Books queryBookByName(String bookName){  return bookMapper.queryBookByName(bookName);  } } |

在resource包下创建并编写spring-dao.xml，spring整合dao层

|  |
| --- |
| *<?*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:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">   *<!--1.spring关联数据库配置文件-->* <context:property-placeholder location="classpath:database.properties"/>   *<!--2.数据库连接池  dbcp: 班自动化操作，需要手动连接  c3p0: 自动化操作（自动化加载配置文件，并且可以自动设置到对象中）,这里用c3p0  druid:  hikari: spring 2.0 默认的连接池  -->* <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">  <property name="driverClass" value="${jdbc.driver"/>  <property name="jdbcUrl" value="${jdbc.url}"/>  <property name="user" value="${jdbc.username}"/>  <property name="password" value="${jdbc.password"/>  *<!--c3p0连接池的私有属性-->* <property name="maxPoolSize" value="30"/>  <property name="minPoolSize" value="10"/>  *<!--关闭连接后不自动commit-->* <property name="autoCommitOnClose" value="false"/>  *<!--获取连接超时时间-->* <property name="checkoutTimeout" value="10000"/>  *<!--当前连接获取失败重试次数-->* <property name="acquireRetryAttempts" value="2"/>  </bean>   *<!--3.sqlSessionFactory-->* <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">  <property name="dataSource" ref="dataSource"/>  *<!--绑定Mybatis的配置文件-->* <property name="configLocation" value="classpath:mybatis-config.xml"/>  </bean>   *<!--配置dao接口扫描包，实现了dao接口动态注入到spring容器中-->* <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  *<!--注入sqlSessionFactory-->* <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  *<!--要扫描的包-->* <property name="basePackage" value="com.xiang.dao"/>  </bean>  </beans> |

在resource包下创建并编写spring-service.xml

|  |
| --- |
| *<?*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:context="http://www.springframework.org/schema/context"  xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx.xsd">   *<!--1.扫描service下的包-->* <context:component-scan base-package="com.xiang.service"/>   *<!--2.将我们的所有业务类，注入到spring中，可以通过配置，或者注解实现-->* <bean id="bookServiceImpl" class="com.xiang.service.BookServiceImpl">  <property name="bookMapper" ref="bookMapper"/>  </bean>   *<!--3.声明式事务配置-->* <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  *<!--注入数据源-->* <property name="dataSource" ref="dataSource"/>  </bean>   *<!--4.aop事务支持-->  <!--结合aop实现事务的织入-->  <!--配置事务通知-->* <tx:advice id="txAdvice" transaction-manager="transactionManager">  *<!--给哪些方法配置事务-->  <!--配置事务的传播特性-->* <tx:attributes>  <tx:method name="\*" propagation="REQUIRED"/>  </tx:attributes>  </tx:advice>   *<!--配置事务的切入点-->* <aop:config>  <aop:pointcut id="txPointCut" expression="execution(\* com.xiang.dao.\*.\*(..))"/>  <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointCut"/>  </aop:config>  </beans> |

添加web框架支持

编写web.xml文件

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"  version="4.0">   *<!--DispatchServlet-->* <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mvc.xml</param-value>  </init-param>  *<!--随服务器一起启动-->* <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern> *<!--"/"不包括.jsp文件；"/\*"包括.jsp文件-->* </servlet-mapping>   *<!--乱码过滤-->* <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>   *<!--Session-->* <session-config>  <session-timeout>15</session-timeout>  </session-config>  </web-app> |

在resource包下创建并编写spring-mvc.xml文件

|  |
| --- |
| *<?*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:mvc="http://www.springframework.org/schema/mvc"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  https://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd">   *<!--1.注解驱动-->* <mvc:annotation-driven/>   *<!--2.静态资源过滤-->  <!--在springMVC-servlet.xml中配置<mvc:default-servlet-handler />后，  会在Spring MVC上下文中定义一个org.springframework.web.servlet.resource.DefaultServletHttpRequestHandler，  它会像一个检查员，对进入DispatcherServlet的URL进行筛查，如果发现是静态资源的请求，就将该请求转由Web应用服务器默认的Servlet处理，  如果不是静态资源的请求，才由DispatcherServlet继续处理。一般Web应用服务器默认的Servlet名称是"default"，  因此DefaultServletHttpRequestHandler可以找到它。如果你所有的Web应用服务器的默认Servlet名称不是"default"，则需要通过default-servlet-name属性显示指定：  <mvc:default-servlet-handler default-servlet-name="所使用的Web服务器默认使用的Servlet名称" />-->* <mvc:default-servlet-handler/>   *<!--3.扫描包：controller-->* <context:component-scan base-package="com.xiang.controller"/>   *<!--4.视图解析器-->* <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix" value="/WEB-INF/jsp/"/>  <property name="suffix" value=".jsp"/>  </bean> </beans> |

[**【Spring框架】<mvc:default-servlet-handler/>的作用**](https://www.cnblogs.com/dflmg/p/6393416.html)

优雅REST风格的资源URL不希望带 .html 或 .do 等后缀.由于早期的Spring MVC不能很好地处理静态资源，所以在web.xml中配置DispatcherServlet的请求映射，往往使用 \*.do 、 \*.xhtml等方式。这就决定了请求URL必须是一个带后缀的URL，而无法采用真正的REST风格的URL。

如果将DispatcherServlet请求映射配置为"/"，则Spring MVC将捕获Web容器所有的请求，包括静态资源的请求，Spring MVC会将它们当成一个普通请求处理，因此找不到对应处理器将导致错误。

如何让Spring框架能够捕获所有URL的请求，同时又将静态资源的请求转由Web容器处理，是可将DispatcherServlet的请求映射配置为"/"的前提。由于REST是Spring3.0最重要的功能之一，所以Spring团队很看重静态资源处理这项任务，给出了堪称经典的两种解决方案。

先调整web.xml中的DispatcherServlet的配置，使其可以捕获所有的请求：

[复制代码](javascript:void(0);)

<servlet>

<servlet-name>springMVC</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springMVC</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

[复制代码](javascript:void(0);)

通过上面url-pattern的配置，所有URL请求都将被Spring MVC的DispatcherServlet截获。

**方法1.采用<mvc:default-servlet-handler />**

<mvc:default-servlet-handler />

在springMVC-servlet.xml中配置<mvc:default-servlet-handler />后，会在Spring MVC上下文中定义一个org.springframework.web.servlet.resource.DefaultServletHttpRequestHandler，它会像一个检查员，对进入DispatcherServlet的URL进行筛查，如果发现是静态资源的请求，就将该请求转由Web应用服务器默认的Servlet处理，如果不是静态资源的请求，才由DispatcherServlet继续处理。

一般Web应用服务器默认的Servlet名称是"default"，因此DefaultServletHttpRequestHandler可以找到它。如果你所有的Web应用服务器的默认Servlet名称不是"default"，则需要通过default-servlet-name属性显示指定：

<mvc:default-servlet-handler default-servlet-name="所使用的Web服务器默认使用的Servlet名称" />

**方法2.采用<mvc:resources />**

<mvc:default-servlet-handler />将静态资源的处理经由Spring MVC框架交回Web应用服务器处理。而<mvc:resources />更进一步，由Spring MVC框架自己处理静态资源，并添加一些有用的附加值功能。

首先，<mvc:resources />允许静态资源放在任何地方，如WEB-INF目录下、类路径下等，你甚至可以将JavaScript等静态文件打到JAR包中。通过location属性指定静态资源的位置，由于location属性是Resources类型，因此可以使用诸如"classpath:"等的资源前缀指定资源位置。传统Web容器的静态资源只能放在Web容器的根路径下，<mvc:resources />完全打破了这个限制。

其次，<mvc:resources />依据当前著名的Page Speed、YSlow等浏览器优化原则对静态资源提供优化。你可以通过cacheSeconds属性指定静态资源在浏览器端的缓存时间，一般可将该时间设置为一年，以充分利用浏览器端的缓存。在输出静态资源时，会根据配置设置好响应报文头的Expires 和 Cache-Control值。

在接收到静态资源的获取请求时，会检查请求头的Last-Modified值，如果静态资源没有发生变化，则直接返回303相应状态码，提示客户端使用浏览器缓存的数据，而非将静态资源的内容输出到客户端，以充分节省带宽，提高程序性能。

在springMVC-servlet中添加如下配置：

<mvc:resources location="/,classpath:/META-INF/publicResources/" mapping="/resources/\*\*"/>

以上配置将Web根路径"/"及类路径下 /META-INF/publicResources/ 的目录映射为/resources路径。假设Web根路径下拥有images、js这两个资源目录,在images下面有bg.gif图片，在js下面有test.js文件，则可以通过 /resources/images/bg.gif 和 /resources/js/test.js 访问这二个静态资源。

假设WebRoot还拥有images/bg1.gif 及 js/test1.js，则也可以在网页中通过 /resources/images/bg1.gif 及 /resources/js/test1.js 进行引用。

编写applicationContext.xml文件

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8" *?>* <beans xmlns="http://www.springframework.org/schema/beans"  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.xsd">   <import resource="classpath:spring-dao.xml"/>  <import resource="classpath:spring-service.xml"/>  <import resource="classpath:spring-mvc.xml"/>  </beans> |

在controller包下创建并编写BookController

|  |
| --- |
| package com.xiang.controller;  import com.sun.org.apache.xpath.internal.operations.Mod; import com.xiang.pojo.Books; import com.xiang.service.BookService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.beans.factory.annotation.Qualifier; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestMapping;  import java.awt.print.Book; import java.util.ArrayList; import java.util.List;  @Controller @RequestMapping("/book") public class BookController {  *// controller层调service层* @Autowired  @Qualifier("bookServiceImpl")  private BookService bookService;   *// 查询全部的书籍，并且返回到一个书籍展示页面* @RequestMapping("/allBook")  public String list(Model model){  List<Books> list = bookService.queryAllBook();  model.addAttribute("list", list);  System.*out*.println("查询全部书籍");  return "allBook";  }   *// 跳转到增加书籍页面* @RequestMapping("/toAddBook")  public String toAddPage(){  return "addBook";  }   *// 添加书籍的请求* @RequestMapping("/addBook")  public String addBook(Books books){  System.*out*.println("addBook==>" + books);  bookService.addBook(books);  return "redirect:/book/allBook";  }   *// 跳转到修改书籍页面* @RequestMapping("/toUpdate")  public String toUpdatePage(int id, Model model){  Books books = bookService.queryBookById(id);  model.addAttribute("QBook", books);  return "updateBook";  }   *// 修改书籍* @RequestMapping("/updateBook")  public String updateBook(Books books){  System.*out*.println("updateBook==>" + books);  bookService.updateBook(books);  return "redirect:/book/allBook";  }   *// 删除书籍* @RequestMapping("/deleteBook/{bookId}")  public String deleteBook(@PathVariable("bookId") int id){  bookService.deleteBookById(id);  return "redirect:/book/allBook";  }   *// 查询书籍* @RequestMapping("/queryBook")  public String queryBookByName(String bookName, Model model){  Books books = bookService.queryBookByName(bookName);  System.*err*.println("queryBook==>" + books);  List<Books> list = new ArrayList<Books>();  list.add(books);  if (books == null){  list = bookService.queryAllBook();  model.addAttribute("error","所查书籍不存在");  }  model.addAttribute("list", list);  return "allBook";  }  } |

在WEB-INF包下创建jsp文件夹，并在jsp文件夹下创建并编写allBook.jsp、updateBook.jsp、addBook.jsp前端文件

首先编写index.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html>  <head>  <title>首页</title>   <style>  a{  text-decoration: none;  color: dodgerblue;  font-size: 16px;  }  h3{  width: 170px;  height: 38px;  margin: 100px auto;  text-align: center;  line-height: 38px;  background: blanchedalmond;  border-radius: 5px;  }  </style>  </head>  <body>   <h3>  *<%--href: 取绝对地址--%>* <a href="${pageContext.request.contextPath}/book/allBook">进入书籍展示页面</a>  </h3>   </body> </html> |

编写allBook.jsp：

|  |
| --- |
| <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %> <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>书籍展示</title>  *<%--引入BootStrap美化界面--%>* <link href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet"> </head> <body>  <div class="container">   <div class="row clearfix">  <div class="col-md-12 column">  <div class="page-header">  <h1>  <small>书籍列表----展示所有书籍</small>  </h1>  </div>  </div>  <div class="row">  <div class="col-md-4 column">  *<%--toAddBook--%>* <a class="btn btn-primary" href="${pageContext.request.contextPath}/book/toAddBook">添加书籍</a>  <a class="btn btn-primary" href="${pageContext.request.contextPath}/book/allBook">显示全部书籍</a>  </div>  <div class="col-md-4 column">  *<%--查询书籍--%>* <form action="${pageContext.request.contextPath}/book/queryBook" method="post" style="float: right" class="form-inline">  <input type="text" name="bookName" class="form-control" placeholder="请输入要查询的书籍名称">  <input type="submit" value="查询" class="btn btn-primary">  <span style="color:red;font-weight: bold">${error}</span>  </form>  </div>  </div>  </div>  <div class="row clearfix">  <div class="col-md-12 column">  <table class="table table-hover table-striped">  <thead>  <tr>  <th>书籍编号</th>  <th>书籍名称</th>  <th>书籍数量</th>  <th>书籍详情</th>  <th>相关操作</th>  </tr>  </thead>  *<%--书籍从数据库中查询出来，我们要从list中遍历出书籍：foreach--%>* <tbody>  <c:forEach var="book" items="${list}">  <tr>  <td>${book.bookID}</td>  <td>${book.bookName}</td>  <td>${book.bookCounts}</td>  <td>${book.detail}</td>  <td>  <a href="${pageContext.request.contextPath}/book/toUpdate?id=${book.bookID}">修改</a>  &nbsp; | &nbsp;  <a href="${pageContext.request.contextPath}/book/deleteBook/${book.bookID}">删除</a>  </td>  </tr>  </c:forEach>  </tbody>  </table>  </div>  </div>  </div>  </body> </html> |

编写updateBook.jsp：

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>修改书籍</title>  *<%--引入BootStrap美化界面--%>* <link href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">  </head> <body>  <div class="container">   <div class="row clearfix">  <div class="col-md-12 column">  <div class="page-header">  <h1>  <small>修改书籍</small>  </h1>  </div>  </div>  </div>   <form action="${pageContext.request.contextPath}/book/updateBook" method="post">  *<%--bookID不需要修改，bookID属性使用前端传递隐藏域--%>* <input type="hidden" name="bookID" value="${QBook.bookID}">  <div class="form-group">  <label >书籍名称</label>  <input type="text" name="bookName" value="${QBook.bookName}" class="form-control" required>  </div>  <div class="form-group">  <label>书籍数量</label>  <input type="text" name="bookCounts" value="${QBook.bookCounts}" class="form-control" required>  </div>  <div class="form-group">  <label>书籍描述</label>  <input type="text" name="detail" value="${QBook.detail}" class="form-control" required>  </div>  <div class="form-group">  <input type="submit" class="form-control" value="确认修改">  </div>   </form>  </div> </body> </html> |

编写addBook.jsp：

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>添加书籍</title>   <title>添加书籍</title>  *<%--引入BootStrap美化界面--%>* <link href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">  </head> <body>  <div class="container">   <div class="row clearfix">  <div class="col-md-12 column">  <div class="page-header">  <h1>  <small>添加书籍</small>  </h1>  </div>  </div>  </div>   <form action="${pageContext.request.contextPath}/book/addBook" method="post">  <div class="form-group">  <label >书籍名称</label>  <input type="text" name="bookName" class="form-control" required>  </div>  <div class="form-group">  <label>书籍数量</label>  <input type="text" name="bookCounts" class="form-control" required>  </div>  <div class="form-group">  <label>书籍描述</label>  <input type="text" name="detail" class="form-control" required>  </div>  <div class="form-group">  <input type="submit" class="form-control" value="确认添加">  </div>   </form>  </div> </body> </html> |

前端组件或框架可以去Bootstrap网站获得：https://www.bootcss.com/

Bootstrap可视化布局系统，可编辑前端页面：https://www.bootcss.com/p/layoutit/