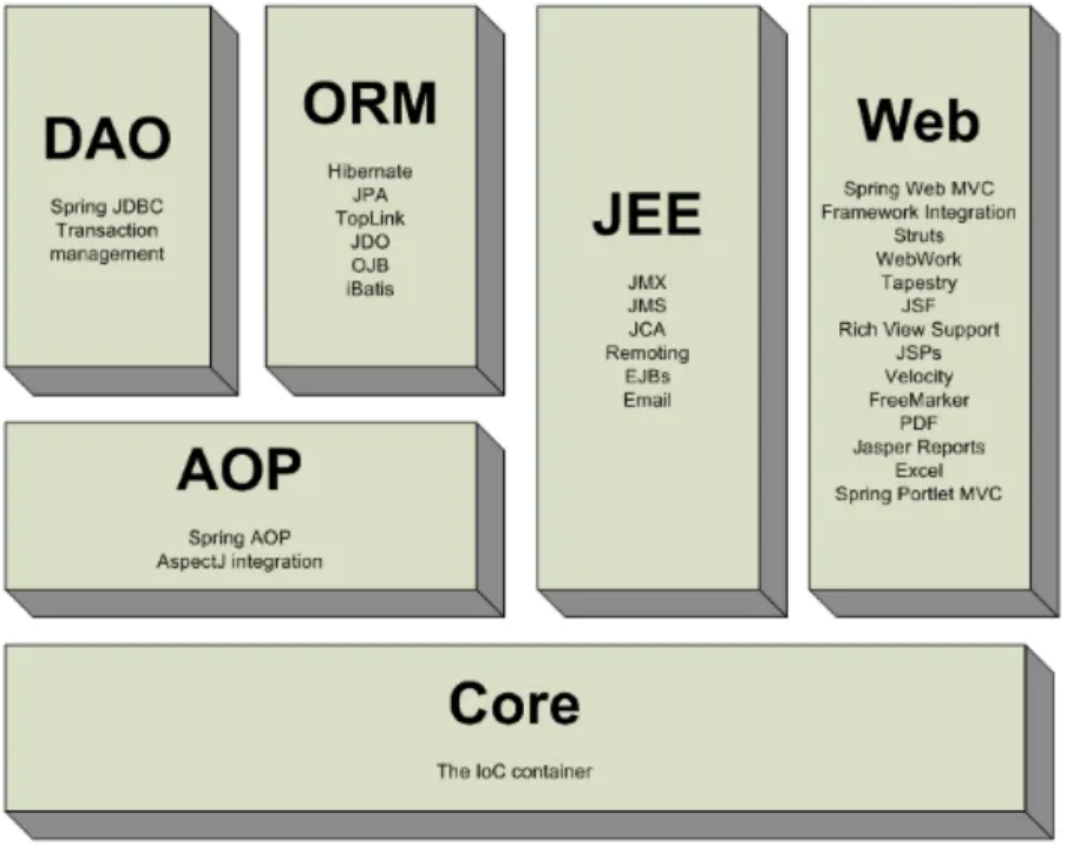
SpringMvc

1. SpringMVC的简介

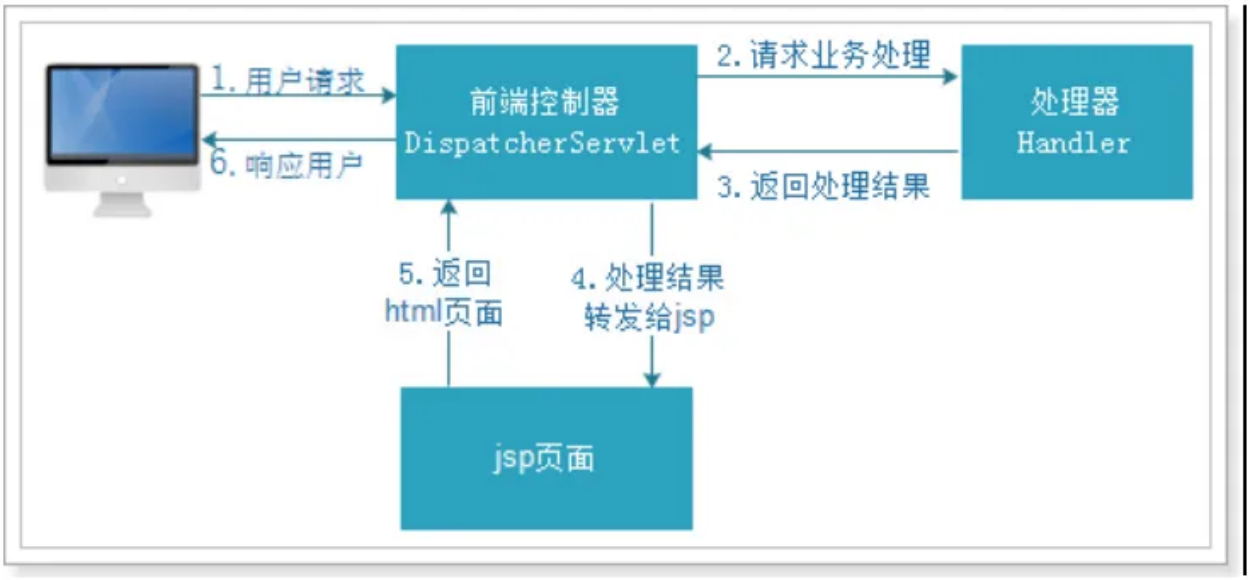
1.1 SpringMVC概述

SpringMVC 是一种基于 Java 的实现 MVC 设计模型的请求驱动类型的轻量级 Web 框架，属于 SpringFrameWork 的后续产品，已经融合在 Spring Web Flow 中。

SpringMVC 已经成为目前最主流的MVC框架之一，并且随着Spring3.0 的发布，全面超越 Struts2，成为最优秀的 MVC 框架。它通过一套注解，让一个简单的 Java 类成为处理请求的控制器，而无须实现任何接口。同时它还支持 RESTful 编程风格的请求。



1.2 SpringMvc处理流程



1.3 SpringMVC入门程序

需求：客户端发起请求，服务器端接收请求，执行逻辑并进行视图跳转。

开发步骤

①导入SpringMVC相关坐标

②配置SpringMVC核心控制器DispathcerServlet

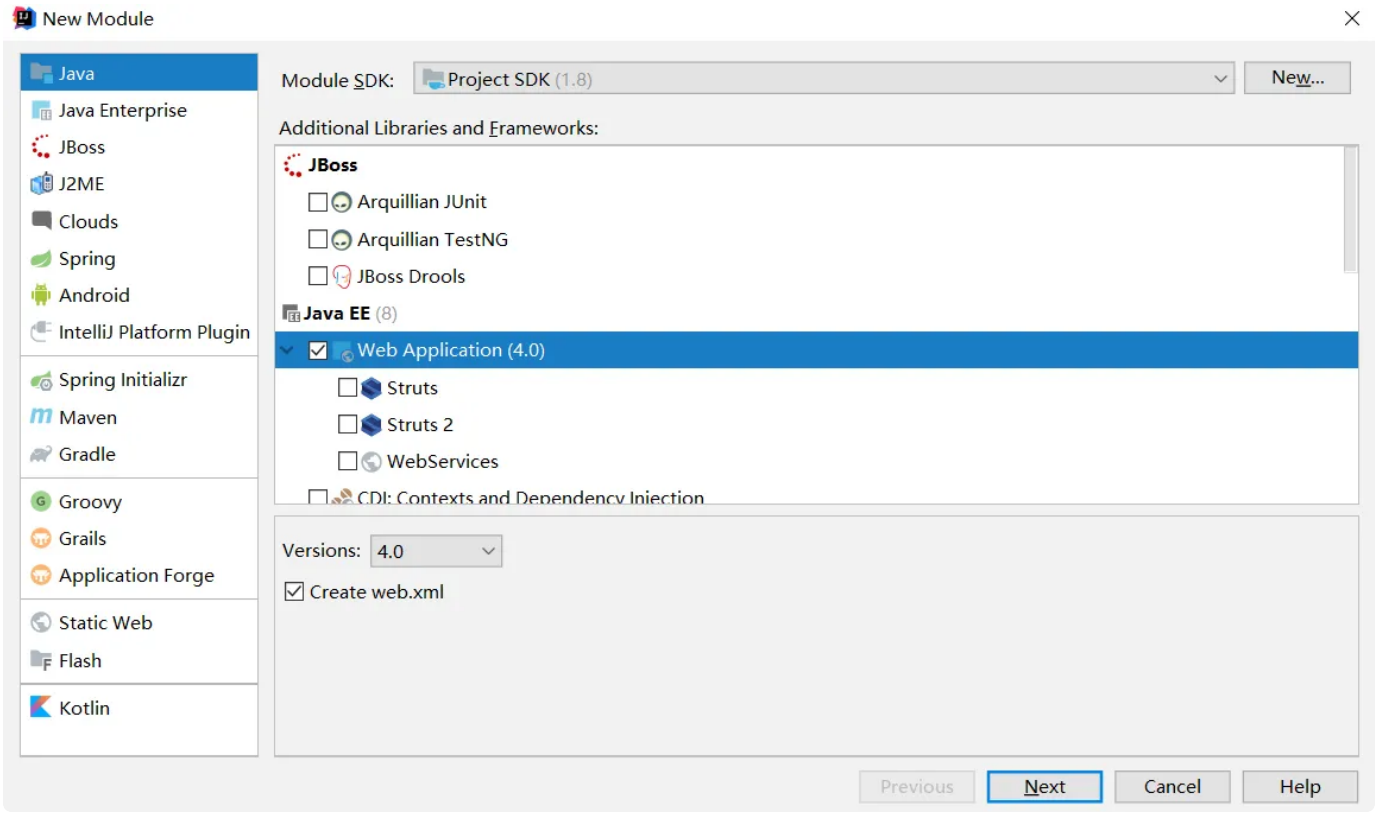
③创建Controller类和视图页面

④使用注解配置Controller类中业务方法的映射地址

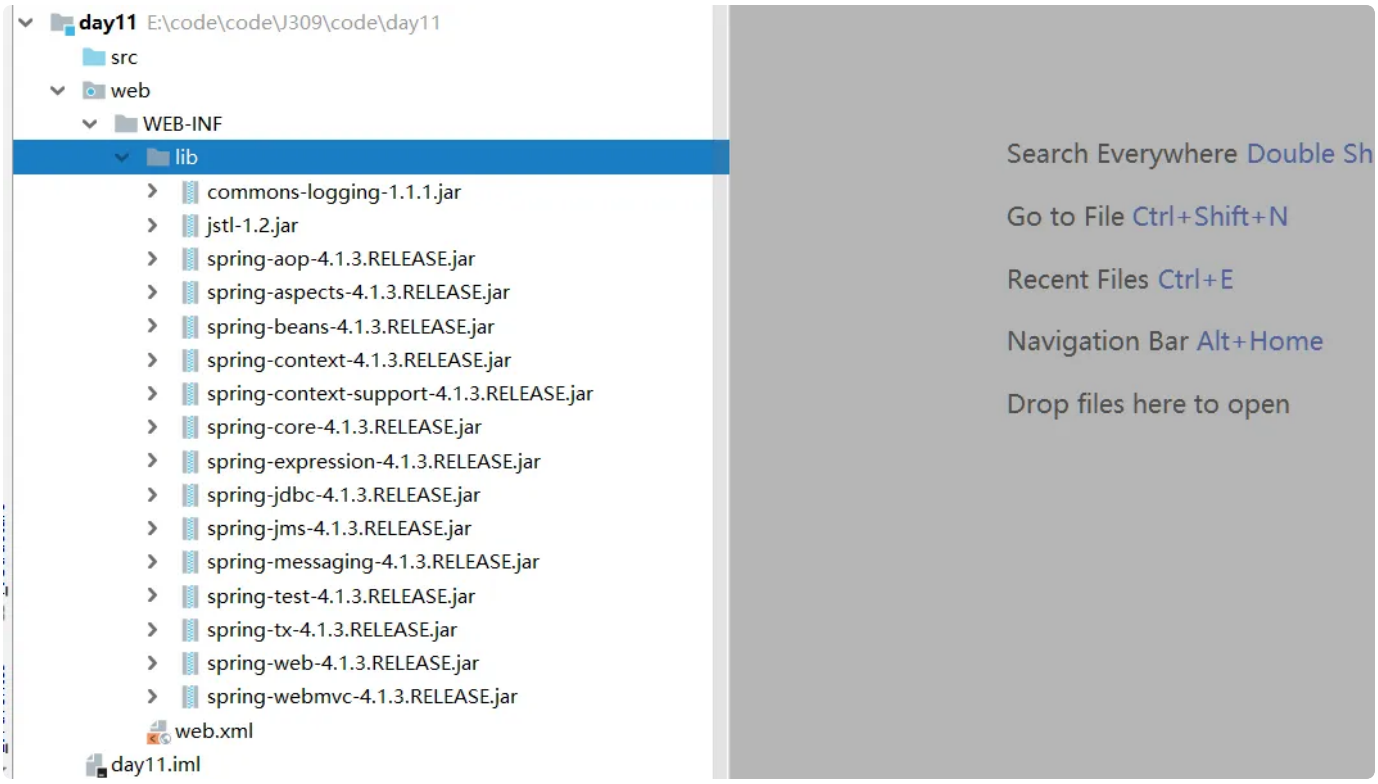
⑤配置SpringMVC核心文件 spring-mvc.xml

⑥客户端发起请求测试

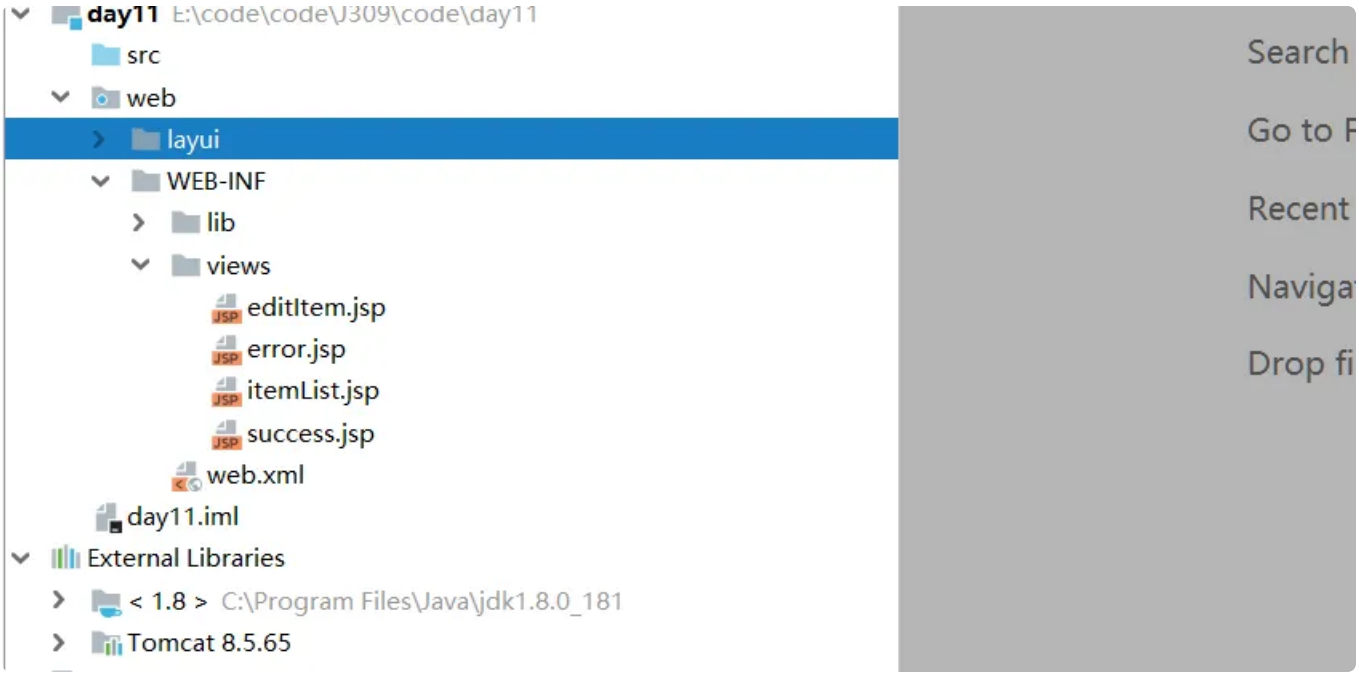
1.3.1 新建工程


New Module
Java
Java Enterprise
JBoss
J2ME
Clouds
Spring
Android
IntelliJ Platform Plugin
Spring Initializr
Maven
Gradle
Groovy
Grails
Application Forge
Static Web
Flash
Kotlin
Module SDK:
Project SDK (1.8)
New...
Additional Libraries and Frameworks:
JBoss
Arquillian JUnit
Arquillian TestNG
JBoss Drools
Java EE (8)
Web Application (4.0)
Struts
Struts 2
WebServices
CDI: Contexts and Dependency Injection
Versions:
4.0
Create web.xml
Previous
Next
Cancel
Help

1.3.2 导包


day11 E:\code\code\J309\code\day11
src
web
WEB-INF
lib
commons-logging-1.1.1.jar
jstl-1.2.jar
spring-aop-4.1.3.RELEASE.jar
spring-aspects-4.1.3.RELEASE.jar
spring-beans-4.1.3.RELEASE.jar
spring-context-4.1.3.RELEASE.jar
spring-context-support-4.1.3.RELEASE.jar
spring-core-4.1.3.RELEASE.jar
spring-expression-4.1.3.RELEASE.jar
spring-jdbc-4.1.3.RELEASE.jar
spring-jms-4.1.3.RELEASE.jar
spring-messaging-4.1.3.RELEASE.jar
spring-test-4.1.3.RELEASE.jar
spring-tx-4.1.3.RELEASE.jar
spring-web-4.1.3.RELEASE.jar
spring-webmvc-4.1.3.RELEASE.jar
web.xml
day11.iml
Search Everywhere Double Sh
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open

1.3.3 拷贝jsp和静态资源



1.3.4 拷贝pojo和导入sql

Java

```java
public class Item implements Serializable {

    private Integer id;

    private String name;

    private Float price;

    private String pic;

    private Date createtime;

    private String detail;

    public Item(){

    }

    public Item(Integer id, String name, Float price, String pic, Date createtime, String detail) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.pic = pic;
        this.createtime = createtime;
        this.detail = detail;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name == null ? null : name.trim();
    }

    public Float getPrice() {
```

```
        return price;
    }

    public void setPrice(Float price) {
        this.price = price;
    }

    public String getPic() {
        return pic;
    }

    public void setPic(String pic) {
        this.pic = pic == null ? null : pic.trim();
    }

    public Date getCreatetime() {
        return createtime;
    }

    public void setCreatetime(Date createtime) {
        this.createtime = createtime;
    }

    public String getDetail() {
        return detail;
    }

    public void setDetail(String detail) {
        this.detail = detail == null ? null : detail.trim();
    }
}
```

1.3.5 编写controller

```java
/**
 * Created by zxd on 2022/10/27 9:47
 */
@Controller
public class ItemController {



    //配置方法的映射路径  访问路径
    @RequestMapping("/queryItemList")
    public ModelAndView queryItemList(){

        //1.查询商品列表
        List list=new ArrayList();
        list.add(new Item(1,"华为mate50",8000F,"xxx",new Date(),"xxx"));
        list.add(new Item(2,"华为mate60",8000F,"xxx",new Date(),"xxx"));
        list.add(new Item(3,"华为mate80",8000F,"xxx",new Date(),"xxx"));


        //2.放入数据到request域
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.addObject("list",list);

        //3.转发到jsp
        modelAndView.setViewName("/WEB-INF/views/itemList.jsp");

        return modelAndView;
    }


}
```

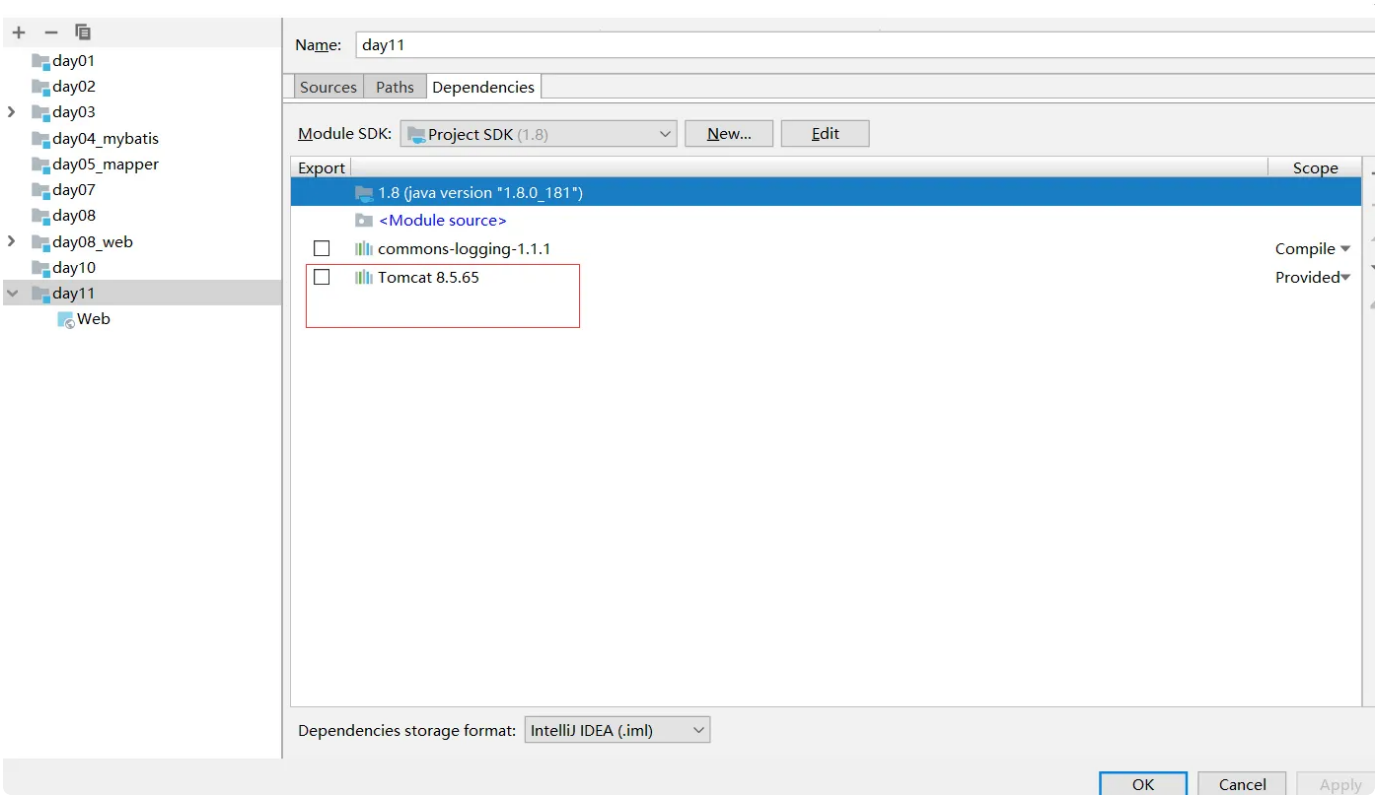
1.3.6 springmvc.xml

```java
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">


    <!--配置扫描controller包-->
    <context:component-scan base-package="com.gxa.controller"/>


</beans>
```

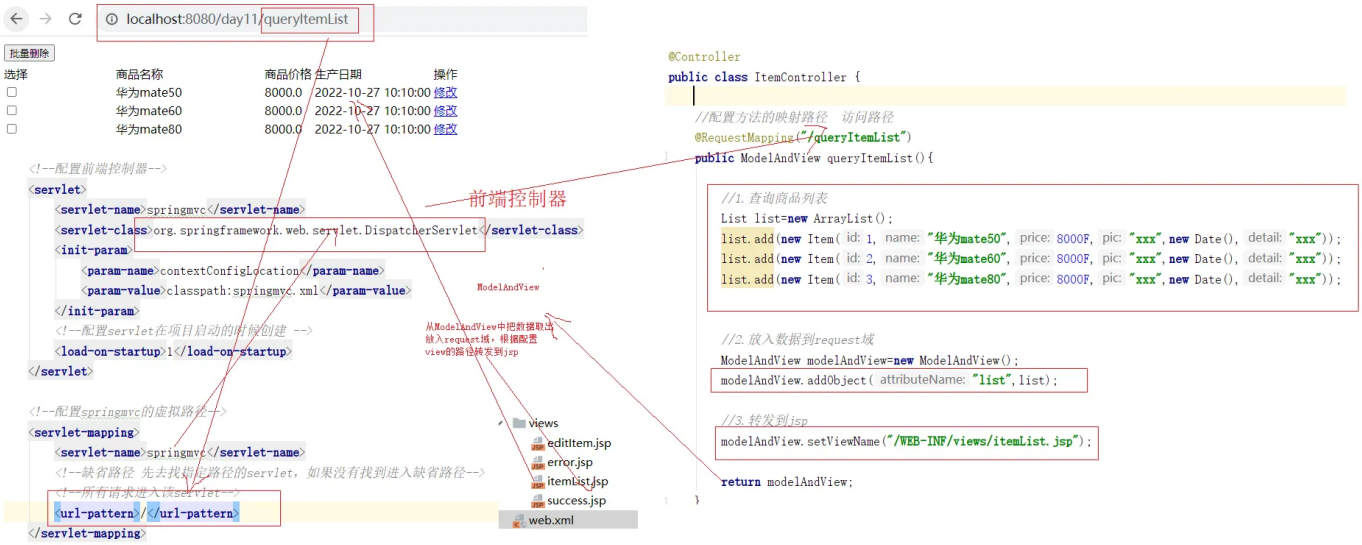
1.3.7 配置web.xml

Java

```xml
<?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">


    <!--配置前端控制器-->
    <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:springmvc.xml</param-value>
        </init-param>
        <!--配置servlet在项目启动的时候创建 -->
        <load-on-startup>1</load-on-startup>
    </servlet>

    <!--配置springmvc的虚拟路径-->
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <!--缺省路径 先去找指定路径的servlet，如果没有找到进入缺省路径-->
        <!--所有请求进入该servlet-->
        <url-pattern>/</url-pattern>
    </servlet-mapping>



</web-app>
```

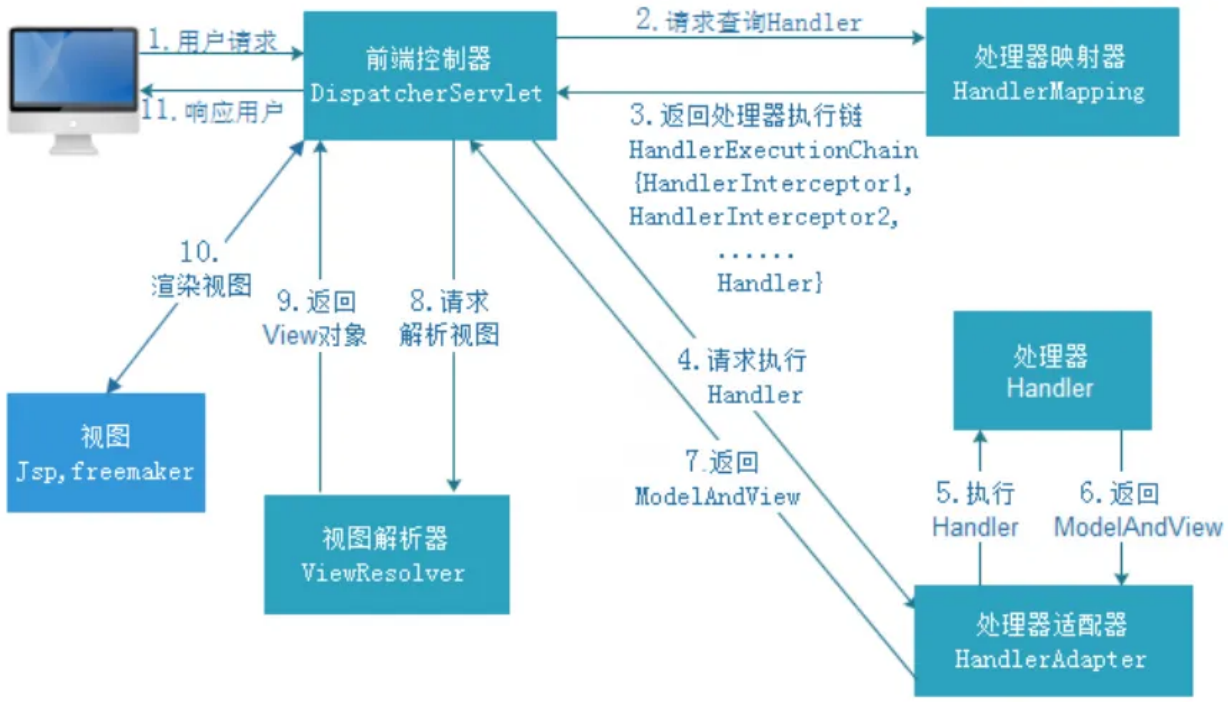


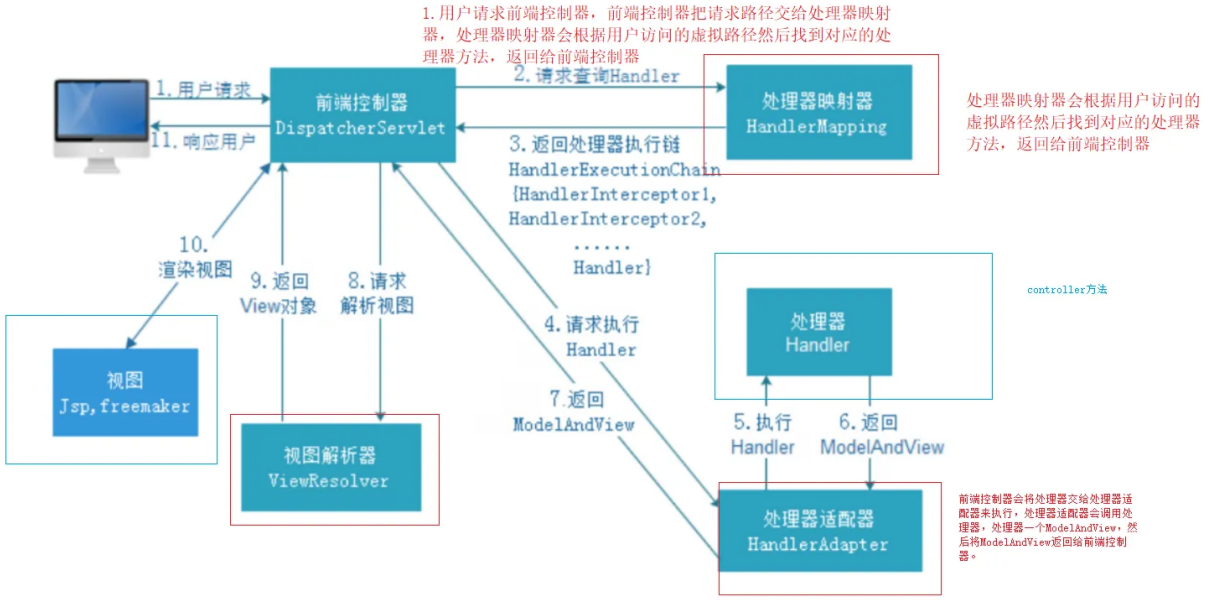
1.3.8 springmvc基本流程



2.SpringMvc架构

2.1 SpringMVC访问流程





①用户发送请求至前端控制器DispatcherServlet。

②DispatcherServlet收到请求调用HandlerMapping处理器映射器。

③处理器映射器找到具体的处理器(可以根据xml配置、注解进行查找)，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

④DispatcherServlet调用HandlerAdapter处理器适配器。

⑤HandlerAdapter经过适配调用具体的处理器(Controller，也叫后端控制器)。

⑥Controller执行完成返回ModelAndView。

⑦HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet。

⑧DispatcherServlet将ModelAndView传给ViewReslover视图解析器。

⑨ViewReslover解析后返回具体View。

⑩DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）。DispatcherServlet响应用户。

2.2 SpringMVC组件解析

1. 前端控制器：DispatcherServlet

用户请求到达前端控制器，它就相当于 MVC 模式中的 C，DispatcherServlet 是整个流程控制的中心，由

它调用其它组件处理用户的请求，DispatcherServlet 的存在降低了组件之间的耦合性。

2. 处理器映射器：HandlerMapping

HandlerMapping 负责根据用户请求找到 Handler 即处理器，SpringMVC 提供了不同的映射器实现不同的

映射方式，例如：配置文件方式，实现接口方式，注解方式等。

3. 处理器适配器：HandlerAdapter

通过 HandlerAdapter 对处理器进行执行，这是适配器模式的应用，通过扩展适配器可以对更多类型的处理

器进行执行。

4. 处理器：Handler

它就是我们开发中要编写的具体业务控制器。由 DispatcherServlet 把用户请求转发到 Handler。由 Handler 对具体的用户请求进行处理。

5. 视图解析器：View Resolver

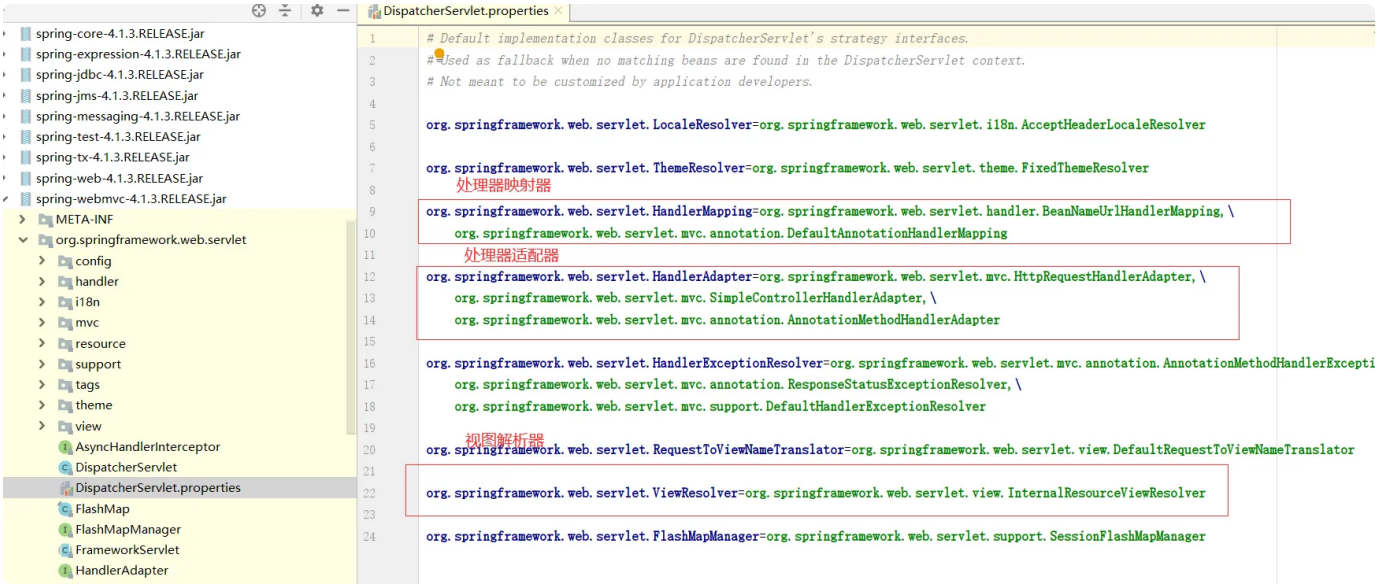
View Resolver 负责将处理结果生成 View 视图，View Resolver 首先根据逻辑视图名解析成物理视图名，即具体的页面地址，再生成 View 视图对象，最后对 View 进行渲染将处理结果通过页面展示给用户。

6. 视图：View

SpringMVC 框架提供了很多的 View 视图类型的支持，包括：jstlView、freemarkerView、pdfView 等。最常用的视图就是 jsp。一般情况下需要通过页面标签或页面模版技术将模型数据通过页面展示给用户，需要由程序员根据业务需求开发具体的页面

2.3 默认加载组件

我们刚才没有做任何的配置，就可以使用三大组件，springmvc做了默认的配置



2.4 配置扫描controller

Java

```
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">


    <!--配置扫描controller包-->
    <context:component-scan base-package="com.gxa.controller"/>


</beans>
```

2.5 配置处理器映射器和处理器适配器

Java

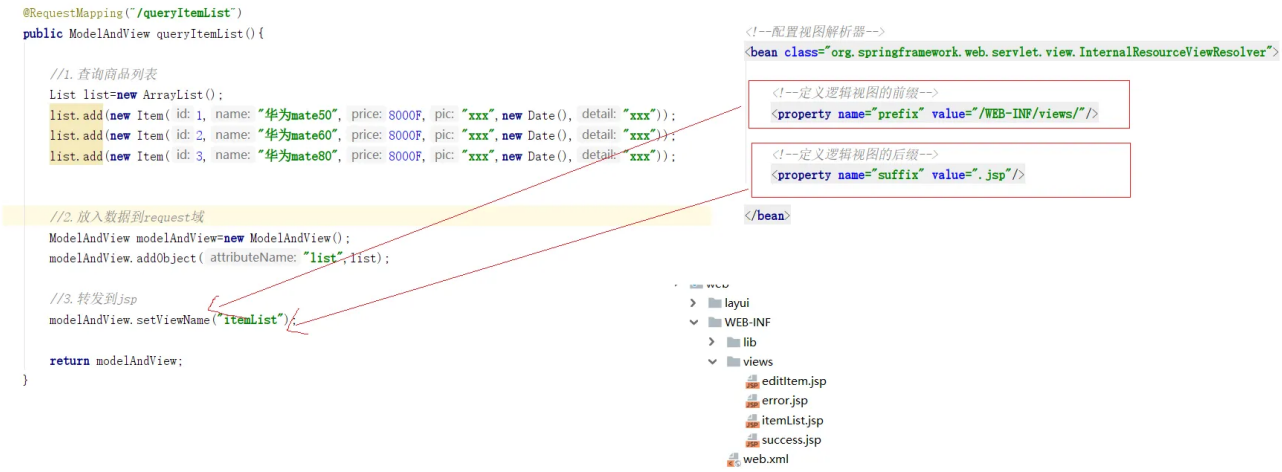
```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">


    <!--配置扫描controller包-->
    <context:component-scan base-package="com.gxa.controller"/>

    <!--配置处理器映射器-->
    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>

    <!--配置处理适配器-->
    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>


</beans>
```

注解驱动

```Java
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">


    <!--配置扫描controller包-->
    <context:component-scan base-package="com.gxa.controller"/>

    <!--注解驱动 自动配置注解版本的处理器映射器+适配器-->
    <mvc:annotation-driven/>

</beans>
```

2.6 配置视图解析器

```Java


    <!--配置视图解析器-->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

        <!--定义逻辑视图的前缀-->
        <property name="prefix" value="/WEB-INF/views/"/>

        <!--定义逻辑视图的后缀-->
        <property name="suffix" value=".jsp"/>

    </bean>

```



2.7 处理不拦截静态资源

思考为什么没有样式?

web.xml Java

```
    <!--配置springmvc的虚拟路径-->
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <!--缺省路径 先去找指定路径的servlet，如果没有找到进入缺省路径-->
        <!--所有请求进入该servlet-->
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

思路:

1.静态资源的请求不进入springmvc

2.进入了springmvc，但是我告诉springmvc他是一个静态资源，springmvc不要处理。

```java
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

    <!--配置不拦截静态资源-->
    <mvc:resources mapping="/layui/**" location="/layui/"/>

    <!--配置扫描controller包-->
    <context:component-scan base-package="com.gxa.controller"/>

    <!--注解驱动 自动配置注解版本的处理器映射器+适配器-->
    <mvc:annotation-driven/>


    <!--配置视图解析器-->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

        <!--定义逻辑视图的前缀-->
        <property name="prefix" value="/WEB-INF/views/"/>

        <!--定义逻辑视图的后缀-->
        <property name="suffix" value=".jsp"/>

    </bean>

</beans>
```

3.SSM整合

spring+springmvc+mybatis

整合包

整合包

spring+mybatis

hibernate jpa

3.1 整合思路

dao层

1.SqlMapConfig.xml mybatis配置文件 废弃

applicationContext-dao.xml

1.配置数据库连接池

2.SqlSessionFactory 交给spring管理 FactoryBean mybatis-spring整合包

3.配置扫描mapper接口

service

applicationContext-service.xml 扫描service包

applicationContext-tx.xml 配置事务管理

applicationContext.xml

controller层

springmvc.xml

1.配置扫描controller包

2.配置注解驱动

3.配置视图解析器

web.xml

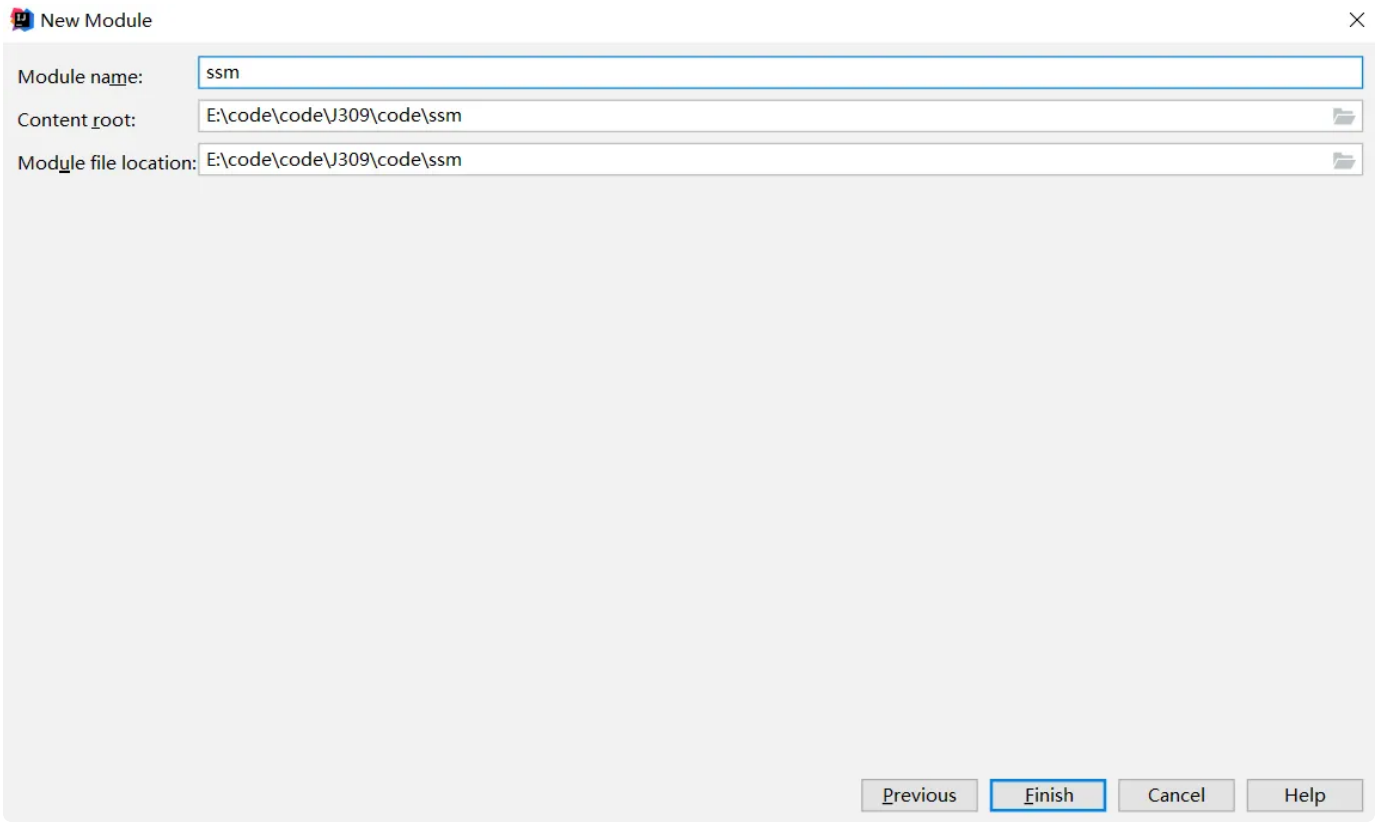
配置前端控制器

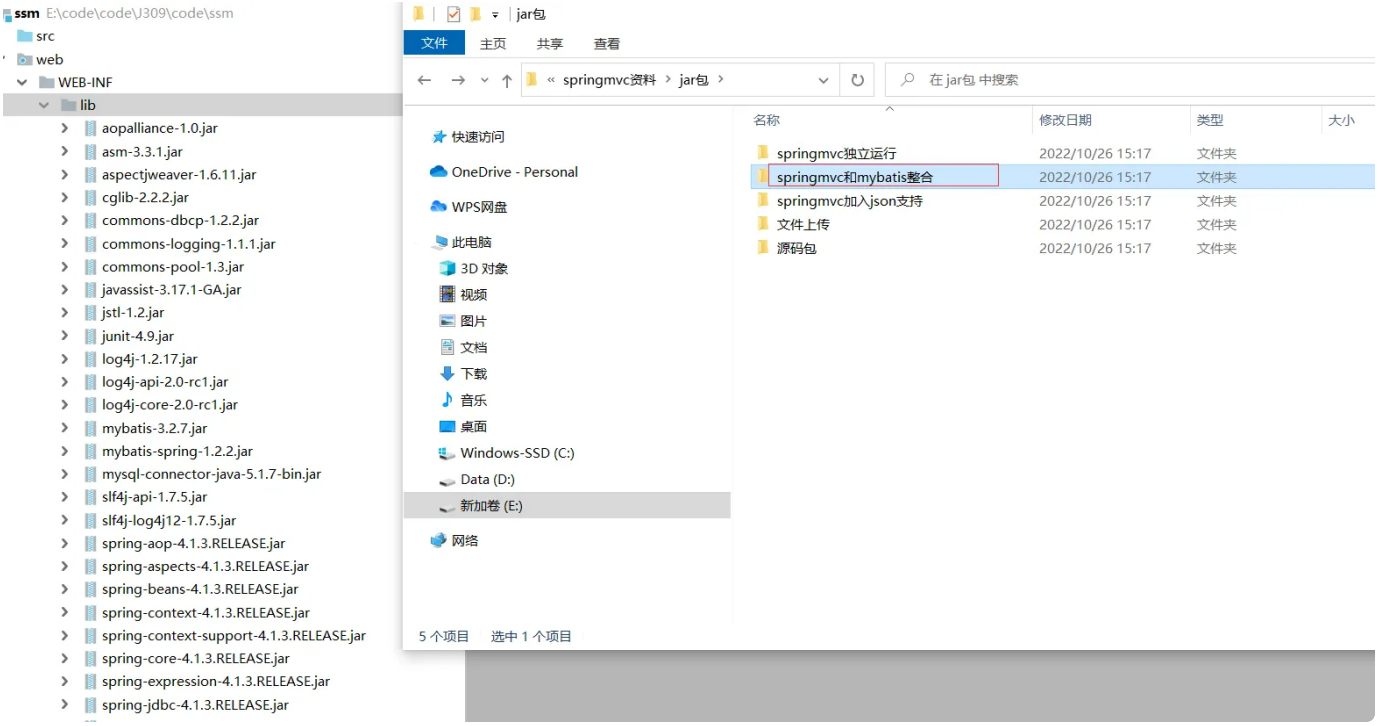
配置spring的监听器

spring父子容器

3.2 配置dao

新建工程导入jar包





applicationContext-dao.xml

1.配置数据库连接池

2.SqlSessionFactory 交给spring管理　　FactoryBean　mybatis-spring整合包

3.配置扫描mapper接口

```Java
<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-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
      ">


    <!--1.加载外部的properties文件-->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!--2.配置数据库连接池-->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName" value="${jdbc.driverClassName}"></property>
        <property name="url" value="${jdbc.url}"></property>
        <property name="username" value="${jdbc.username}"></property>
        <property name="password" value="${jdbc.password}"></property>
        <property name="minIdle" value="${jdbc.minIdle}"></property>
        <property name="maxActive" value="${jdbc.maxActive}"></property>
    </bean>

    <!--3.配置SqlSessionFactoryBean-->
    <bean class="org.mybatis.spring.SqlSessionFactoryBean">
        <!--注入连接池-->
        <property name="dataSource" ref="dataSource"/>
        <!--配置别名-->
        <property name="typeAliasesPackage" value="com.gxa.pojo"/>
    </bean>


    <!--配置扫描mapper接口-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.gxa.mapper"/>
    </bean>


</beans>
```

3.3 配置service

applicationContext-service.xml 扫描service包

applicationContext-service.xml Java

```
<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-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
      ">

    <!--配置扫描service包-->
    <context:component-scan base-package="com.gxa.service"/>

</beans>
```

applicationContext-tx.xml 配置事务管理

Java

```
<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:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
       http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd">


      <!--1.配置事务管理器-->
      <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
          <property name="dataSource" ref="dataSource"/>
      </bean>


     <!--2.配置事务的通知-->
      <tx:advice id="txAdvice" transaction-manager="transactionManager">
          <tx:attributes>
              <tx:method name="insert*" propagation="REQUIRED" isolation="REPEATABLE_READ" read-only="false"/>
              <tx:method name="update*" propagation="REQUIRED" isolation="REPEATABLE_READ" read-only="false"/>
              <tx:method name="delete*" propagation="REQUIRED" isolation="REPEATABLE_READ" read-only="false"/>
              <tx:method name="find*" propagation="REQUIRED" isolation="REPEATABLE_READ" read-only="true"/>
              <tx:method name="query*" propagation="REQUIRED" isolation="REPEATABLE_READ" read-only="true"/>
          </tx:attributes>
      </tx:advice>


     <!--3.配置织入-->
    <aop:config>

        <!--切面-->

```

```
        <aop:advisor advice-ref="txAdvice" pointcut="execution(* com.gxa.service..*ServiceImpl.*(..))"/>

    </aop:config>



</beans>
```

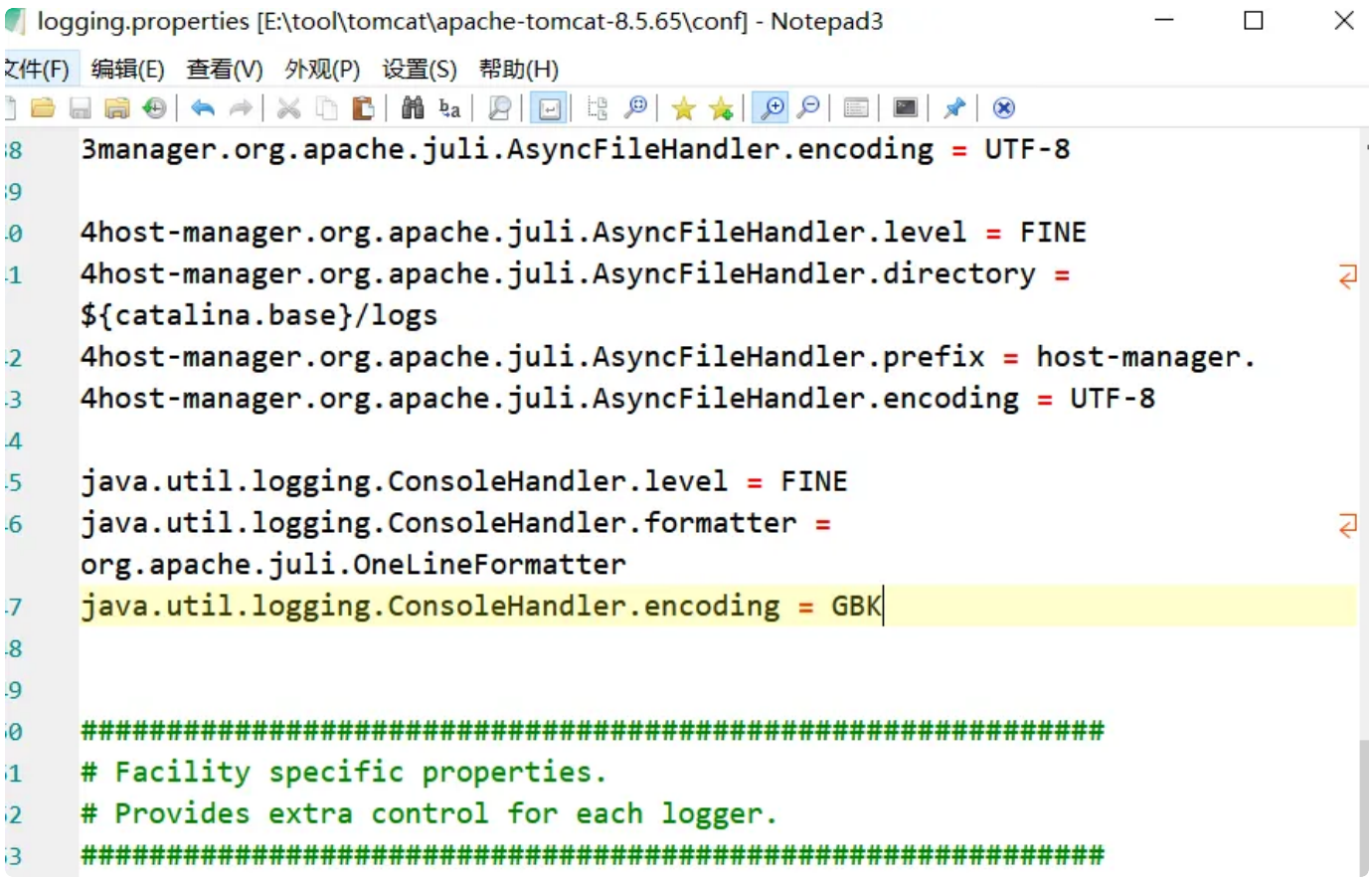
applicationContext.xml Java

```
<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-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
      ">


       <import resource="applicationContext-dao.xml"/>
       <import resource="applicationContext-service.xml"/>
       <import resource="applicationContext-tx.xml"/>


</beans>
```

3.4 配置监听器

Java

```xml
<?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">


    <!--配置监听器-->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!--配置spring初始化文件的位置-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring/applicationContext.xml</param-value>
    </context-param>


</web-app>
```



3.5 配置controller

controller层

springmvc.xml

1.配置扫描controller包

2.配置注解驱动

3.配置视图解析器

springmvc.xml Java

```xml
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

       <!--配置扫描controller包-->
        <context:component-scan base-package="com.gxa.controller"/>

        <!--配置注解驱动-->
         <mvc:annotation-driven/>

        <!--配置视图解析器-->
        <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <property name="prefix" value="/WEB-INF/views/"/>
            <property name="suffix" value=".jsp"/>
        </bean>


</beans>
```

web.xml Java

```xml
<?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">


    <!--配置监听器-->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!--配置spring初始化文件的位置-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring/applicationContext.xml</param-value>
    </context-param>


    <!--配置前端控制器-->
    <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/springmvc.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>




</web-app>
```

3.6 mybatis注解开

Java

```java
/**
 * Created by zxd on 2022/10/27 15:47
 */
public class Item implements Serializable {
    private Integer id;

    private String name;

    private Float price;

    private String pic;

    private Date createtime;

    private String detail;

    public Item(){

    }

    public Item(Integer id, String name, Float price, String pic, Date cre
atetime, String detail) {
        this.id = id;
        this.name = name;
        this.price = price;
        this.pic = pic;
        this.createtime = createtime;
        this.detail = detail;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name == null ? null : name.trim();
    }
```

```
45
46      public Float getPrice() {
47          return price;
48      }
49
50      public void setPrice(Float price) {
51          this.price = price;
52      }
53
54      public String getPic() {
55          return pic;
56      }
57
58      public void setPic(String pic) {
59          this.pic = pic == null ? null : pic.trim();
60      }
61
62      public Date getCreatetime() {
63          return createtime;
64      }
65
66      public void setCreatetime(Date createtime) {
67          this.createtime = createtime;
68      }
69
70      public String getDetail() {
71          return detail;
72      }
73
74      public void setDetail(String detail) {
75          this.detail = detail == null ? null : detail.trim();
76      }
77  }
78
```

注解	说明
@Results	代替的是标签<resultMap>该注解中可以使用单个@Result注解，也可以使用@Result集合。使用格式：@Results（{@Result（），@Result（）}）或@Results（@Result（））
@Resut	代替了<id>标签和<result>标签 @Result中属性介绍： column：数据库的列名 property：需要装配的属性名 one：需要使用的@One 注解（@Result（one=@One）（）） many：需要使用的@Many 注解（@Result（many=@many）（））
@One （一对一）	代替了<assocation> 标签，是多表查询的关键，在注解中用来指定子查询返回单一对象 @One注解属性介绍： select: 指定用来多表查询的 sqlmapper 使用格式：@Result(column=" “,property=”“,one=@One(select=””))
@Many （多对一）	代替了<collection>标签, 是是多表查询的关键，在注解中用来指定子查询返回对象集合。 使用格式：@Result(property=“”,column=“”,many=@Many(select=“”))

Java

```java
/**
 * Created by zxd on 2022/10/27 15:26
 */
public interface ItemMapper {


    //resultMap
    @Results({@Result(id = true,column = "id",property = "id"),
              @Result(column = "name",property = "name")})
    @Select("select id,name,price,detail,pic,createtime from item where id =#{id}")
    Item findItemById(Integer id);

    @Select("select id,name,price,detail,pic,createtime from item")
    List<Item>  findItemList();

    @Insert("insert into item(name,price,detail,pic,createtime) values(#{name},#{price},#{detail},#{pic},#{createtime})")
    void insertItem(Item item);

    @Update("update item set name=#{name},price=#{price},detail=#{detail},pic=#{pic},createtime=#{createtime} where id=#{id}")
    void updateItem(Item item);

    @Delete("delete from item where id=#{id}")
    void deleteItemById(Integer id);

    @Select("<script> select * from item     <where>" +
            "           <if test=\"name !=null and name!=''\">" +
            "               and `name` like '%${name}%'" +
            "           </if>" +
            "          </where> </script>")
    List<Item> findItemByQueryDto(QueryItemDto queryItemDto);


}
```

```java
**
 * Created by zxd on 2022/10/27 15:44
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath:spring/applicationContext.xml")
public class TestItemMapper {


    @Autowired
    private ItemMapper itemMapper;

    @Test
    public void testInsert(){
        Item item=new Item(null,"华为mate100",18000F,"xx",new Date(),"xxx");
        itemMapper.insertItem(item);
    }

    @Test
    public void testUpdate(){

        Item item=new Item(3,"华为mate60",18000F,"xx",new Date(),"xxx");

        itemMapper.updateItem(item);

    }


    @Test
    public void testFind(){

        System.out.println(itemMapper.findItemById(3));

        System.out.println(itemMapper.findItemList());
    }

    @Test
    public void testQuery(){

        QueryItemDto queryItemDto=new QueryItemDto();
        queryItemDto.setName("华");

        List<Item> itemList = itemMapper.findItemByQueryDto(queryItemDto);

```

```
44          System.out.println(itemList);
45
46      }
47
48
49
50  }
51
```

3.7 整合测试

3.7.1 编写service

编写接口

```java
/**
 * Created by zxd on 2022/10/27 15:59
 */
public interface ItemService {

    /**
     * 查找所有商品
     * @return
     */
    List<Item>  findItemList();

}
```

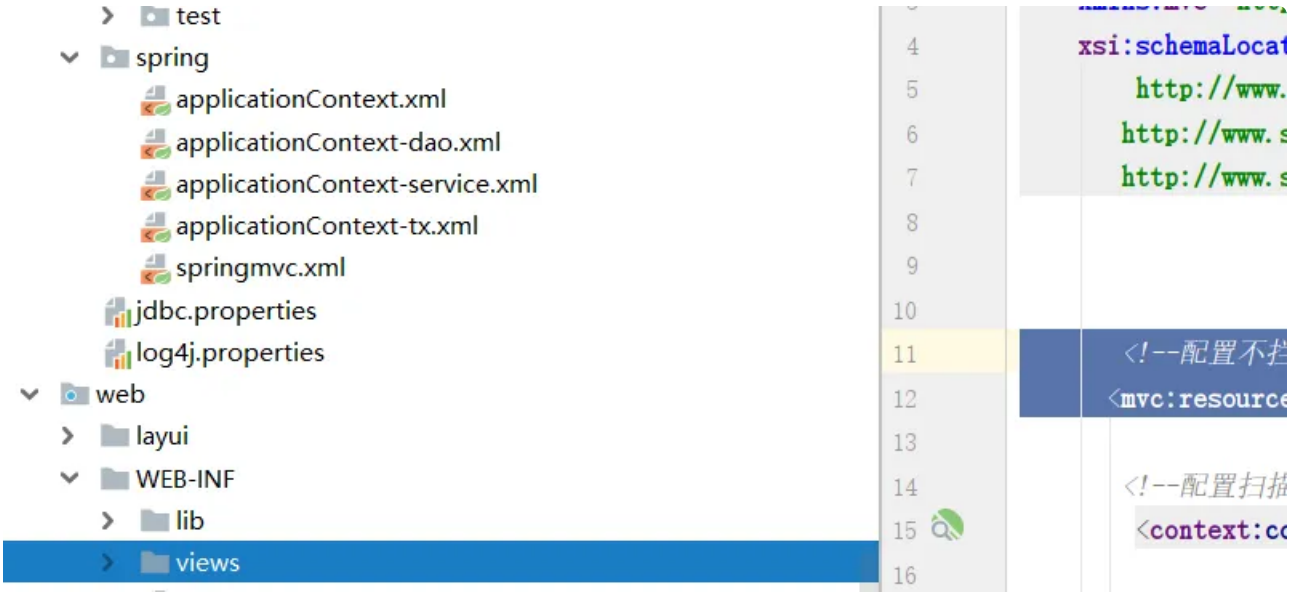
编写实现类

```java
/**
 * Created by zxd on 2022/10/27 16:00
 */
@Service
public class ItemServiceImpl implements ItemService {

    @Autowired
    private ItemMapper itemMapper;

    @Override
    public List<Item> findItemList() {
        return itemMapper.findItemList();
    }
}

```

3.7.2 编写controller

```java
/**
 * Created by zxd on 2022/10/27 16:01
 */
@Controller
public class ItemController {


    @Autowired
    private ItemService itemService;

    @RequestMapping("/queryItemList")
    public ModelAndView queryItemList(){

        List<Item> list = itemService.findItemList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("list",list);
        mv.setViewName("itemList");
        return mv;
    }

}

```

3.7.3 拷贝资源



配置不拦截静态资源

springmvc.xml Java

```
        <!--配置不拦截静态资源-->
    <mvc:resources mapping="/layui/**" location="/layui/"/>

```

4.参数绑定

4.1 默认的参数绑定

需求:

修改商品，查询商品信息 展示商品信息

根据id查询一个商品的信息，展示在修改页面

编写mapper

```java
@Select("select id,name,price,detail,pic,createtime from item where id=#{id}")
Item findItemById(Integer id);
```

编写service

编写接口

```java
/**
 * Created by zxd on 2022/10/27 15:59
 */
public interface ItemService {

    /**
     * 查找所有商品
     * @return
     */
    List<Item>  findItemList();


    /**
     * 根据id查询商品
     * @param id
     * @return
     */
    Item findItemById(Integer id);

}
```

编写实现类

```java
/**
 * Created by zxd on 2022/10/27 16:00
 */
@Service
public class ItemServiceImpl implements ItemService {

    @Autowired
    private ItemMapper itemMapper;

    @Override
    public List<Item> findItemList() {
        return itemMapper.findItemList();
    }

    @Override
    public Item findItemById(Integer id) {
        return itemMapper.findItemById(id);
    }
}

```

编写controller

选择	商品名称	商品价格	生产日期	操作
☐	华为mate60	18000.0	2022-10-27 15:50:16	修改
☐	华为mate60	8000.0	2022-06-27 14:46:26	修改
☐	华为mate100	18000.0	2022-10-27 15:48:50	修改

Java

```java
@RequestMapping("/updateItemUI")
 public ModelAndView updateItemUI(HttpServletRequest request){

     //1.获取请求参数id
     String id = request.getParameter("id");

     //2.调用service进行查询
     Item item = itemService.findItemById(Integer.valueOf(id));

     //3.封装ModelAndView对象
     ModelAndView mv=new ModelAndView();
     mv.addObject("item",item);
     mv.setViewName("editItem");

     return mv;
 }

```

默认的参数绑定

我们只需要在方法上声明HttpServletRequest，处理器适配器自动绑定request对象

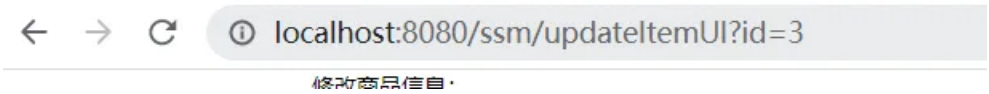
HttpServletRequest response

HttpSession session

```java

@RequestMapping("/updateItemUI")
    public void updateItemUI(HttpServletRequest request, HttpServletResponse response, HttpSession session) throws ServletException, IOException {

        //1.获取请求参数id
        String id = request.getParameter("id");

        //2.调用service进行查询
        Item item = itemService.findItemById(Integer.valueOf(id));


        request.setAttribute("item",item);
        request.getRequestDispatcher("/WEB-INF/views/editItem.jsp").forward(request,response);

    }
```

不推荐直接使用原生对象，因为这样的就没有使用springmvc封装的相关工程

4.2 绑定简单类型



我们可以直接在处理器方法上声明我们需要接收的参数，当请求参数的名字和处理器形参名一致的时候，springmvc会进行自动的参数绑定。

```java
@RequestMapping("/updateItemUI")
public ModelAndView updateItemUI(Integer id){


    //2.调用service进行查询
    Item item = itemService.findItemById(id);

    //3.封装ModelAndView对象
    ModelAndView mv=new ModelAndView();
    mv.addObject("item",item);
    mv.setViewName("editItem");

    return mv;
}

```

整型 Integer int

Long long

字符串 String

单精度 Float float

双精度 Double double

布尔类型 Boolean boolean 请求参数传递 true或者false 1或0

4.3 RequestParam注解(重点)

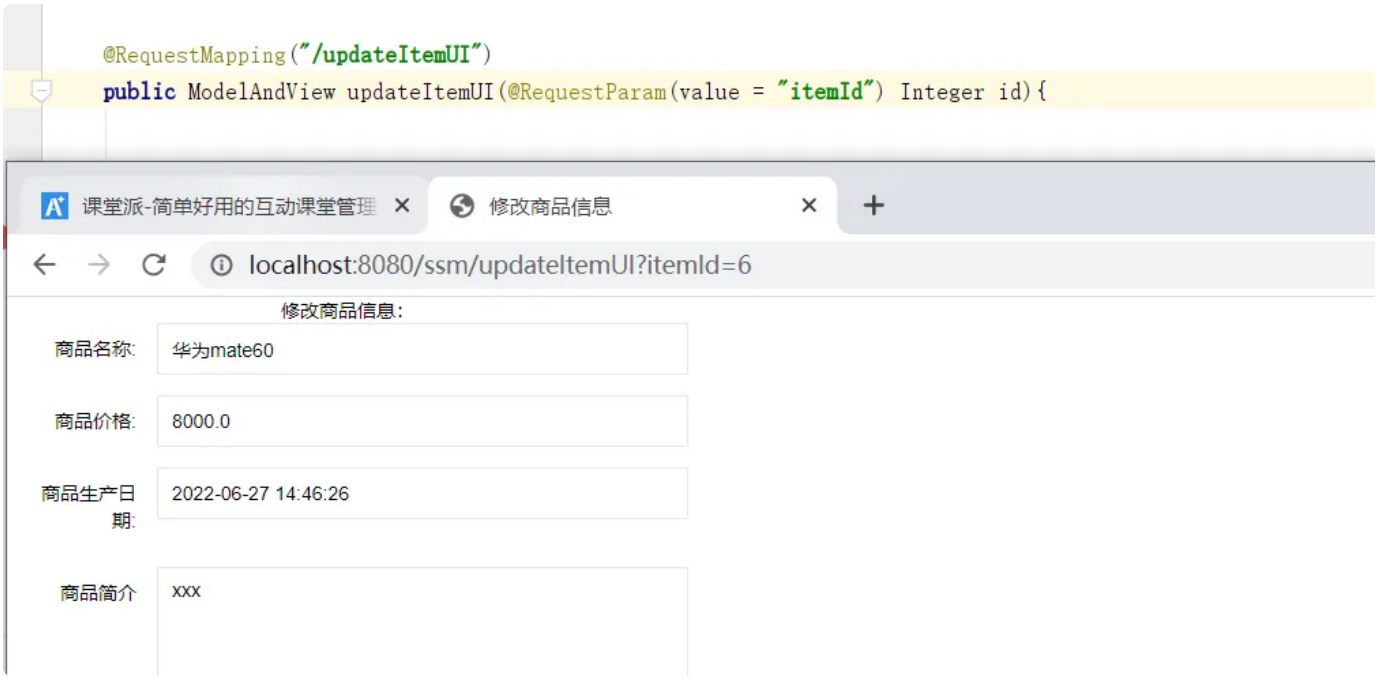
```java
@Target({ElementType.PARAMETER})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface RequestParam {

    //处理器参数名称不一致的情况
    String value() default "";

    //参数是否必填
    boolean required() default true;

    //默认值
    String defaultValue() default "\n\t\t\n\t\t\n\ue000\ue001\ue002\n\t\t\t\t\n";
}
```

@RequestParam 注解有三个属性 三个作用

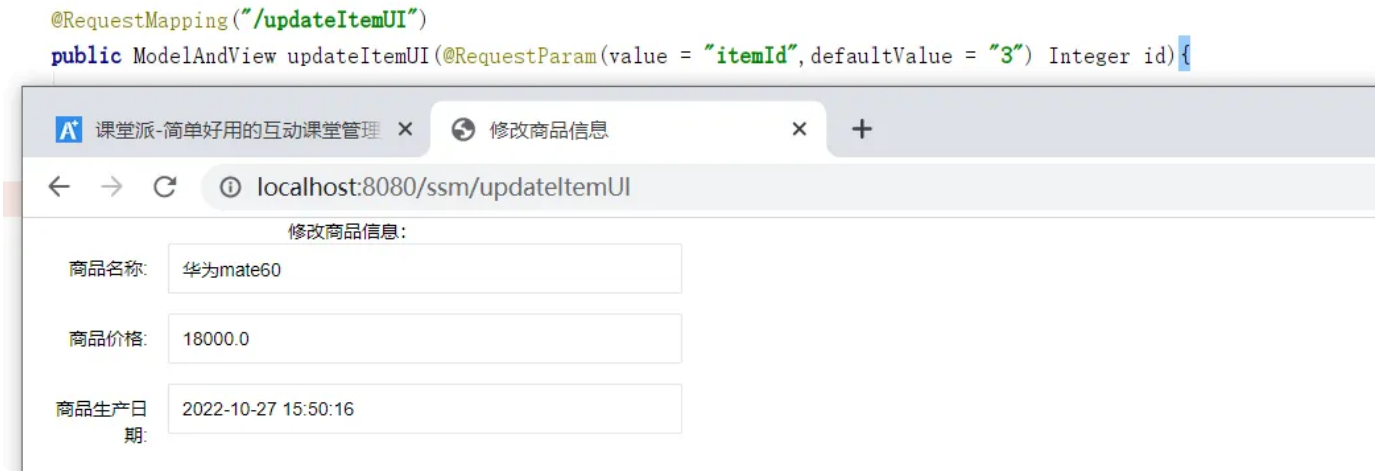
required 参数是否必须 默认为true，没有传递参数就会报错

```java
@RequestMapping("/updateItemUI")
public ModelAndView updateItemUI(@RequestParam Integer id){
```

value 前端传递的参数名称 可以处理器前后端参数不一致的情况 如果一致可以不配置



defaultValue 默认值,如果required 设置为true，代表参数必传，如果配置了默认值，不传递参数使用默认值



4.4 Model/ModelMap

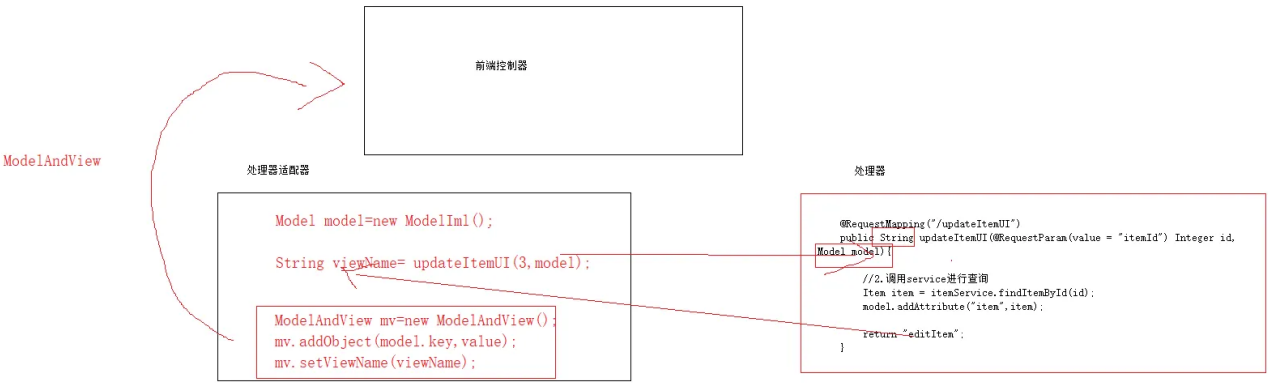
使用Model向request域放置数据，直接在方法的参数上面进行声明即可

Model就是map集合，处理器适配器会传给我一个map集合，我们就可以往map集合设置数据，处理器方法的返回

是String类型，就是视图的名字。

处理器适配器就会自动创建一个ModelAndView对象给前端控制器

```java
    @RequestMapping("/updateItemUI")
    public String updateItemUI(@RequestParam(value = "itemId") Integer id, Model model){

        //2.调用service进行查询
        Item item = itemService.findItemById(id);
        model.addAttribute("item",item);

        return "editItem";
    }
```



ModelMap是一个实现类

```java
    @RequestMapping("/updateItemUI")
      public String updateItemUI(@RequestParam(value = "itemId") Integer id, ModelMap model){

          //2.调用service进行查询
          Item item = itemService.findItemById(id);
          model.addAttribute("item",item);

          return "editItem";
      }
```

4.5 绑定pojo类型

需求: 提交修改请求，修改商品信息

编写mapper

```java
@Update("update item set name=#{name},price=#{price},detail=#{detail},pic=#{pic},createtime=#{createtime} where id=#{id}")
    void updateItem(Item item);
```

编写service

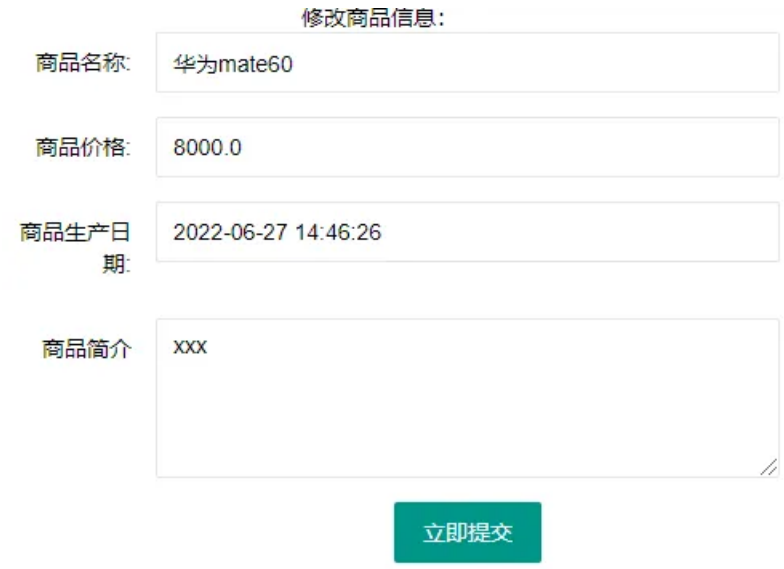
接口

```java
    /**
     * 修改商品
     * @param item
     */
    void updateItem(Item item);
```

编写实现

```java
@Override
    public void updateItem(Item item) {
        itemMapper.updateItem(item);
    }
```

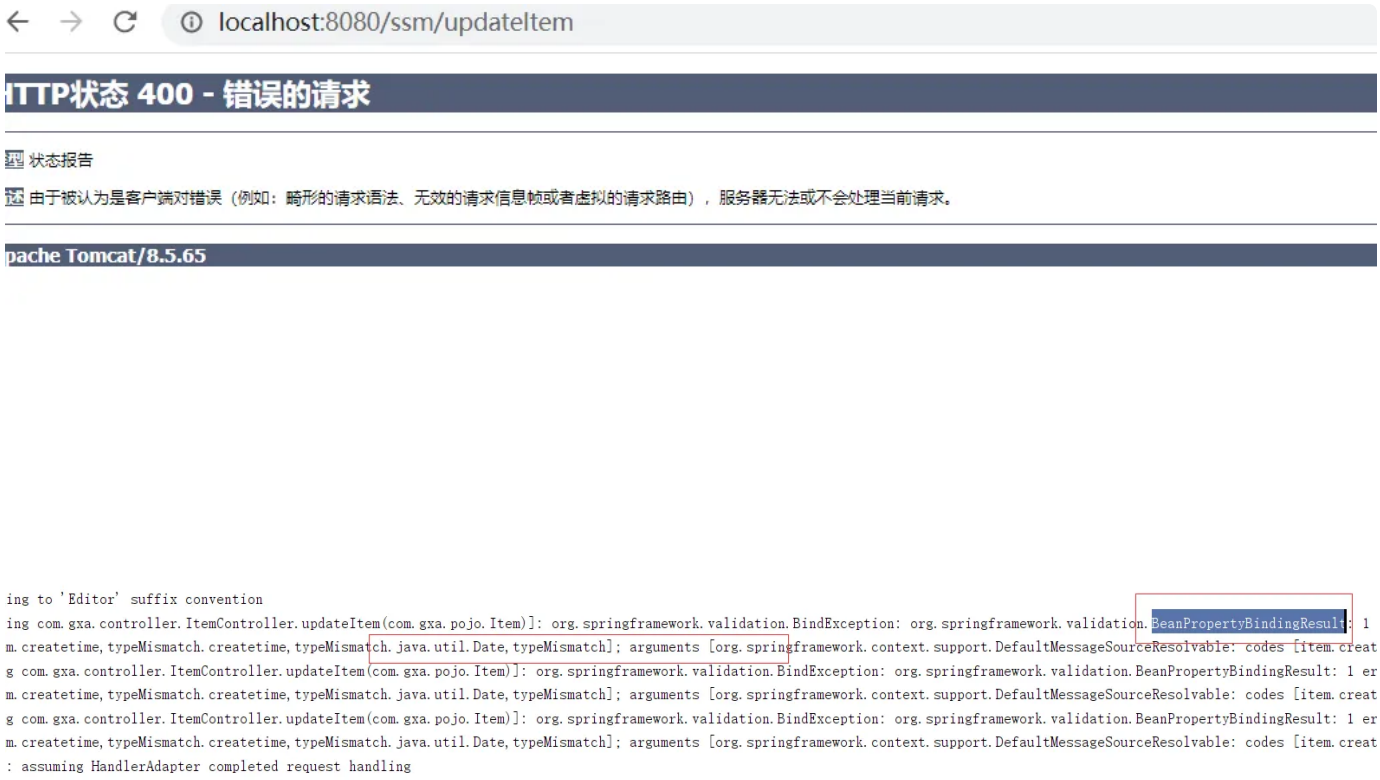
编写controller




```
<body>
    <!-- 上传图片是需要指定属性 enctype="multipart/form-data" -->
    <!-- <form id="itemForm" action="" method="post" enctype="multipart/form-data"> -->
    <form id="itemForm" class="layui-form"
        action="${pageContext.request.contextPath }/updateItem"
            method="post" style="..." align="center">
        <input type="hidden" name="id" value="${item.id }" /> 修改商品信息:


        <div class="layui-form-item">
            <label class="layui-form-label">商品名称:</label>
            <div class="layui-input-block">
                <input type="text" name="name" value="${item.name }"
                    lay-verify="required" placeholder="请输入商品名称" autocomplete="off"
                    class="layui-input" >
            </div>
        </div>
        <div class="layui-form-item">
            <label class="layui-form-label">商品价格:</label>
            <div class="layui-input-block">
                <input type="text" name="price" value="${item.price }"
```

Java

```java
@RequestMapping("/updateItem")
  public String updateItem(Item item){
      itemService.updateItem(item);
      return "success";
  }
```


localhost:8080/ssm/updateItem
HTTP状态 400 - 错误的请求
状态报告
由于被认为是客户端对错误（例如：畸形的请求语法、无效的请求信息帧或者虚拟的请求路由），服务器无法或不会处理当前请求。
pache Tomcat/8.5.65
ing to 'Editor' suffix convention
ing com.gxa.controller.ItemController.updateItem(com.gxa.pojo.Item)]: org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindingResult: 1
m.createtime,typeMismatch.createtime,typeMismatch.java.util.Date,typeMismatch]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [item.creat
g com.gxa.controller.ItemController.updateItem(com.gxa.pojo.Item)]: org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindingResult: 1 er
m.createtime,typeMismatch.createtime,typeMismatch.java.util.Date,typeMismatch]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [item.creat
g com.gxa.controller.ItemController.updateItem(com.gxa.pojo.Item)]: org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindingResult: 1 er
m.createtime,typeMismatch.createtime,typeMismatch.java.util.Date,typeMismatch]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [item.creat
: assuming HandlerAdapter completed request handling

4.6 编写日期转换器

```java

/**
 * Created by zxd on 2022/10/28 10:37
 */
public class DateConverter implements Converter<String, Date> {

    @Override
    public Date convert(String source) {
        try{

            SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            Date date = df.parse(source);
            return date;
        }catch (Exception e){
            e.printStackTrace();
            throw new RuntimeException("日转换失败!");
        }
    }
}

```

配置

```Java
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">



       <!--配置不拦截静态资源-->
      <mvc:resources mapping="/layui/**" location="/layui/"/>

       <!--配置扫描controller包-->
        <context:component-scan base-package="com.gxa.controller"/>

        <!--配置注解驱动-->
         <mvc:annotation-driven conversion-service="conversionService"/>

        <!--配置日期转换器-->
        <bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
            <property name="converters">
                <set>
                    <bean class="com.gxa.converter.DateConverter"/>
                </set>
            </property>
        </bean>


        <!--配置视图解析器-->
        <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <property name="prefix" value="/WEB-INF/views/"/>
            <property name="suffix" value=".jsp"/>
        </bean>


</beans>
```

4.7 解决乱码问题

```Java
    <!--配置解决乱码的过滤器-->
     <filter>
         <filter-name>characterEncodingFilter</filter-name>
         <filter-class>org.springframework.web.filter.CharacterEncodingFilt
er</filter-class>
         <init-param>
             <param-name>encoding</param-name>
             <param-value>UTF-8</param-value>
         </init-param>
     </filter>

     <filter-mapping>
         <filter-name>characterEncodingFilter</filter-name>
         <url-pattern>/*</url-pattern>
     </filter-mapping>

```

4.8 绑定包装的javabean类型（了解）

需求：根据id和商品名称进行查询

```java
/**
 * Created by zxd on 2022/10/28 10:52
 */
public class QueryDto {

    private Item item;

    public Item getItem() {
        return item;
    }

    public void setItem(Item item) {
        this.item = item;
    }

}
```

后端接收

```java
    @RequestMapping("/queryItemListByWhere")
     public String queryItemListByWhere(QueryDto queryDto){
         System.out.println(queryDto);
         return "success";
     }
```

```
<form action="${pageContext.request.contextPath }/queryItemListByWhere"
    method="post">


    <table >

                        <h6>查询条件: </h6>
        <div class="layui-form-item">
            <label class="layui-form-label">商品id:</label>
            <div class="layui-input-block">
                <input type="text" name="item.id" required lay-verify="required"
                    placeholder="请输入标题" autocomplete="off" class="layui-input" style="width:200px">
            </div>
        </div>
        <div class="layui-form-item">
            <label class="layui-form-label">商品名称:</label>
            <div class="layui-input-block">
                <input type="text" name="item.name" required lay-verify="required"
                    placeholder="请输入标题" autocomplete="off" class="layui-input" style="width:200px">
            </div>
        </div>

        <tr>
            <td><input type="submit" class="layui-btn layui-btn-normal"
                value="查询" /></td>
```

```
59
60      @RequestMapping("/queryItemListByWhere")
61      public String queryItemListByWhere(QueryDto queryDto){   queryDto:
62          System.out.println(queryDto);   queryDto: QueryDto@5849
63          return "success";
64
```

Evaluate

Expression:

queryDto

Result:

oo result = {QueryDto@5849}

f item = {Item@5876} "Item{id=1, name='ddd', price=null, pic='null', createtime=

4.9 绑定集合或者数组

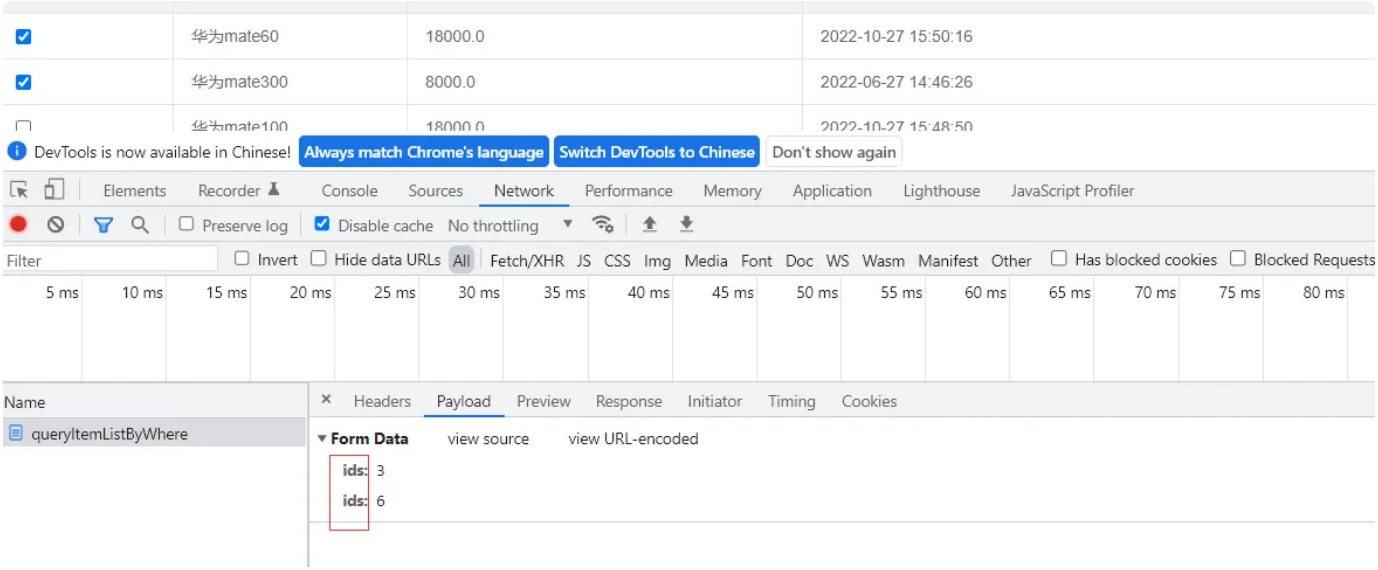
```java

/**
 * Created by zxd on 2022/10/28 10:52
 */
public class QueryDto {

    private Item item;

    private List<Integer>  ids;

    public List<Integer> getIds() {
        return ids;
    }

    public void setIds(List<Integer> ids) {
        this.ids = ids;
    }

    public Item getItem() {
        return item;
    }

    public void setItem(Item item) {
        this.item = item;
    }

}

```

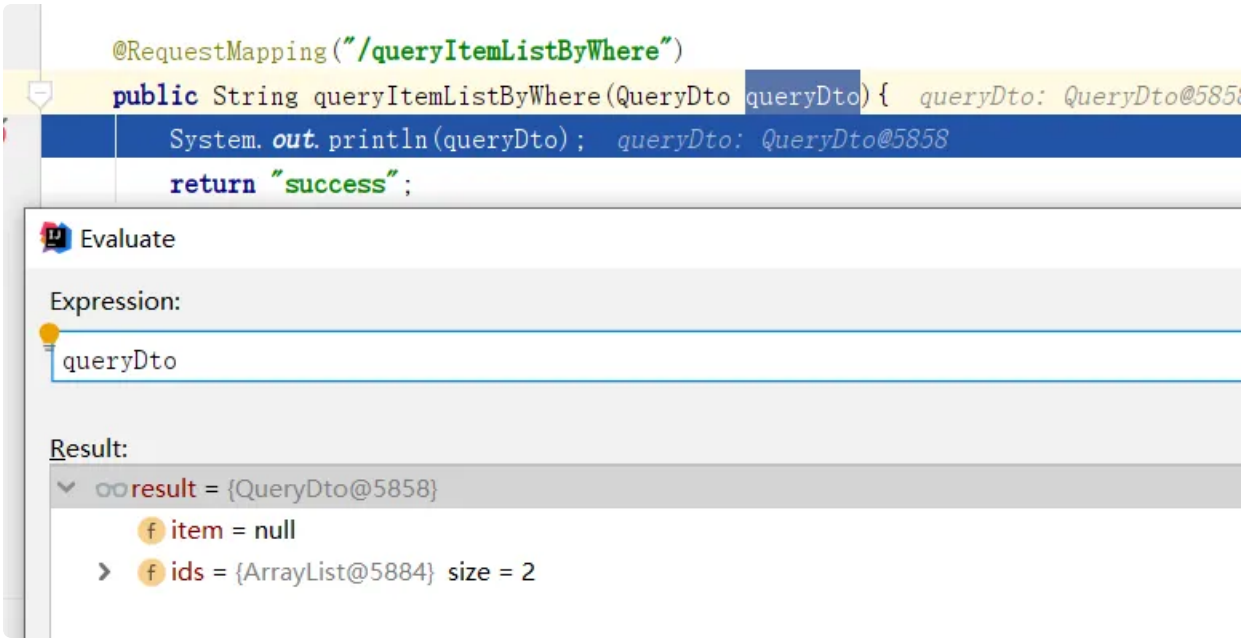
修改前端

```
body>
    <form action="${pageContext.request.contextPath }/queryItemListByWhere"
        method="post">
        <table >
        <%--
                                <h6>查询条件：</h6>
            <div class="layui-form-item">
                <label class="layui-form-label">商品id:</label>
                <div class="layui-input-block">
                    <input type="text" name="item.id" required lay-verify="required"
                        placeholder="请输入标题" autocomplete="off" class="layui-input" style="width:200px">
                </div>
            </div>
            <div class="layui-form-item">
                <label class="layui-form-label">商品名称:</label>
                <div class="layui-input-block">
                    <input type="text" name="item.name" required lay-verify="required"
                        placeholder="请输入标题" autocomplete="off" class="layui-input" style="width:200px">
                </div>
            </div>
    --%>
            <tr>
                <td><input type="submit" class="layui-btn layui-btn-normal"
                    value="批量删除" /></td>
                <td>商品名称</td>
                <td>商品价格</td>
                <td>生产日期</td>
                <td>操作</td>
            </tr>
        </thead>
        <tbody>
            <c:forEach items="${list}" var="item">
                <tr>
                    <td><input type="checkbox" name="ids" value="${item.id}"></td>
                    <td>${item.name }</td>
                    <td>${item.price }</td>
```

后端

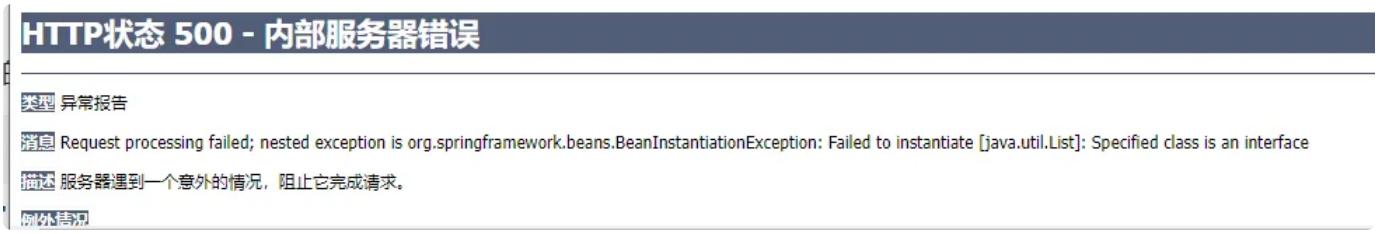
```java
@RequestMapping("/queryItemListByWhere")
public String queryItemListByWhere(QueryDto queryDto){
    System.out.println(queryDto);
    return "success";
}
```





list集合不能直接声明在方法的参数中，需要用javabean类型包装装起来

```java
@RequestMapping("/queryItemListByWhere")
    public String queryItemListByWhere(List<Integer>  ids){
        System.out.println(ids);
        return "success";
    }

```



可以直接在参数上直接绑定数组

```java
@RequestMapping("/queryItemListByWhere")
    public String queryItemListByWhere(Integer[] ids){
        System.out.println(ids);
        return "success";
    }
```

5.RequestMapping

RequestMapping注解可以配置不同处理器的映射路径

5.1 URL映射路径

一个处理器可以有多个映射路径

```java

    @RequestMapping(value = {"/queryItemList","/list"})
    public ModelAndView queryItemList(){

        List<Item> list = itemService.findItemList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("list",list);
        mv.setViewName("itemList");
        return mv;
    }
```

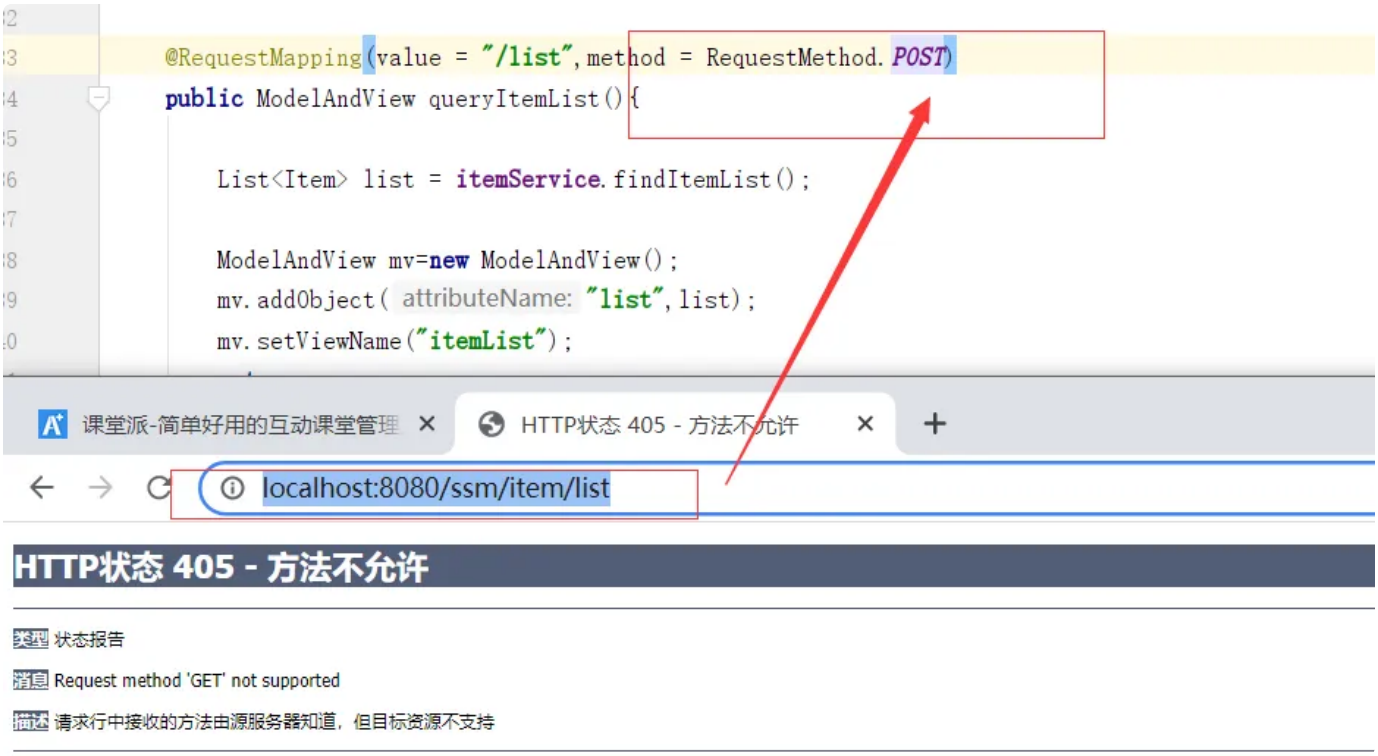
不能出现一个映射路径出现在多个处理器上面

5.2 作用在类上面

如果添加在类上面，以后访问这个类下面所有的处理器加上这个前缀

```java
/**
 * Created by zxd on 2022/10/27 16:01
 */
@Controller
@RequestMapping("/item")
public class ItemController {


    @Autowired
    private ItemService itemService;

    @RequestMapping(value = {"/queryItemList","/list"})
    public ModelAndView queryItemList(){

        List<Item> list = itemService.findItemList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("list",list);
        mv.setViewName("itemList");
        return mv;
    }
```

5.3 限制请求方式



如果不配置就代表不限制请求方式

6.Controller方法的返回

6.1 返回ModelAndView

```java
    @RequestMapping(value = "/list",method = RequestMethod.GET)
      public ModelAndView queryItemList(){

          List<Item> list = itemService.findItemList();

          ModelAndView mv=new ModelAndView();
          mv.addObject("list",list);
          mv.setViewName("itemList");
          return mv;
      }
```

6.2 返回void

```java
    @RequestMapping("/updateItemUI")
     public void updateItemUI(HttpServletRequest request, HttpServletRespon
se response, HttpSession session) throws ServletException, IOException {

          //1.获取请求参数id
          String id = request.getParameter("id");

          //2.调用service进行查询
          Item item = itemService.findItemById(Integer.valueOf(id));


          request.setAttribute("item",item);
          request.getRequestDispatcher("/WEB-INF/views/editItem.jsp").forwar
d(request,response);

      }
```

6.3 返回String类型

6.3.1 逻辑视图的名字

转发到jsp

```java
    @RequestMapping("/updateItemUI")
    public String updateItemUI(@RequestParam(value = "itemId") Integer id, ModelMap model){

        //2.调用service进行查询
        Item item = itemService.findItemById(id);
        model.addAttribute("item",item);

        return "editItem";
    }
```

6.3.2 转发到其他处理器

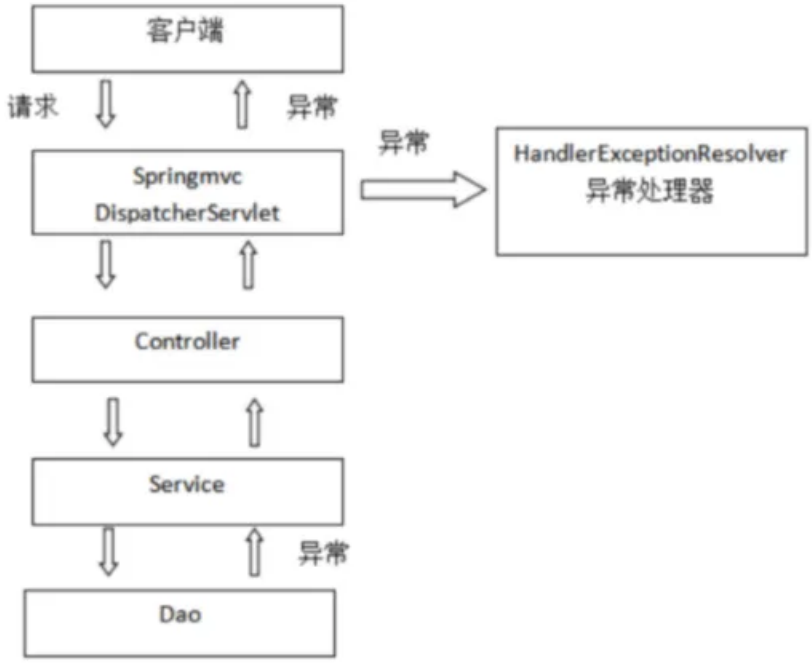
```java
  @RequestMapping("/updateItem")
    public String updateItem(Item item){
        itemService.updateItem(item);
        return "forward:/item/list";
    }
```

6.3.3 重定向

```java
   @RequestMapping("/updateItem")
    public String updateItem(Item item){
        itemService.updateItem(item);
        return "redirect:/item/list";
    }
```

7.异常处理器

7.1 springmvc异常处理



7.2 预期异常(自定义异常)

预期异常：自定义异常

不可预期异常 未知异常 系统繁忙，请待会再来！

7.3 编写自定义异常类

```java

/**
 * Created by zxd on 2022/10/28 14:04
 */
public class CustomerException extends RuntimeException {

    //定义异常信息
    private String message;

    public CustomerException(String message) {
        super(message);
        this.message = message;
    }

    @Override
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}

```

7.4 编写异常处理器

```java
/**
 * Created by zxd on 2022/10/28 14:08
 */
public class MyExceptionHandler implements HandlerExceptionResolver {
    @Override
    public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler, Exception exception) {


        String msg="";

        if(exception instanceof CustomerException){
            //判断是否自定义异常
            CustomerException customerException= (CustomerException) exception;
            msg=customerException.getMessage();
        }else{
            exception.printStackTrace();
            msg="对不起，系统繁忙,请明天再来!";
        }

        ModelAndView modelAndView=new ModelAndView();
        modelAndView.addObject("msg",msg);
        modelAndView.setViewName("error");

        return modelAndView;
    }
}

```

配置

```Java
<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:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">



       <!--配置不拦截静态资源-->
      <mvc:resources mapping="/layui/**" location="/layui/"/>

       <!--配置扫描controller包-->
        <context:component-scan base-package="com.gxa.controller"/>

        <!--配置注解驱动-->
         <mvc:annotation-driven conversion-service="conversionService"/>

        <!--配置日期转换器-->
        <bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
            <property name="converters">
                <set>
                    <bean class="com.gxa.converter.DateConverter"/>
                </set>
            </property>
        </bean>


        <!--配置视图解析器-->
        <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <property name="prefix" value="/WEB-INF/views/"/>
            <property name="suffix" value=".jsp"/>
        </bean>


    <!--配置异常处理器-->
    <bean class="com.gxa.exception.MyExceptionHandler"></bean>

```

```
40
41
    </beans>
```

7.5 测试异常处理

Java

```java
@RequestMapping(value = "/list")
    public ModelAndView queryItemList(){

        if(true){
            throw new CustomerException("用户名密码错误!");
        }

        List<Item> list = itemService.findItemList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("list",list);
        mv.setViewName("itemList");
        return mv;
    }

```

8.图片上传

8.1 nginx

8.1.1 nginx是什么

HTTP和反向代理web服务器

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的，公开版本1.19.6发布于2020年12月15日。[12]

其将源代码以类BSD许可证的形式发布，因它的稳定性、丰富的功能集、简单的配置文件和低系统资源的消耗而闻名。2022年01月25日，nginx 1.21.6发布。[13]

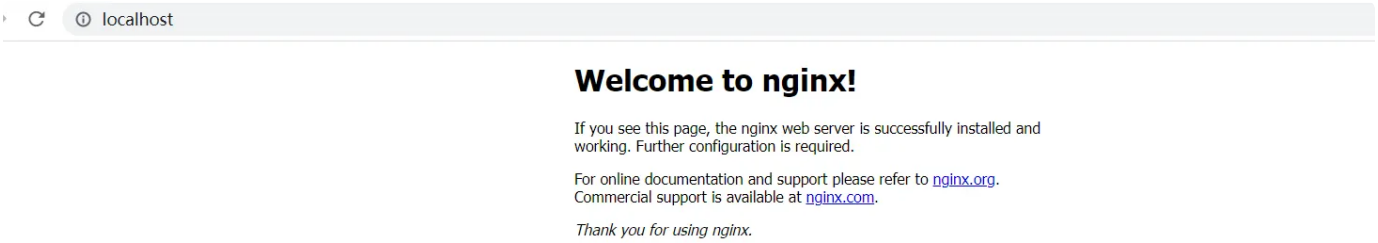
Nginx是一款轻量级的Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，在BSD-like 协议下发行。其特点是占有内存少，并发能力强，事实上nginx的并发能力在同类型的网页服务器中表现较好。

单机并发5发

nginx可以做http服务器，但是nginx只能作为静态资源服务器的

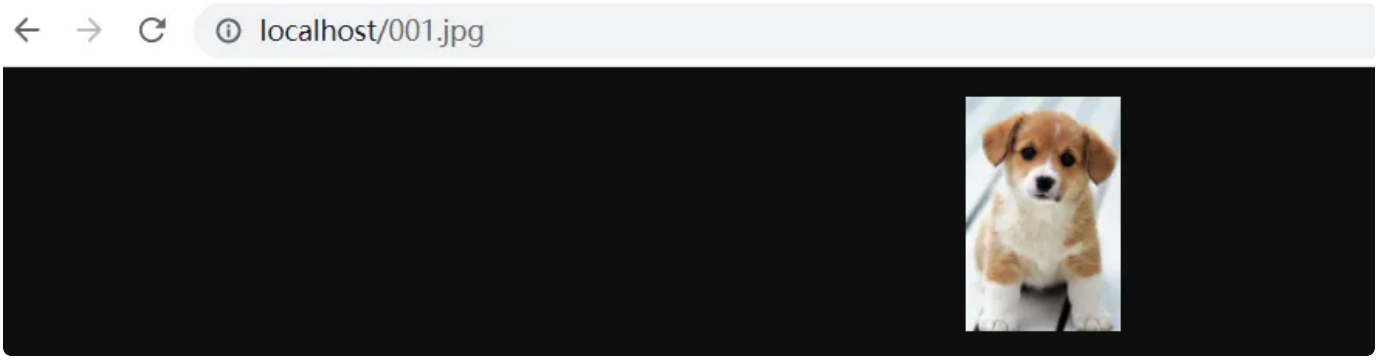
nginx可以做反向代理服务器

8.1.2 nginx做HTTP服务器

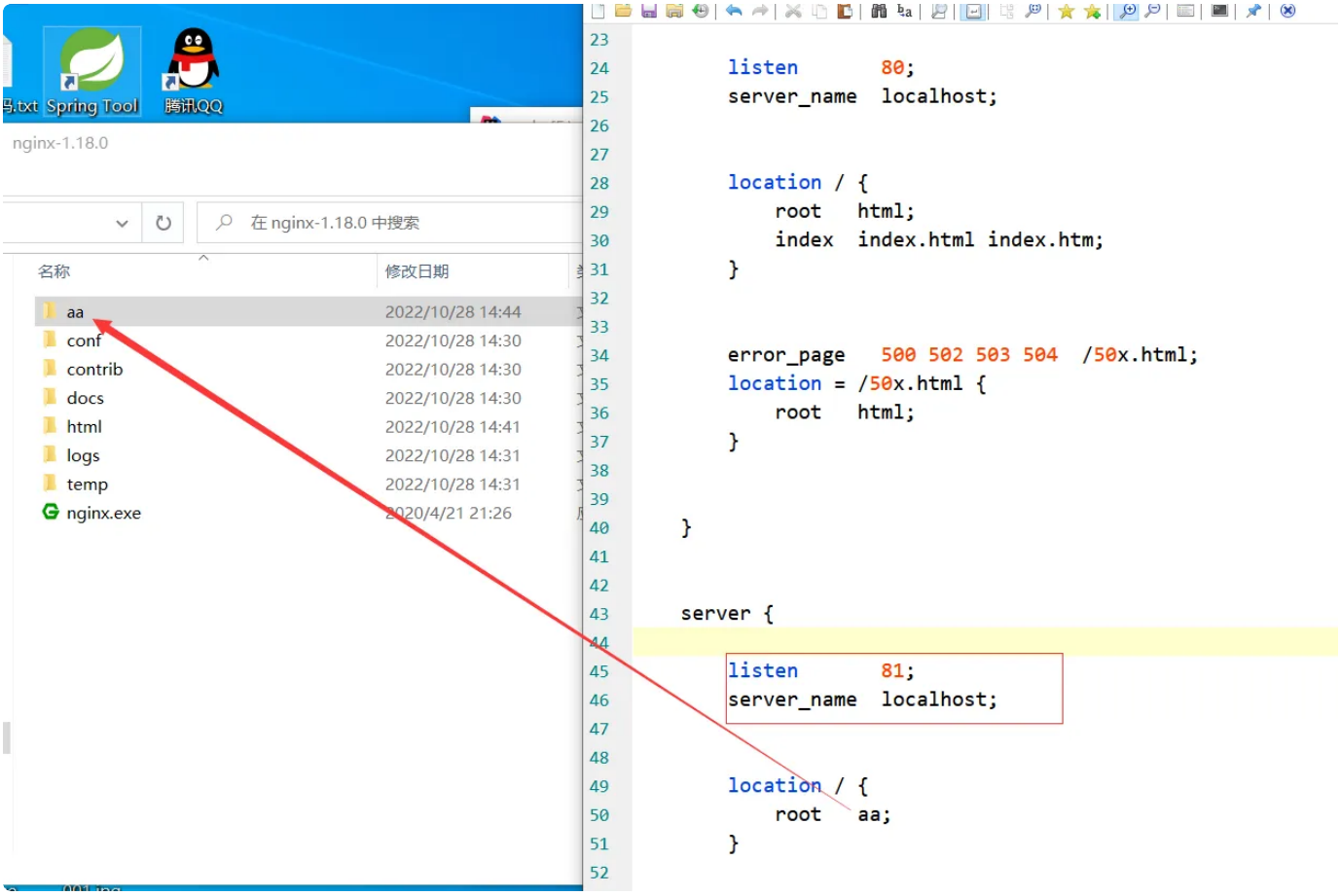


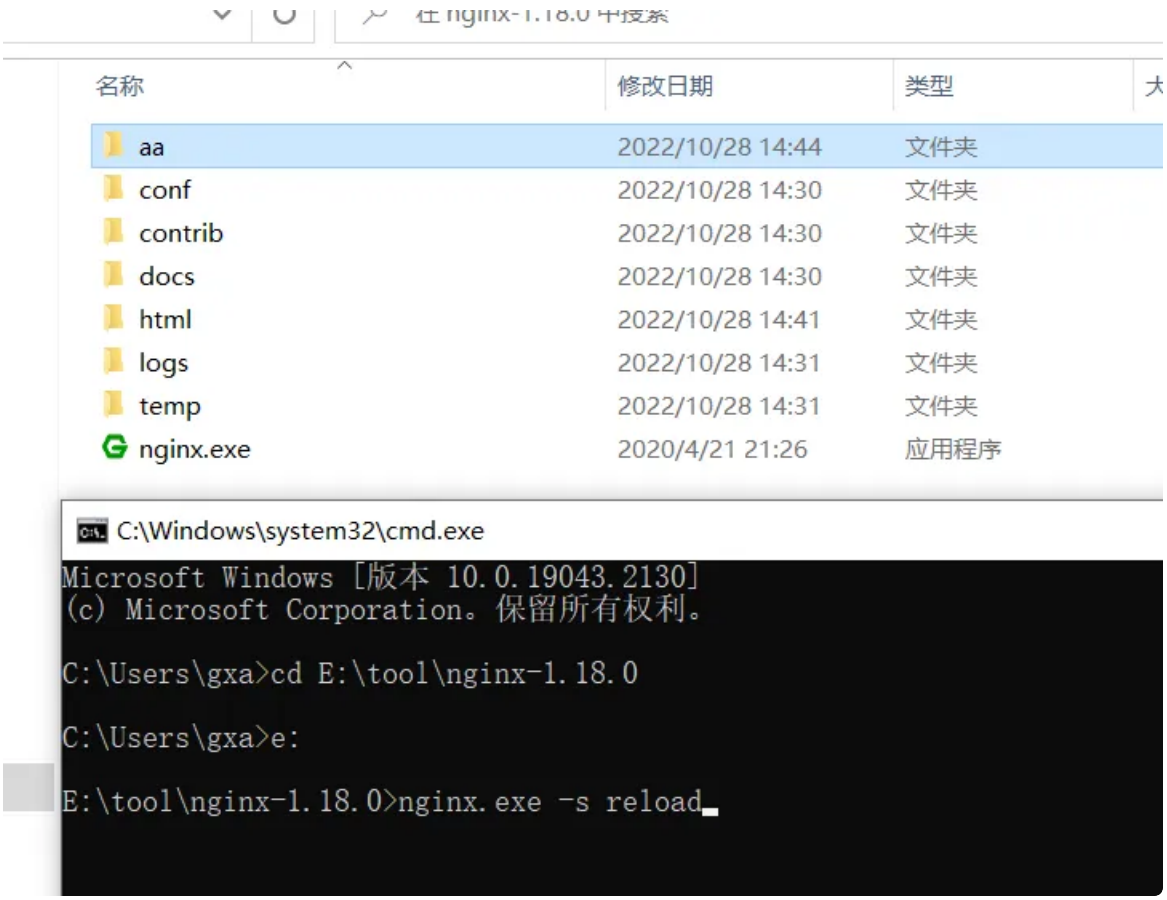
8.1.3 通过端口来区分虚拟主机

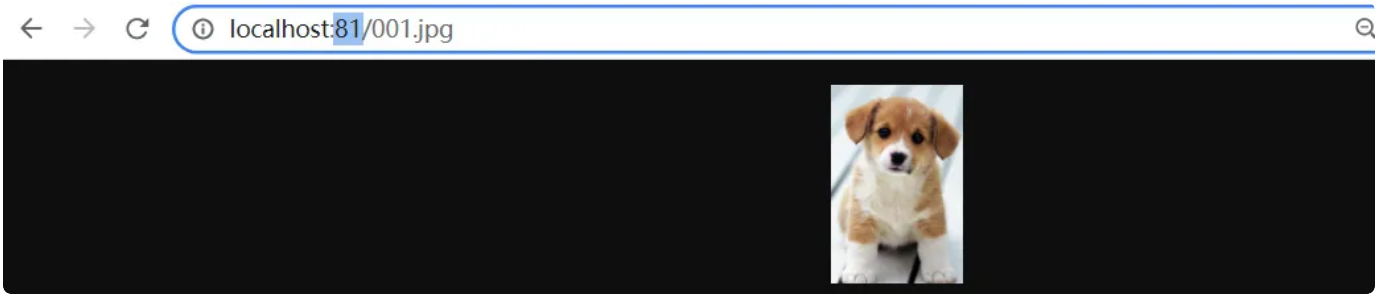
nginx中一个server节点就是一个虚拟主机



nginx可以通过端口来区分同时虚拟出多台服务器



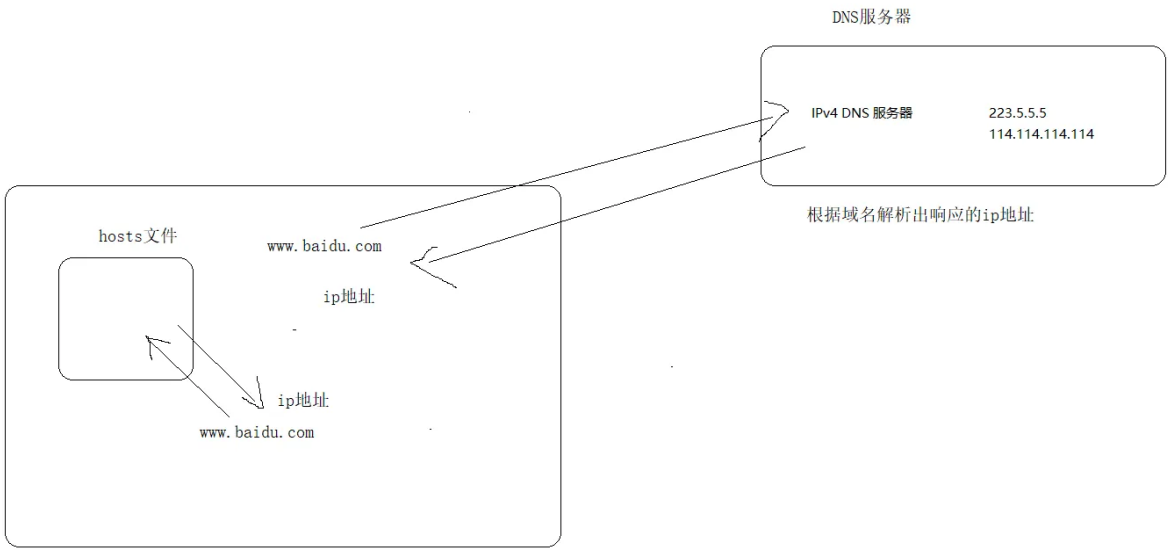




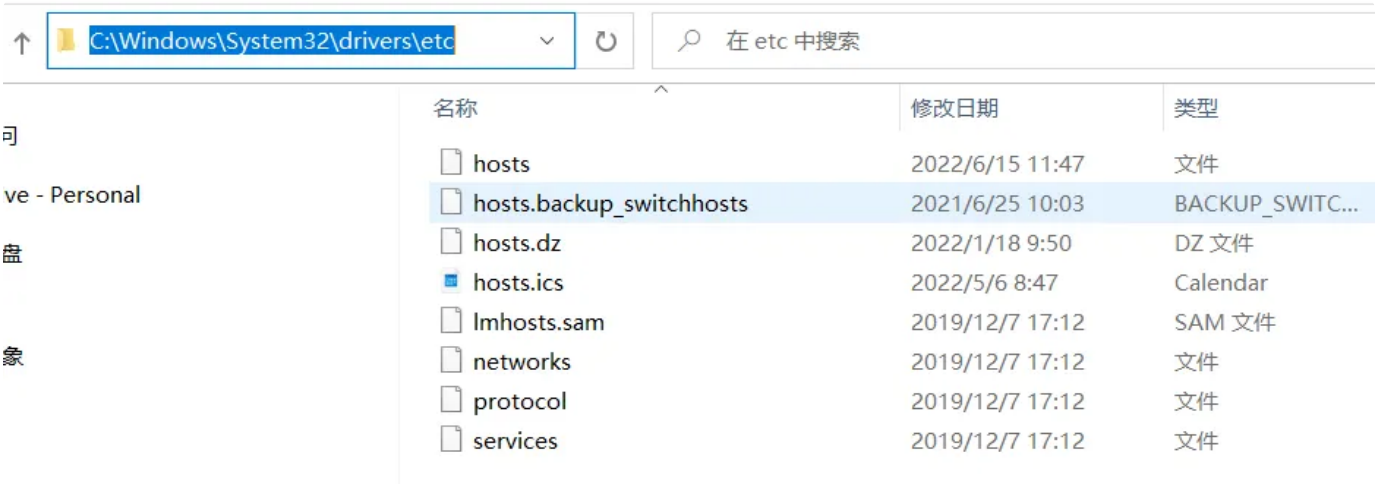
8.1.4 nginx可以通过域名来区分

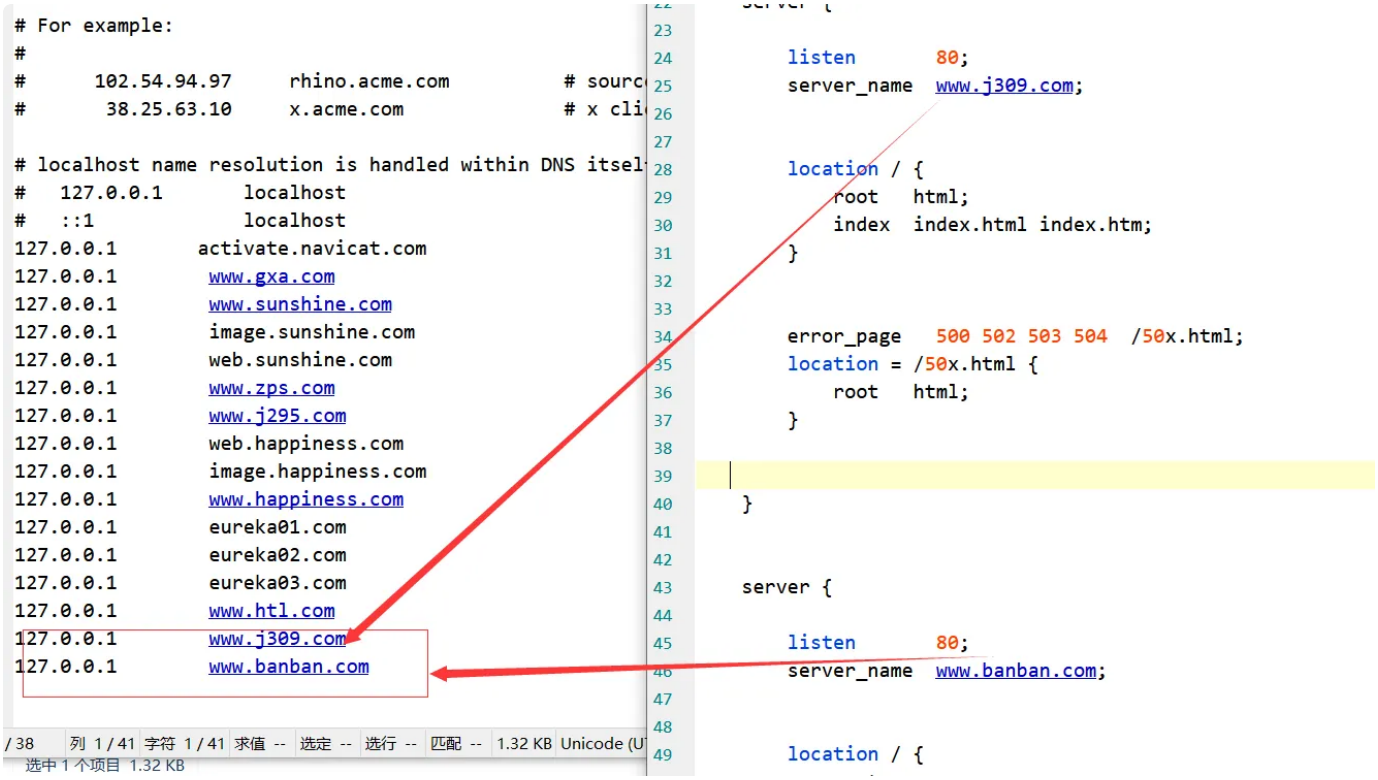
```
    server {

        listen       80;
        server_name  www.j309.com;


        location / {
            root   html;
            index  index.html index.htm;
        }


        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }


    }


    server {

        listen       80;
        server_name  www.banban.com;
```

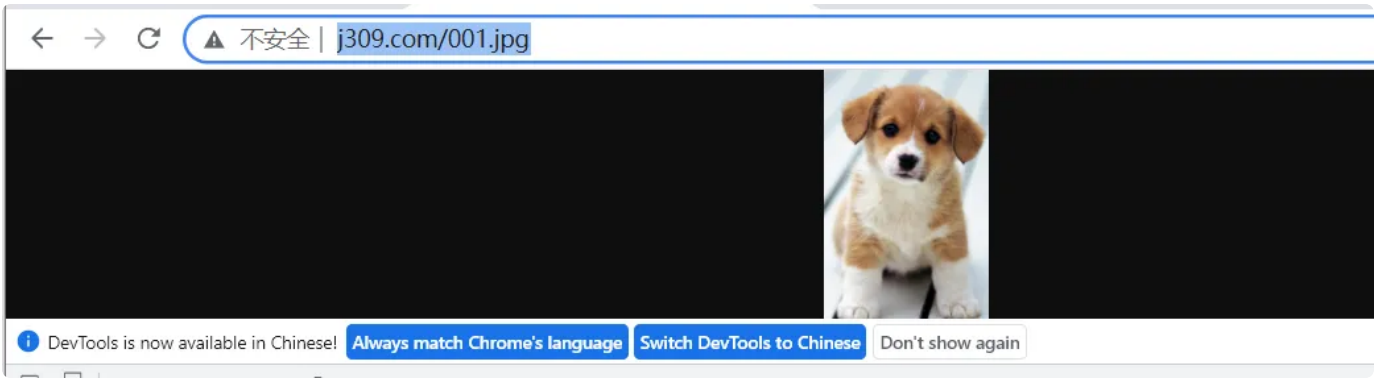
8.1.5 dns解析


DNS服务器
IPv4 DNS 服务器
223.5.5.5
114.114.114.114
根据域名解析出响应的ip地址
hosts文件
www.baidu.com
ip地址
ip地址
www.baidu.com

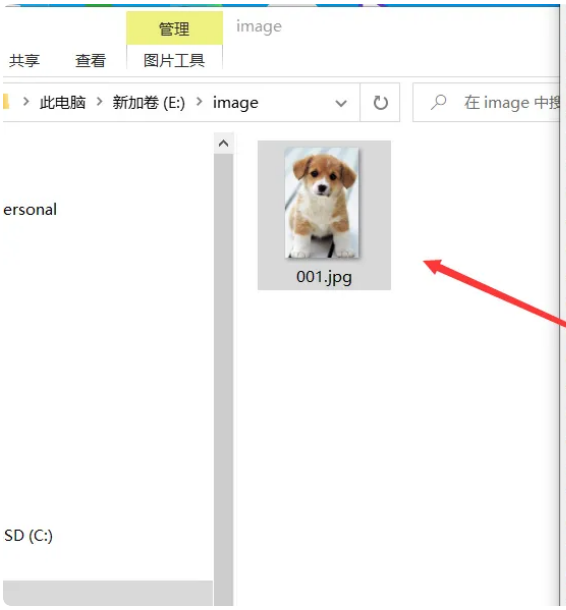
配置hosts文件


C:\Windows\System32\drivers\etc
在 etc 中搜索
名称
修改日期
类型
hosts
2022/6/15 11:47
文件
hosts.backup_switchhosts
2021/6/25 10:03
BACKUP_SWITC...
hosts.dz
2022/1/18 9:50
DZ 文件
hosts.ics
2022/5/6 8:47
Calendar
lmhosts.sam
2019/12/7 17:12
SAM 文件
networks
2019/12/7 17:12
文件
protocol
2019/12/7 17:12
文件
services
2019/12/7 17:12
文件

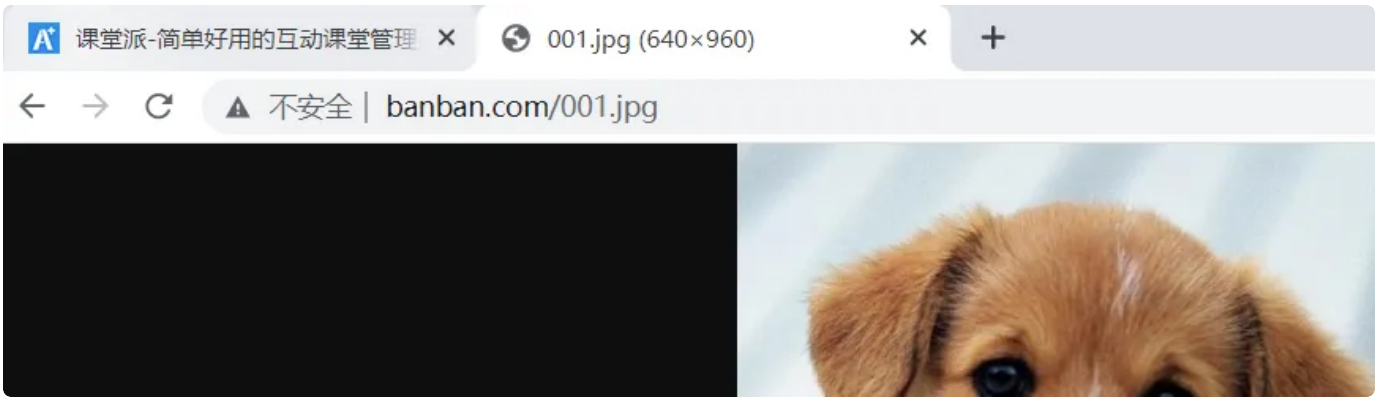

For example:
#
102.54.94.97   rhino.acme.com   # sourc
38.25.63.10   x.acme.com   # x cli
localhost name resolution is handled within DNS itsel
127.0.0.1   localhost
::1   localhost
127.0.0.1   activate.navicat.com
127.0.0.1   www.gxa.com
127.0.0.1   www.sunshine.com
127.0.0.1   image.sunshine.com
127.0.0.1   web.sunshine.com
127.0.0.1   www.zps.com
127.0.0.1   www.j295.com
127.0.0.1   web.happiness.com
127.0.0.1   image.happiness.com
127.0.0.1   www.happiness.com
127.0.0.1   eureka01.com
127.0.0.1   eureka02.com
127.0.0.1   eureka03.com
127.0.0.1   www.htl.com
127.0.0.1   www.j309.com
127.0.0.1   www.banban.com
/ 38 列 1 / 41 字符 1 / 41 求值 -- 选定 -- 选行 -- 匹配 -- 1.32 KB Unicode (U
选中 1 个项目 1.32 KB
listen 80;
server_name www.j309.com;
location / {
root html;
index index.html index.htm;
}
error_page 500 502 503 504 /50x.html;
location = /50x.html {
root html;
}
}
server {
listen 80;
server_name www.banban.com;
location / {

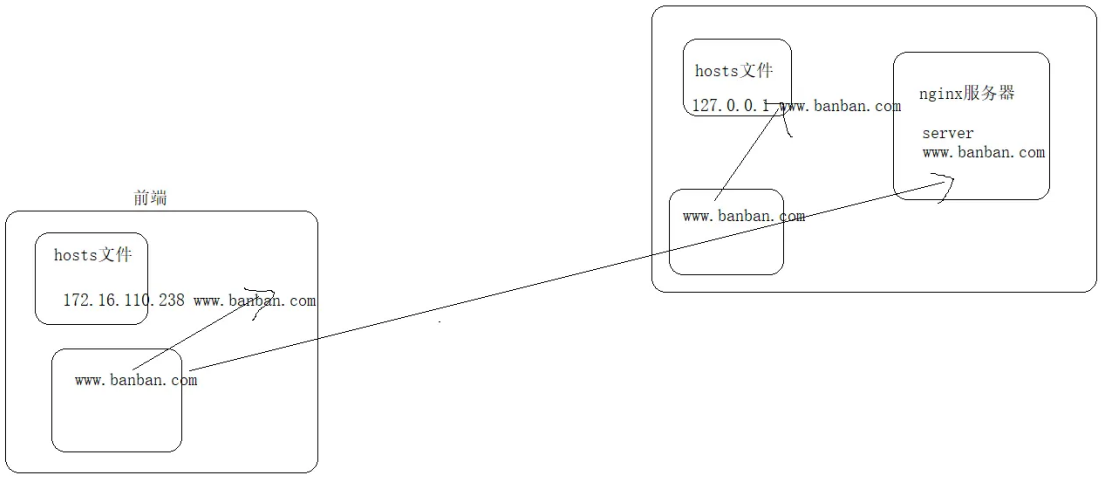

不安全 | j309.com/001.jpg
DevTools is now available in Chinese!
Always match Chrome's language
Switch DevTools to Chinese
Don't show again

8.1.6 搭建图片服务器



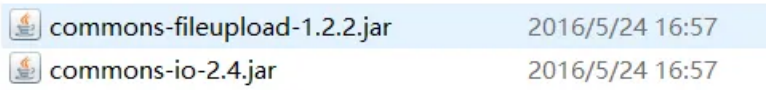

```
34      error_page   500 502 503 504  /50x.html;
35      location = /50x.html {
36          root   html;
37      }
38
39
40  }
41
42
43  server {
44
45      listen       80;
46      server_name  www.banban.com;
47
48
49      location / {
50          root   E:\image;
51      }
52
53  }
54
```





8.2 图片上传

导入jar包



修改jsp

```
<body>
    <!-- 上传图片是需要指定属性 enctype="multipart/form-data" -->
    <!-- <form id="itemForm" action="" method="post" enctype="multipart/form-data"> -->
    <form id="itemForm" class="layui-form"  enctype="multipart/form-data"
        action="${pageContext.request.contextPath }/item/updateItem"
            method="post" style="..." align="center">
        <input type="hidden" name="id" value="${item.id }" /> 修改商品信息:


        <div class="layui-form-item">
            <label class="layui-form-label">商品名称:</label>
            <div class="layui-input-block">
                <input type="text" name="name" value="${item.name }"
                    lay-verify="required" placeholder="请输入商品名称" autocomplete="off"
                    class="layui-input" >
            </div>
        </div>
        <div class="layui-form-item">
            <label class="layui-form-label">商品价格:</label>
            <div class="layui-input-block">
                <input type="text" name="price" value="${item.price }"
```

```
    </div>
    <div class="layui-form-item">
        <label class="layui-form-label">商品图片:</label>
        <div class="layui-input-block">
            <c:if test="${item.pic !=null}">
                <img src="${item.pic}" width=100 height=100 />
                <br />
            </c:if>
            <input  type="file" name="pictureFile" value="${item.pic }"
                lay-verify="required" placeholder="请输入商品生产时间" autocomplete="off"
                class="layui-input">
        </div>
    </div>
    <div class="layui-form-item layui-form-text">
```

编写修改的controller方法

```java
    //MultipartFile 封装了上传文件
    @RequestMapping("/updateItem")
    public String updateItem(Item item,MultipartFile pictureFile) throws IOException {


        //1.获取原来的名字
        String originalFilename = pictureFile.getOriginalFilename();

        //2.获取后缀   .jpg
        String extName=originalFilename.substring(originalFilename.lastIndexOf("."));

        //3.生成新的名字
        String picName= UUID.randomUUID().toString().replace("-","")+extName;

        //4.把图片保存到指定的磁盘中
        pictureFile.transferTo(new File("E:\\image"+File.separator+picName));

        //5.保存图片的访问路径
        item.setPic("http://www.banban.com/"+picName);

        itemService.updateItem(item);


        return "redirect:/item/list";
    }
```

配置

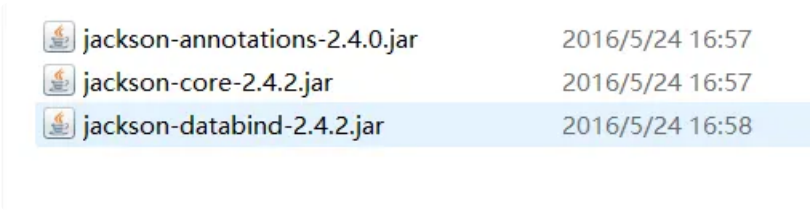
springmvc.xml

```java
    <!--配置上传的解析器-->
    <!--id必须设置为multipartResolver-->
    <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <!--配置上传文件大小-->
        <property name="maxUploadSize" value="6291456"/>
    </bean>
```

9.Json数据的交互

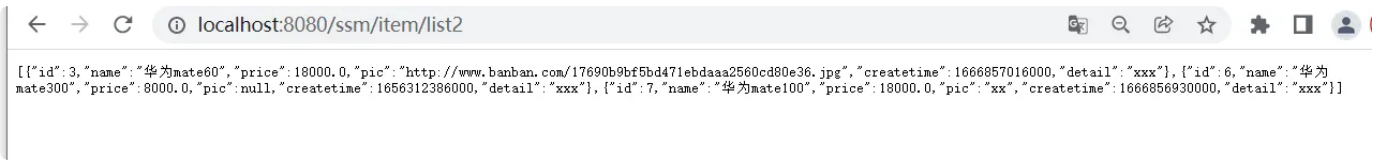
9.1 ResponseBody注解

导包



ResponseBody注解表示这个方法返回json数据

```java
    @RequestMapping(value = "/list2")
     @ResponseBody  //返回json数据
     public List<Item> queryItemList2(){
         List<Item> list = itemService.findItemList();
         return list;
     }

```



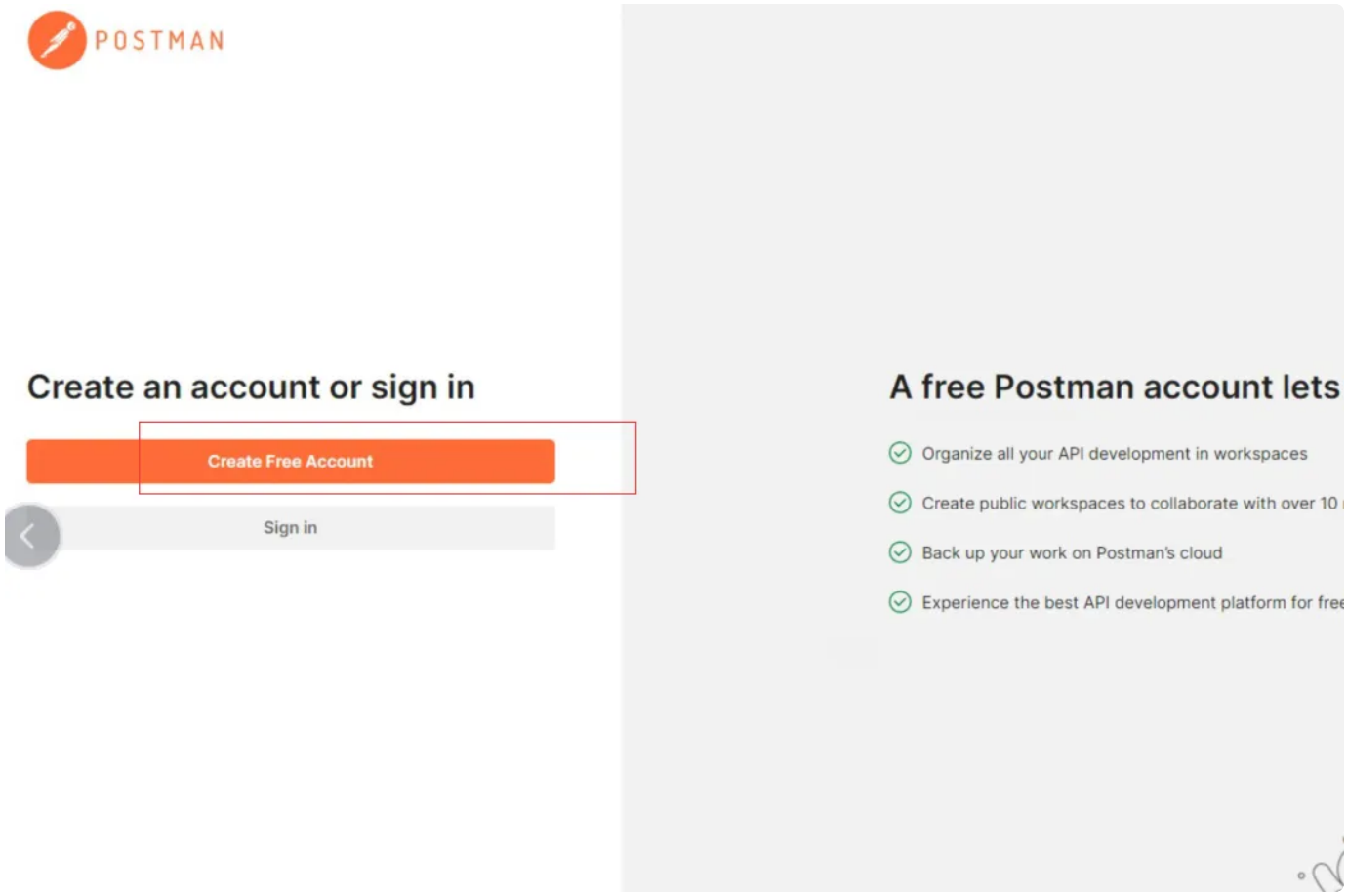
@ResponseBody作用在类上面，代表整个类的所有方法都返回json数据

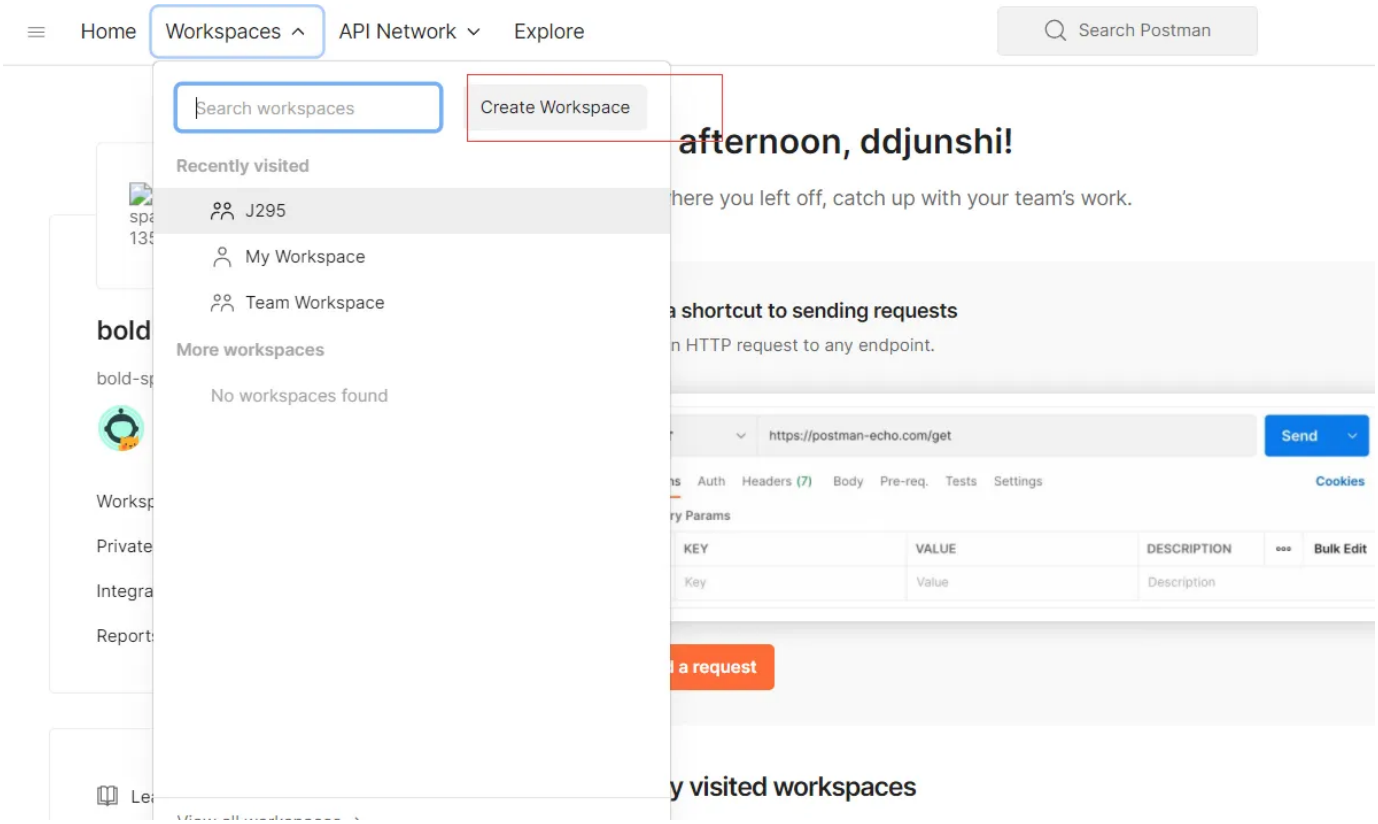
```java
@Controller
@RequestMapping("/item")
@ResponseBody
public class ItemController {

    @RequestMapping(value = "/list2")
    public List<Item> queryItemList2(){
        List<Item> list = itemService.findItemList();
        return list;
    }


}
```

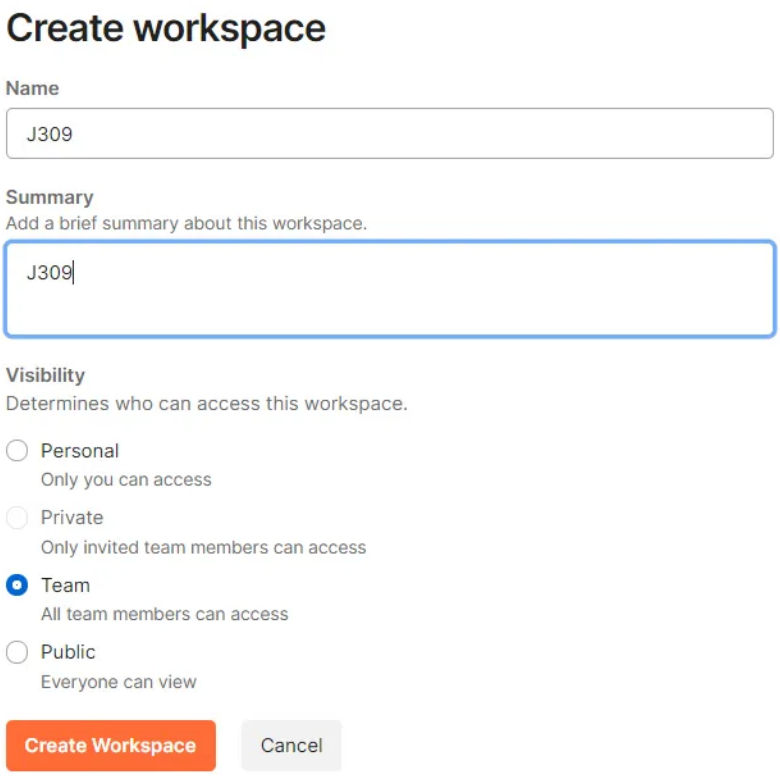
9.2 ResponseEntity

```java
   @RequestMapping(value = "/list3")
    //ResponseEntity 告诉springmvc返回json数据
    public ResponseEntity<List<Item>> queryItemList3(){
        List<Item> list = itemService.findItemList();
        //return ResponseEntity.status(201).body(list);
        return ResponseEntity.ok(list);
    }
```

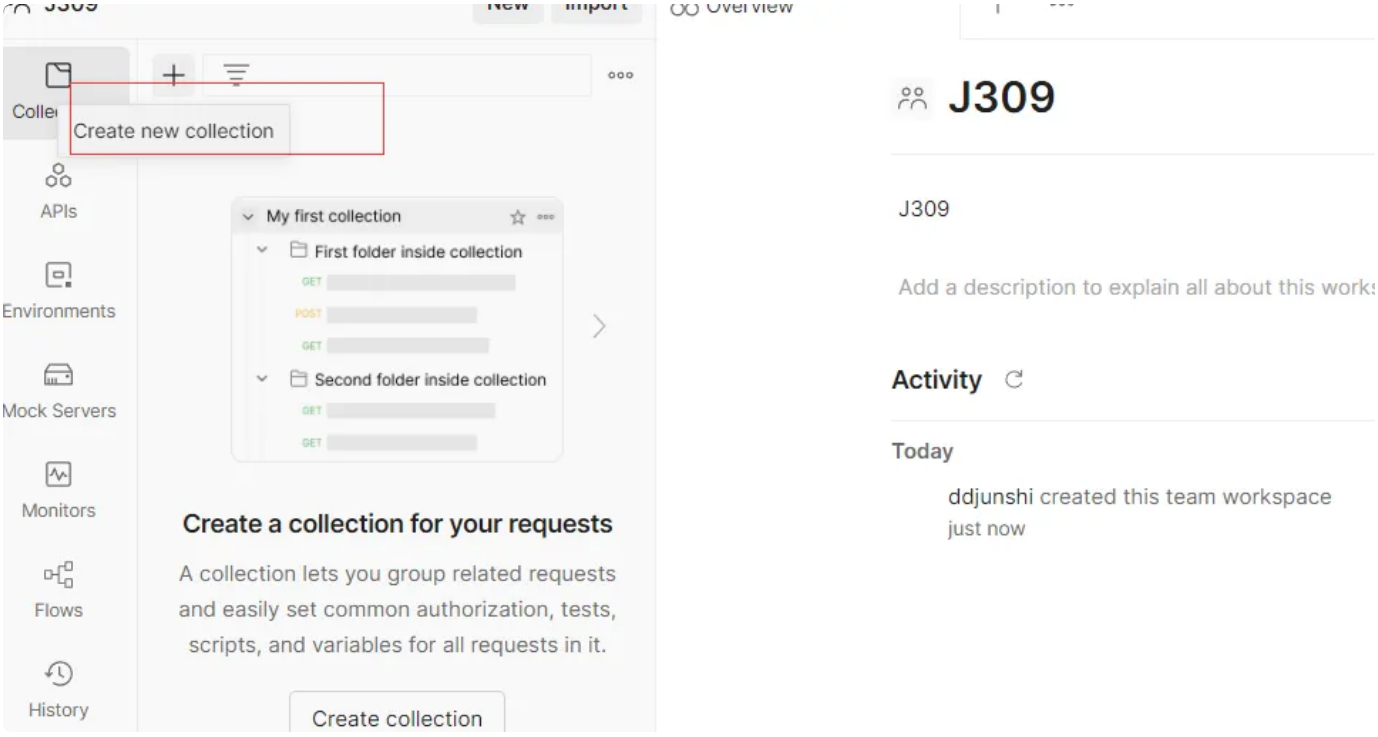
9.3 postman使用



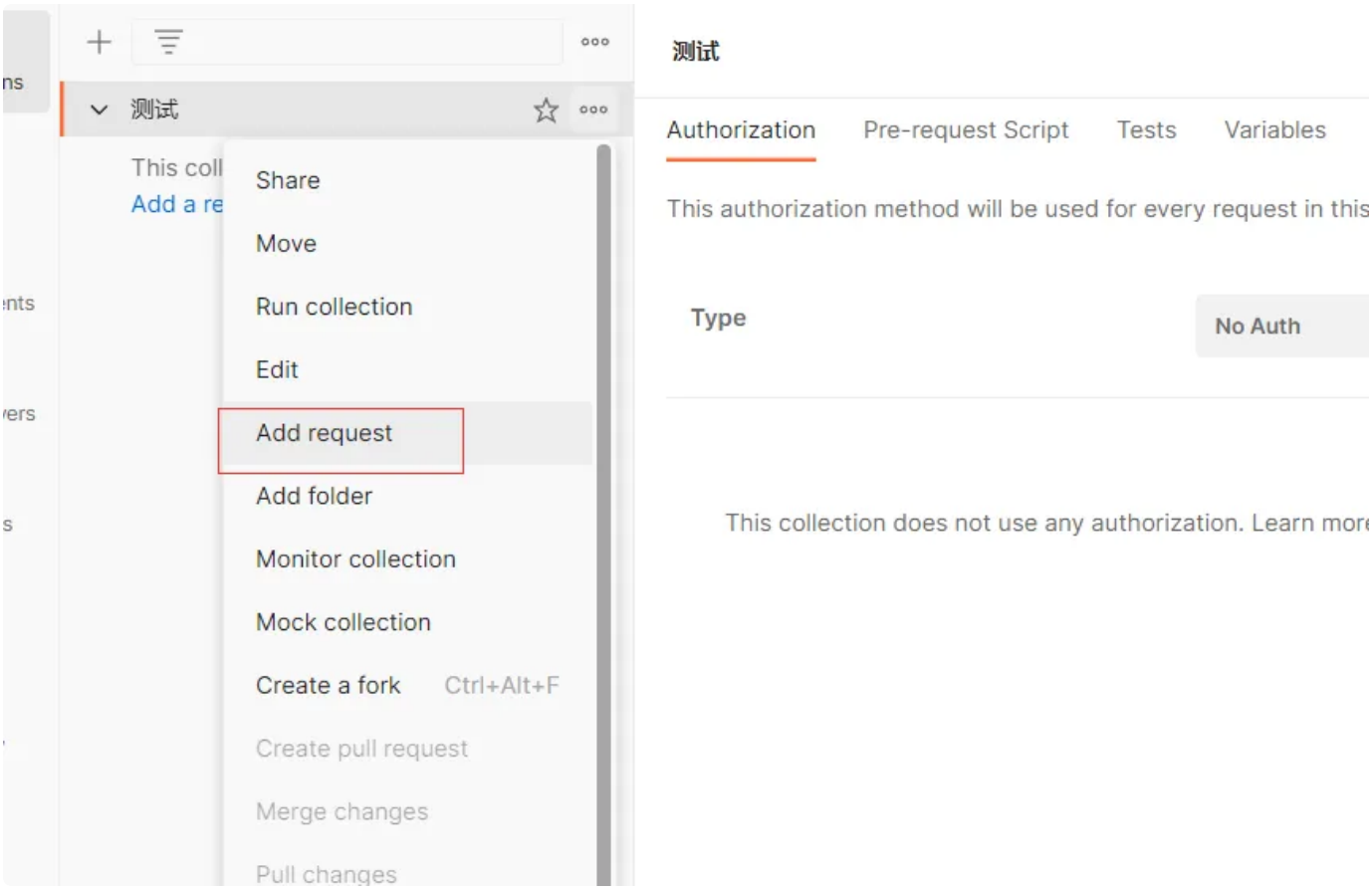

Home
Workspaces
API Network
Explore
Search Postman
Search workspaces
Create Workspace
Recently visited
J295
My Workspace
Team Workspace
More workspaces
No workspaces found
afternoon, ddjunshi!
here you left off, catch up with your team's work.
a shortcut to sending requests
n HTTP request to any endpoint.
https://postman-echo.com/get
Send
Auth
Headers (7)
Body
Pre-req.
Tests
Settings
Cookies
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
a request
y visited workspaces


Create workspace
Name
J309
Summary
Add a brief summary about this workspace.
J309
Visibility
Determines who can access this workspace.
Personal
Only you can access
Private
Only invited team members can access
Team
All team members can access
Public
Everyone can view
Create Workspace
Cancel

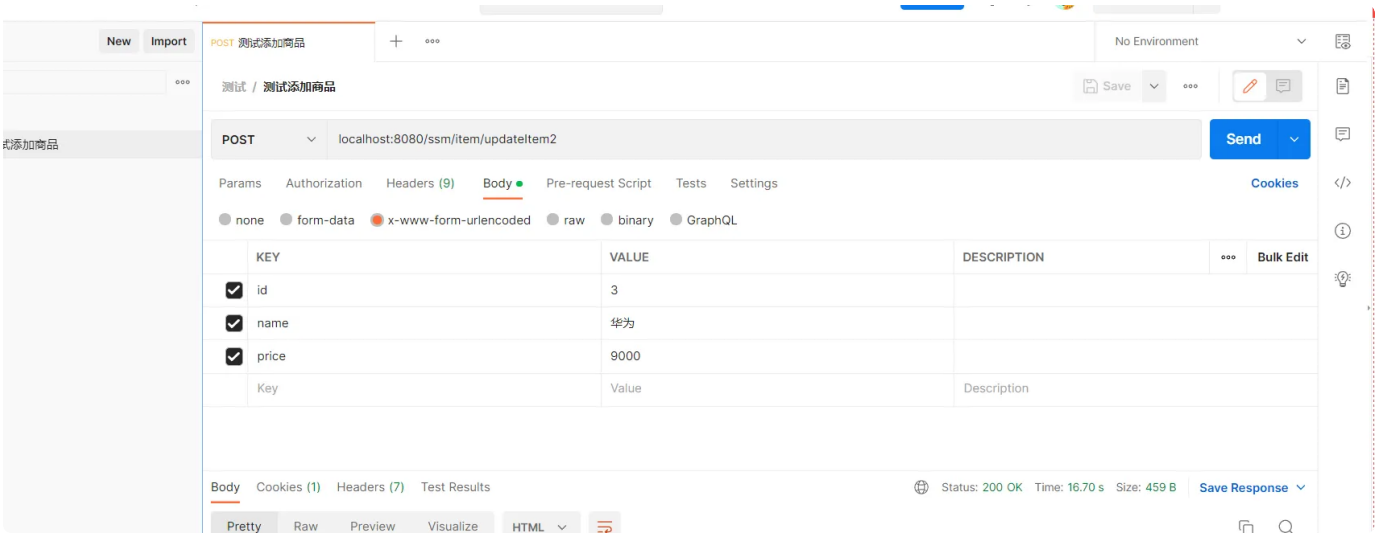
新建文件夹


Overview
Create new collection
APIs
Environments
Mock Servers
Monitors
Flows
History
My first collection
First folder inside collection
Second folder inside collection
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create collection
J309
J309
Add a description to explain all about this work
Activity
Today
ddjunshi created this team workspace
just now

新建请求


测试
Share
Move
Run collection
Edit
Add request
Add folder
Monitor collection
Mock collection
Create a fork Ctrl+Alt+F
Create pull request
Merge changes
Pull changes
测试
Authorization
Pre-request Script
Tests
Variables
This authorization method will be used for every request in this
Type
No Auth
This collection does not use any authorization. Learn more

测试发送普通表单


New
Import
POST 测试添加商品
No Environment
测试 / 测试添加商品
Save
POST
localhost:8080/ssm/item/updateItem2
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
id
3
name
华为
price
9000
Key
Value
Description
Body
Cookies (1)
Headers (7)
Test Results
Status: 200 OK
Time: 16.70 s
Size: 459 B
Save Response
Pretty
Raw
Preview
Visualize
HTML

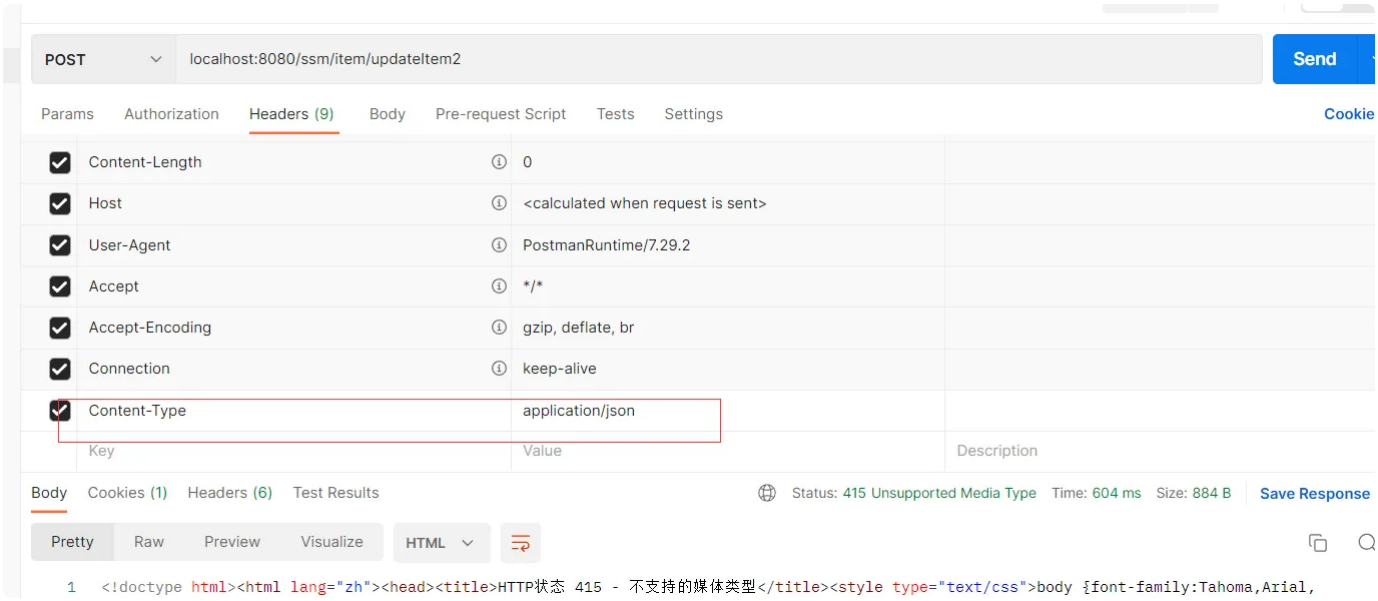
```java
    @RequestMapping(value = "/updateItem2",method = RequestMethod.POST)
    public String update2(Item item){
        System.out.println(item);
        return "success";
    }
```

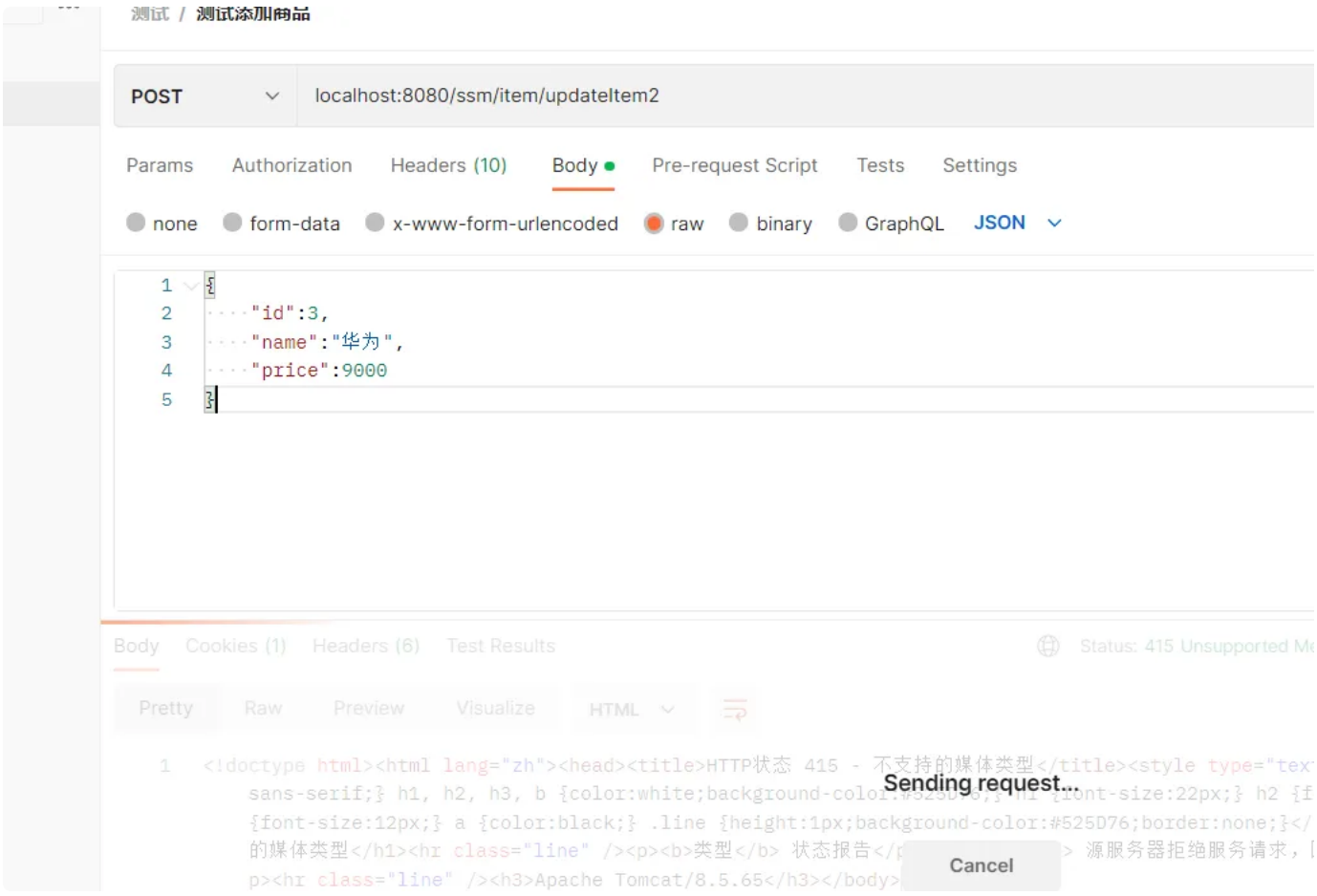
9.4 RequestBody注解

接收前端的json数据

```java
    @RequestMapping(value = "/updateItem2",method = RequestMethod.POST)
    //@RequestBody 前端给后端发送json数据
    public String update2(@RequestBody Item item){
        System.out.println(item);
        return "success";
    }
```

前端需要在请求头 Content-Type 设置为application/json 告诉后端我现在给你发的是json


POST
localhost:8080/ssm/item/updateItem2
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookie
Content-Length
0
Host
<calculated when request is sent>
User-Agent
PostmanRuntime/7.29.2
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
Content-Type
application/json
Key
Value
Description
Body
Cookies (1)
Headers (6)
Test Results
Status: 415 Unsupported Media Type
Time: 604 ms
Size: 884 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 <!doctype html><html lang="zh"><head><title>HTTP状态 415 - 不支持的媒体类型</title><style type="text/css">body {font-family:Tahoma,Arial,


测试 / 测试添加商品
POST
localhost:8080/ssm/item/updateItem2
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
1 {
2 "id":3,
3 "name":"华为",
4 "price":9000
5 }
Body
Cookies (1)
Headers (6)
Test Results
Pretty
Raw
Preview
Visualize
HTML
Sending request...
Cancel