# 基本概念

Spring MVC 是一个基于Java实现了 MVC 设计模式的请求驱动类型的轻量级Web 框架，通过把模型-视图-控制器分离，将web层进行职责解耦，把复杂的web 应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合。

# Spring MVC的优点

1、可以支持各种视图技术，而不仅仅局限于JSP。

2、与Spring框架集成(如 loC 容器、AOP 等) 。

3、清晰的角色分配：前端控制器(dispatcherServlet)，请求到处理器映射(HandlerMapping)，处理器适配器 (HandlerAdapter)，视图解析器(ViewResolver)

4、支持各种请求资源的映射策略。

# Spring MVC的主要组件

## 1、前端控制器DispatcherServlet（不需要程序员开发）

作用：接收请求、响应结果，相当于转发器，有了 DispatcherServlet 就减少了其它组件之间的耦合度。

## 2、处理器映射器HandlerMapping（不需要程序员开发）

作用：根据请求的 URL查找Handler。

## 3、处理器适配器 HandlerAdapter

在编写Handler 的时候要按照 HandlerAdapter 要求的规则去编写，这样适配器HandlerAdapter才可以正确的去执行 Handler。

## 4、处理器 Handler (需要程序员开发)

## 5、视图解析器 ViewResolver (不需要程序员开发)

作用：进行视图的解析，根据视图逻辑名解析成真正的视图 (view)

## 6、视图 View (需要程序员开发jsp)

View 是一个接口，它的实现类支持不同的视图类型 (jsp，freemarker，pdf 等)

# DispatcherServlet

Spring MVC框架是围绕 DispatcherServlet来设计的，它用来处理所有的 HTTP请求和响应。

# Spring MVC框架的控制器

控制器提供一个访问应用程序的行为，此行为通常通过服务接口实现。控制器解析用户输入并将其转换为一个由视图呈现给用户的模型。Spring 用一个非常抽象的方式实现了一个控制层，允许用户创建多种用途的控制器。

# Spring MVC的控制器是不是单例模式,如果是有什么问题,怎么解决?

## 基本说明

默认是单例模式，所以在多线程访问的时候有线程安全问题，不要用同步，会影响性能的，解决方案是在控制器里面不能写字段。

局部变量因为不会和其它线程共享变量，所以线程安全，这种思想能够很好的解决线程安全的问题，这种思想称之为线程封闭。

## 单例模式带来的问题

### 1、线程安全问题

当多个请求并发到达同一个控制器的单一实例时，如果控制器中有成员变量或状态信息，这些信息可能会被多个线程同时访问和修改，从而导致数据不一致和不可预期的行为。

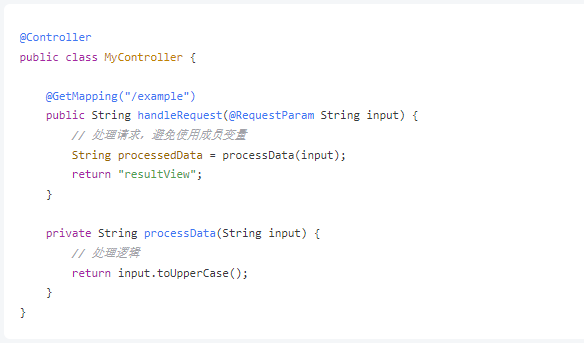
### 2、状态管理问题

如果控制器的某个方法依赖于某些可变状态（如成员变量），那么在不同请求之间，状态可能会相互干扰。这种情况会使得控制器不再是无状态的，违背了MVC设计的初衷。

## 解决方法

### 1、使用无状态的控制器

确保控制器中的所有成员变量都为final或者是完全不可变的。在处理请求时，尽量避免使用成员变量，而是使用方法局部变量。



### 2、使用@SessionScope或@RequestScope注解

可以将控制器的作用域设置为@SessionScope或@RequestScope，这样每个请求或每个会话都会有一个新的控制器实例。



需要注意的是，@SessionScope会在用户会话期间保留状态，而@RequestScope会在每次请求结束后销毁控制器实例。

### 3、依赖注入和服务层

将业务逻辑和状态管理移到服务层，而不是控制器中。控制器应仅负责处理请求，将复杂的业务逻辑委托给服务层，服务层可以使用单例或其他作用域的组件来管理状态。



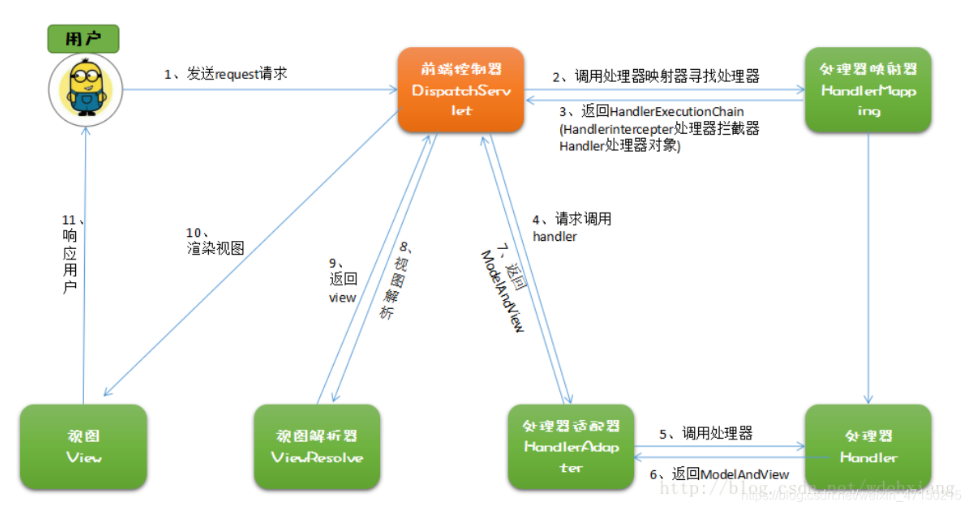
### 4、使用线程安全的数据结构

如果确实需要在控制器中使用状态，可以考虑使用线程安全的数据结构（如ConcurrentHashMap）来存储可变状态，以避免并发问题。

## 总结

Spring MVC控制器的单例模式在提高性能的同时，也带来了线程安全和状态管理的问题。通过使用无状态的控制器、合适的作用域、依赖注入和服务层等方法，可以有效地解决这些问题，确保控制器的线程安全和稳定性。保持控制器无状态是最佳实践，这样可以使其更加简单、易于测试和维护。

# 请描述Spring MVC的工作流程



1、用户发送请求到前端控制器DispatcherServlet。

2、DispatcherServlet 收到请求后，调用HandlerMapping处理器映射器，请求获取Handler。

3、处理器映射器根据请求的url找到具体的处理器，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。

4、DispatcherServlet 调用HandlerAdapter 处理器适配器。

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

6、Handler执行完成后返回 ModelAndView。

7、HandlerAdapter将Handler执行结果ModelAndView 返回给DispatcherServlet。

8、DispatcherServlet 将ModelAndView传给 ViewResolver 视图解析器进行解析。

9、ViewResolver解析后返回具体的View。

10、DispatcherServlet对View进行渲染视图 (即将模型数据填充至视图中)。

11、前端控制器DispatcherServlet向用户响应结果。

# MVC 是什么? MVC 设计模式的好处有哪些

## 基本概念

MVC是一种设计模式 (设计模式就是日常开发中编写代码的一种好的方法和经验的总结)。模型 (model) -视图 (view) -控制器 (controller) ，三层架构的设计模式。用于实现前端页面的展现与后端业务数据处理的分离。

## 好处

1、分层设计，实现了业务系统各个组件之间的解耦，有利于业务系统的可扩展

性，可维护性。

2、 有利于系统的并行开发，提升开发效率。

# SpingMVC中的控制器的注解一般用哪个,有没有别的注解可以替代?

一般用@Controller 注解,也可以使用@RestController,@RestController注解相当于@ResponseBody+@Controller，表示是表现层,除此之外，一般不用别的注解代替。

# Spring MVC怎么设定重定向和转发的

## 转发

在返回值前面加"forward:比如"forward:user.do?name=method4

## 重定向

在返回值前面加"redirect:"，譬如“redirect:www.baidu.com

# Spring MVC怎么和AJAX 相互调用的

通过Jackson框架就可以把Java里面的对象直接转化成Js可以识别的Json 对象，具体步骤如下：

1、加入 Jackson.jar文件

2、在配置文件中配置json的映射

3、在接受 Ajax 方法里面可以直接返回 Object,List 等,但方法前面要加上@ResponseBody注解。

# 如何解决POST请求中文乱码问题

解决post请求乱码问题

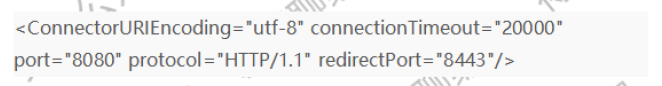
在 web.xml中配置一个 CharacterEncodingFilter 过滤器，设置成 utf-8;



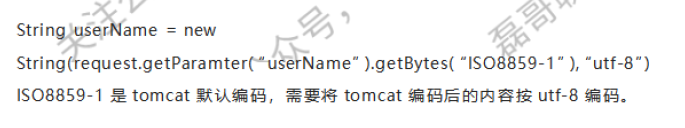
# 如何解决GET请求中文乱码问题

get请求中文参数出现乱码解决方法有两个:

1、修改tomcat 配置文件，添加编码与工程编码一致，如下



2、另外一种方法对参数进行重新编码



# Spring MVC的异常处理

可以将异常抛给 Spring 框架，由Spring框架来处理，我们只需要配置简单的异常处理器，在异常处理器中添加视图页面即可。

# 怎样在方法里面得到Request或者Session对象

直接在方法的形参中声明 request，Spring MVC 就自动把 request 对象传入。

# Spring MVC中函数的返回值是什么

返回值可以有很多类型,有 String,ModelAndView。ModelAndView 类把视图和数据都合并的一起的，但一般用 String 比较好。

# Spring MVC用什么对象从后台向前台传递数据的

通过ModelMap对象，可以在这个对象里面调用 put 方法,把对象加到里面,前台就可以通过 el 表达式拿到。

# Spring MVC里面拦截器是怎么写

有两种写法,一种是实现 Handlerlnterceptor接口，另外一种是继承适配器类，接着在接口方法当中实现处理逻辑，然后在 Spring MVC 的配置文件中配置拦截器即可。



# 怎么将File对象转换为MultipartFile接口

InputStream inputStream = new FileInputStream(file);

MultipartFile multipartFile = new MockMultipartFile("file",fileName, "image/png", inputStream);

# Controller中的请求方法，private 和 public有什么区别

https://mp.weixin.qq.com/s/J5M0eBPfrRqmlq3ap\_bfFA

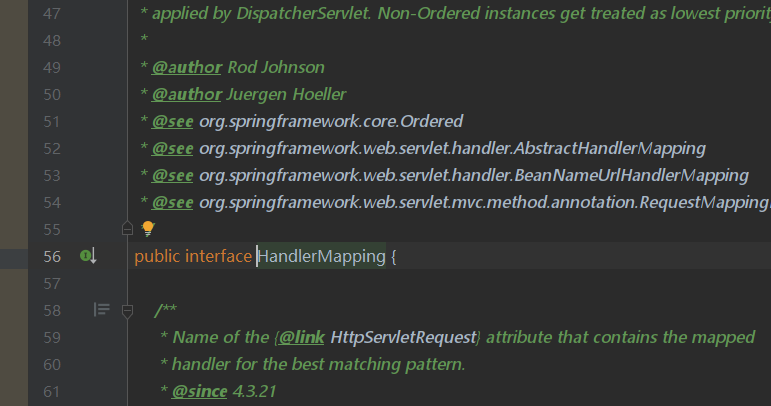
# 接口

## HandlerMapping

### 基本概念

HandlerMapping接口在Spring MVC中扮演着非常重要的角色，用于将HTTP请求映射到具体的处理器（通常是Controller）。它的主要职责是根据请求的信息（如URL、请求方法等）找到与之匹配的处理器，并返回一个处理链，以便后续处理。

### 源码



### 主要功能

#### 请求映射

**请求的解析和映射**：HandlerMapping负责根据请求的URL、HTTP方法等信息来查找合适的处理器。根据不同的请求特征，HandlerMapping可以精确地匹配到一个或多个处理器。

#### 返回处理链

**HandlerExecutionChain**：一旦找到匹配的处理器，HandlerMapping会返回一个HandlerExecutionChain对象。这个对象包含了处理器本身以及与之关联的拦截器（如果有的话）

### 工作流程

1、请求到达：用户发送HTTP请求，DispatcherServlet接收到这个请求。

2、查找处理器：

DispatcherServlet依次调用所有配置的HandlerMapping实例的getHandler()方法，将当前请求的信息传入。

每个HandlerMapping会尝试匹配请求，找到合适的处理器。

3、执行处理器：

一旦找到处理器，DispatcherServlet会调用其对应的handle方法进行请求处理，并执行相关的拦截器。

处理器的返回结果（通常是视图名或模型数据）将被用于生成最终的响应。

4、返回结果：处理完成后，DispatcherServlet会根据视图解析器将返回的视图名转化为具体的视图进行渲染。

### 实现类

#### RequestMappingHandlerMapping

这是最常用的实现，基于注解的请求映射（如@RequestMapping、@GetMapping、@PostMapping等）。它能够处理复杂的请求映射，包括路径变量、请求参数等。

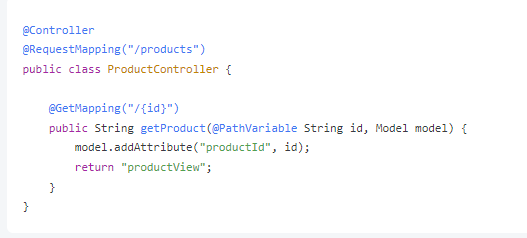
#### SimpleUrlHandlerMapping

这是一个简单的URL映射实现，主要用于不使用注解的情况。它可以通过配置文件来定义URL与处理器的映射关系。

#### BeanNameUrlHandlerMapping

这个实现可以根据Spring Bean的名称来映射URL，适用于一些特殊场景。

### 示例



请求流程示例：

1、用户发出请求 GET /products/123。

2、DispatcherServlet接收到请求，开始查找匹配的处理器。

3、RequestMappingHandlerMapping会解析这个请求，找到ProductController中的getProduct方法。

4、返回的HandlerExecutionChain包含找到的处理器和可能的拦截器。

5、DispatcherServlet调用getProduct方法，处理请求，并返回视图名productView。

6、最后，DispatcherServlet将视图名解析为具体的视图并返回给用户。

### 总结

HandlerMapping在Spring MVC中负责处理请求的路由工作，是实现请求与处理逻辑解耦的重要组件。通过灵活的映射策略，开发者能够轻松管理和扩展Web应用中的请求处理，提高了系统的可维护性和可扩展性。

## HandlerAdapter

### 基本概念

HandlerAdapter接口在Spring MVC中是一个重要的组成部分，它的主要作用是将请求处理逻辑（即控制器）与请求的具体处理方式（即如何调用控制器的方法）解耦。

### 作用

#### 1、****解耦控制器与请求处理机制****

HandlerAdapter使得Spring MVC能够支持多种类型的控制器。不同的控制器可能会有不同的请求处理方式，通过HandlerAdapter，Spring MVC能够灵活地选择适当的处理方式。

#### **2、**支持不同的控制器类型****

Spring MVC支持多种类型的控制器，如@Controller注解的类、SimpleController、HttpRequestHandler等。每种控制器的处理方式可能不同，HandlerAdapter可以根据具体的控制器类型来执行相应的处理逻辑。

#### **3、**提供统一的调用方式****

HandlerAdapter定义了一个统一的接口，用于处理请求和返回结果，使得整个请求处理过程更加简洁和一致。

### 工作原理

当一个HTTP请求到达Spring MVC时，处理流程大致如下：

**1、请求映射**：DispatcherServlet根据请求的URL找到对应的HandlerMapping，并获取到相应的控制器（Handler）。

**2、HandlerAdapter选择**：DispatcherServlet会通过HandlerAdapter来确定如何处理该控制器。它会查找适合该控制器类型的HandlerAdapter实例。

**3、请求处理**：找到适合的HandlerAdapter后，DispatcherServlet调用该适配器的handle方法，传递请求和响应对象，以及控制器的实例。

**4、返回结果处理**：HandlerAdapter处理完请求后，会返回一个ModelAndView对象，DispatcherServlet会根据返回的视图信息进行视图解析，最终将结果返回给客户端。

### 实现类

#### RequestMappingHandlerAdapter

用于处理使用@RequestMapping注解的控制器方法，支持请求参数、路径变量、请求体等多种类型的参数映射。

#### SimpleControllerHandlerAdapter

支持实现了Controller接口的控制器（如SimpleController），是一种较为简单的适配器。

#### HttpRequestHandlerAdapter

用于处理实现了HttpRequestHandler接口的控制器，这种控制器通常直接处理HTTP请求。

#### AnnotationMethodHandlerAdapter

（在Spring 3.x之前使用，3.x之后被RequestMappingHandlerAdapter取代）

支持处理使用注解的方法。

### 自定义HandlerAdapter

如果需要支持自定义的控制器类型，可以通过实现HandlerAdapter接口来创建自定义的适配器。实现步骤如下：

1、实现HandlerAdapter接口

（1）重写supports()方法来检查该适配器是否支持特定的控制器类型。

（2）实现handle()方法来执行控制器逻辑。



2、注册自定义的HandlerAdapter到Spring的配置中，以便在请求处理时使用。

### 总结

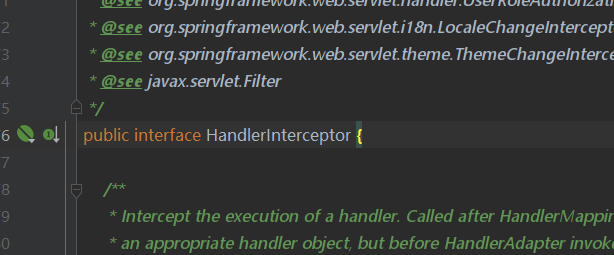
HandlerAdapter接口在Spring MVC中扮演着桥梁的角色，它实现了请求处理逻辑与控制器之间的解耦，使得框架可以支持多种控制器类型。通过使用适配器模式，Spring MVC可以灵活地处理不同类型的请求，使得开发者能够专注于业务逻辑的实现。

## HandlerInterceptor

### 基本概念

HandlerInterceptor 是 Spring MVC 中的一个接口，用于在处理请求的过程中定义一些额外的逻辑。它类似于 Servlet 的过滤器，但提供了更强大的功能，允许你在请求处理的多个阶段进行操作。

### 源码



### 方法

#### default boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

在控制器方法调用之前执行

HttpServletRequest request: 当前请求对象

HttpServletResponse response: 当前响应对象

Object handler: 处理器对象（通常是控制器方法）

返回值: 返回 boolean 类型。如果返回 true，请求将继续向下处理，调用下一个拦截器或目标方法；如果返回 false，请求将被终止，后续的处理将不会执行。



#### default void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,@Nullable ModelAndView modelAndView)

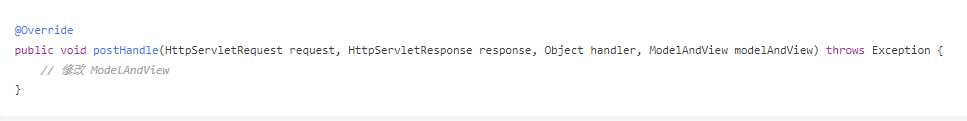
在控制器方法调用之后，但在视图渲染之前执行

HttpServletRequest request

HttpServletResponse response

Object handler

ModelAndView modelAndView: 控制器返回的 ModelAndView 对象，可以在这里进行修改。



#### default void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler,@Nullable Exception ex)

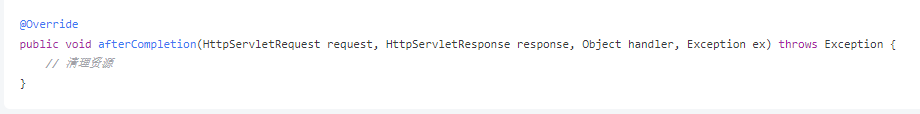
在整个请求完成之后执行，通常用于清理资源。

HttpServletRequest request

HttpServletResponse response

Object handler

Exception ex: 如果请求处理过程中抛出了异常，此参数将不为 null。



### 使用示例

#### 1、创建拦截器类



#### 2、注册拦截器

在 Spring 配置类中注册拦截器



### 应用场景

#### 1、身份认证与授权

（1）验证用户是否登录

（2）检查用户权限是否足够访问当前资源

实现方式：



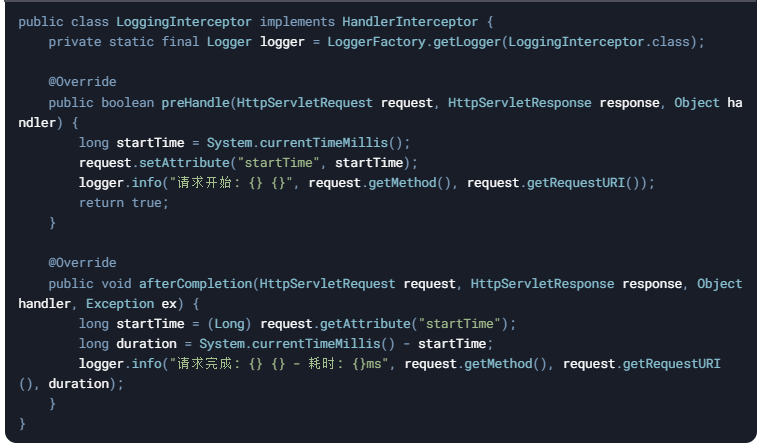
#### 2、日志记录

（1）记录请求和响应信息

（2）统计接口耗时

（3）审计用户操作

实现方式：



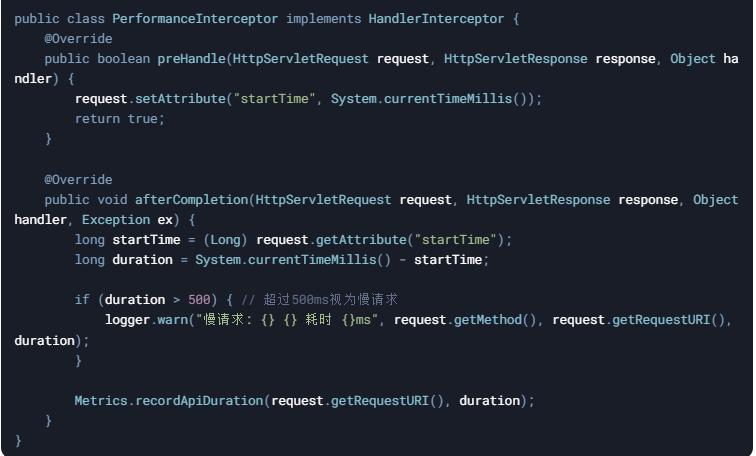
#### 3、性能监控

（1）监控接口响应时间

（2）统计慢请求

（3）系统性能分析

实现方式：

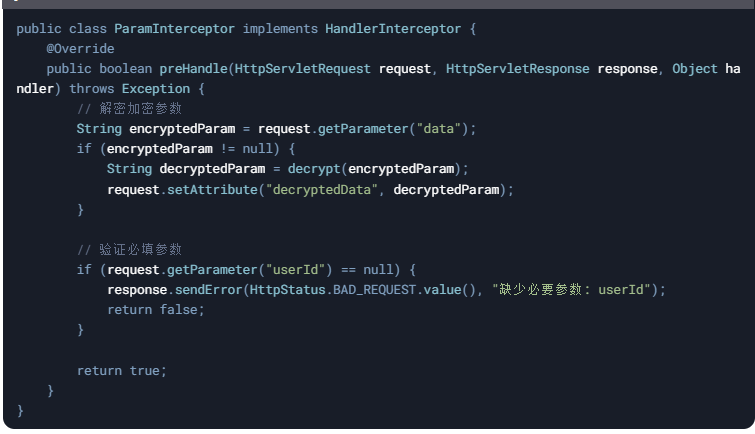


#### 4、请求参数预处理

（1）参数校验

（2）参数解密

（3）参数格式转换



#### 5、跨域处理

（1）统一处理CORS跨域请求

（2）动态设置跨域头

实现方式：



#### 6、防重复提交

（1）防止表单重复提交

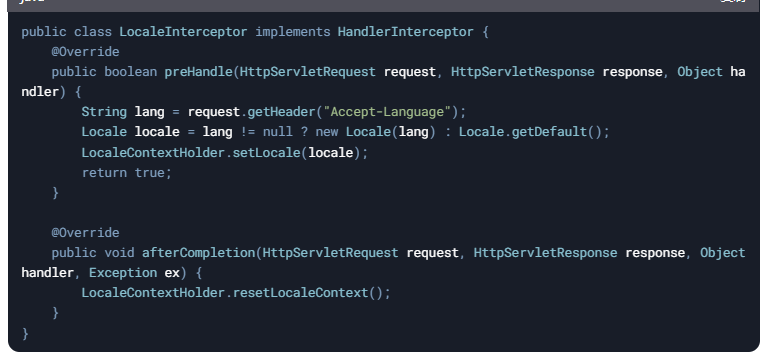
（2）实现幂等性控制



#### 7、国际化处理

（1）根据请求头或参数设置语言环境

（2）统一处理多语言资源



#### 8、缓存控制

（1）响应缓存处理

（2）防止缓存敏感数据

实现方式：



#### 拦截器配置

配置拦截器需要继承WebMvcConfigurer



#### 拦截器与过滤器的区别



### 总结

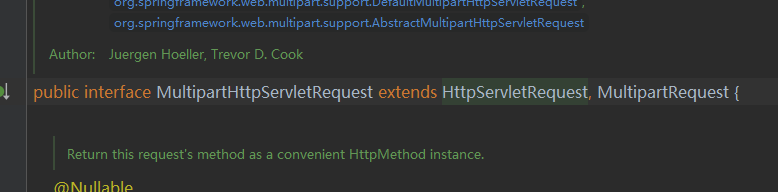
HandlerInterceptor 提供了一种便利的方式来在 Spring MVC 中处理请求的不同阶段。通过实现这个接口，你可以添加请求前的检查、结果后的处理，以及请求完成后的清理工作，非常适合用来实现日志记录、权限检查、性能监控等功能。

## MultipartHttpServletRequest

### 基本概念

MultipartHttpServletRequest 是 Spring Web 框架中的一个核心接口，用于处理 **HTTP 多部分请求**（即 multipart/form-data 类型的请求，通常用于文件上传）。它扩展了标准的 HttpServletRequest，提供了更便捷的方法来处理文件上传和表单数据。

### 源码



### 作用

**1、解析**multipart/form-data**请求**（如文件上传表单）。

**2、获取上传的文件**（MultipartFile）。

**3、获取普通表单字段**（文本、数字等）。

**4、提供额外的元数据**（如请求头、文件名等）。

### 方法

#### MultipartFile getFile(String name);

获取指定名称的单个文件

## MultipartFile

### 基本概念

主要用于处理 HTTP 请求中上传的文件。

1、前端发送文件

在前端页面中，用户可以选择一个或多个文件进行上传。一般情况下，上传文件的表单会使用 <input type="file"> 元素来实现。

2、后端接收文件

在后端的 Spring Web 应用中，你可以使用 MultipartFile 接口作为方法参数或字段类型来接收上传的文件。Spring 框架会自动解析 HTTP 请求，并将文件信息封装到 MultipartFile 对象中。

3、处理上传的文件

通过 MultipartFile 接口，你可以获取上传文件的一些重要信息，例如文件名、文件大小等。此外，MultipartFile 还提供了一些常用方法来操作上传的文件内容。你可以根据需要使用这些方法来读取、保存、转换或验证上传的文件。

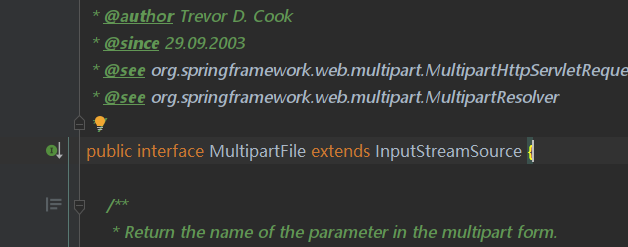
4、返回响应

在后端处理完上传的文件后，你可以根据需要返回适当的响应给前端。例如，你可以返回一个成功消息，表示文件上传成功，或者返回一个包含错误信息的响应，以指示文件上传过程中出现了问题。

### 说明

这个接口在spring-web-5.3.31.jar包中。

### 源码



### 作用

通过它你可以方便地接收和处理 HTTP 请求中上传的文件。它封装了文件的相关信息和常用操作方法，让你可以轻松地在后端应用中处理上传的文件。无论是保存文件、读取文件内容还是执行其他操作，MultipartFile 接口都是一个有用的工具。

### 方法

#### String getName()

返回参数的名称

#### String getOriginalFilename();

返回原始文件的名称

#### String getContentType();

返回内容类型

#### boolean isEmpty();

返回上传的文件是否为空，即在多部分表单中未选择任何文件或所选文件没有内容。

#### long getSize();

返回文件的大小，以字节为单位

#### InputStream getInputStream()

返回一个输入流以从中读取文件的内容

#### void transferTo(File dest)

将文件内容传输到指定的目标文件。

## FileItem

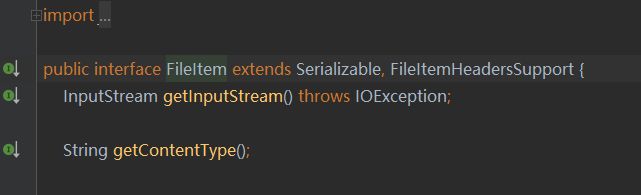
### 基本概念

FileItem接口是Apache Commons FileUpload库中的一个重要组成部分，专门用于处理文件上传。它提供了一种抽象方式来表示上传的文件或表单数据的单个部分。

### 说明

这个接口在commons-fileupload-1.3.2.jar包中

### 源码



### 作用

FileItem接口主要用于表示HTTP请求中表单提交的单个文件或字段。它的作用包括：

#### 1、抽象表示上传的数据

FileItem表示上传的文件或表单字段，提供了处理这些数据的标准方式。

#### 2、支持多种数据类型

它不仅支持文件上传，还可以处理普通表单字段，使得同一个接口可以处理多种类型的数据。

#### 3、简化文件上传处理

通过提供简单的方法，FileItem简化了文件上传时的复杂性，例如处理文件大小、文件类型、编码等问题。

### 方法

### 总结

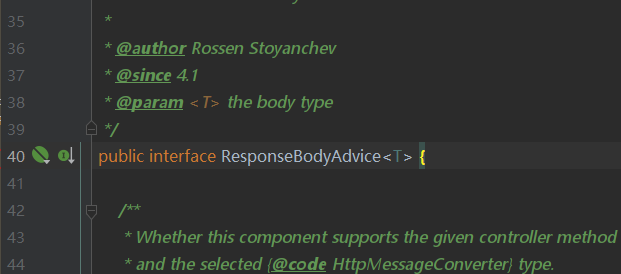
FileItem接口在文件上传处理中发挥着重要作用。它提供了一组方法，允许开发者轻松获取和处理上传的文件和表单字段。通过使用Apache Commons FileUpload库，可以方便地实现文件上传功能，简化开发过程。

## ResponseBodyAdvice

### 基本概念

ResponseBodyAdvice接口是Spring MVC提供的一个扩展点，允许开发者在响应体被写入客户端之前对其进行处理。它可以用于添加自定义的响应头、修改响应体内容、进行日志记录等操作。

### 源码



### 方法

#### boolean supports(MethodParameter returnType, Class<? extends HttpMessageConverter<?>> converterType);

该方法用于判断当前的ResponseBodyAdvice是否支持处理特定的返回类型和消息转换器。返回true表示支持，返回false表示不支持。这个方法可以根据方法返回类型、消息转换器类型等进行判断，以实现条件性拦截。

#### T beforeBodyWrite(@Nullable T body, MethodParameter returnType, MediaType selectedContentType,Class<? extends HttpMessageConverter<?>> selectedConverterType,ServerHttpRequest request, ServerHttpResponse response);

该方法在响应体写入之前被调用，可以对返回的对象进行修改。

body：原始响应体

returnType：方法返回类型

selectedContentType：选择的内容类型（如JSON、XML等）

selectedConverterType：选择的消息转换器

request：当前请求对象

response：当前响应对象

返回值是修改后的响应体，返回null则不会写入响应。

### 作用

ResponseBodyAdvice的主要作用是拦截并修改返回给客户端的响应体。它在处理器方法返回结果后、实际写入响应之前被调用，可以在这个阶段实现一些通用的功能，如：

#### 修改响应内容

对返回的数据进行格式化或包装，比如添加额外的属性、修改数据结构等。

#### 添加响应头

可以在响应中添加自定义的HTTP头部。

#### 处理通用的响应格式

例如，将所有响应包装成一个标准格式，以便前端统一处理。

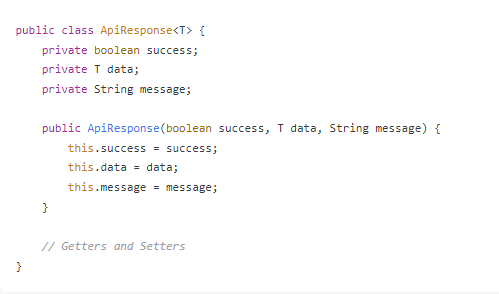
#### 日志记录

记录响应信息，例如响应时间、状态等。

### 示例

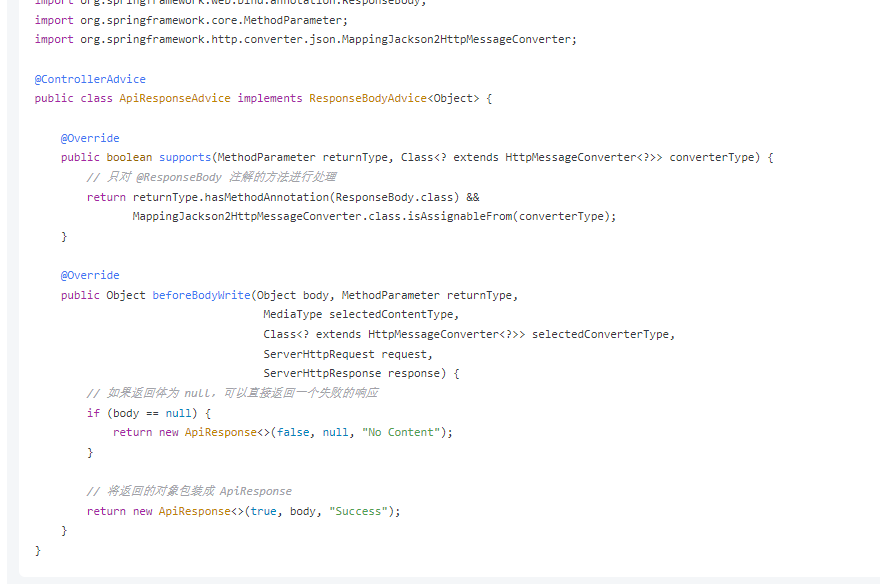
#### 1、创建一个响应包装类

首先，定义一个简单的响应包装类



#### 2、实现 ResponseBodyAdvice 接口

接下来，实现ResponseBodyAdvice接口，使用这个包装类来包装所有的响应体。



#### 3、使用 ApiResponseAdvice

一旦你实现了ResponseBodyAdvice，Spring MVC会自动应用它。你只需在Controller中返回普通对象即可



### 总结

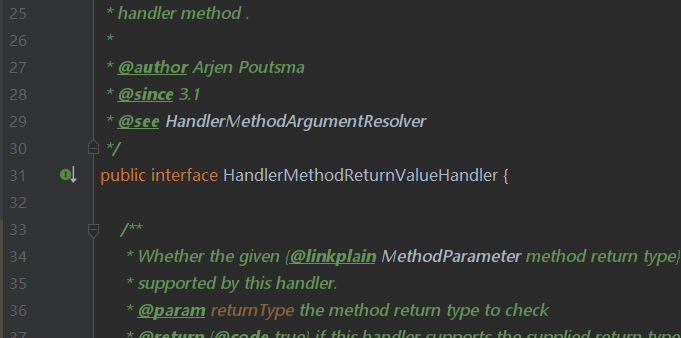
ResponseBodyAdvice接口是一个强大的工具，允许开发者在响应体写入前进行全局处理。通过实现该接口，可以方便地添加统一的响应格式、修改响应内容、添加日志等，极大地增强了Spring MVC应用的灵活性和可维护性。使用时只需实现接口方法并注册到Spring上下文中，Spring会自动将其应用于所有符合条件的请求。

## HandlerMethodReturnValueHandler

### 基本概念

HandlerMethodReturnValueHandler 接口在 Spring Web 中负责处理控制器方法的返回值。它的主要作用是将控制器方法返回的对象转换为适合 HTTP 响应的格式，确保返回值正确地传递给客户端。

### 源码



### 方法

#### boolean supportsReturnType(MethodParameter returnType);

检查当前的处理器是否支持给定的返回类型。如果支持，返回 true

#### void handleReturnValue(@Nullable Object returnValue, MethodParameter returnType,ModelAndViewContainer mavContainer, NativeWebRequest webRequest)

处理返回值。实现逻辑包括将返回值转换为适合的格式并设置到 HTTP 响应中。

### 主要功能

#### 1、处理返回值

根据控制器方法的返回类型，选择合适的处理器来处理返回值。

#### 2、多种返回类型支持

可以处理多种类型的返回值，例如对象、ResponseEntity、ModelAndView 等。

#### 3、灵活性

开发者可以自定义实现该接口，以支持特定的返回值处理逻辑。

### 示例

假设我们需要处理一个特殊的返回类型，例如自定义的 MyResponse 对象。我们可以实现 HandlerMethodReturnValueHandler



注册自定义处理器

要使用自定义的返回值处理器，需要将其注册到 Spring 的配置中



### 总结

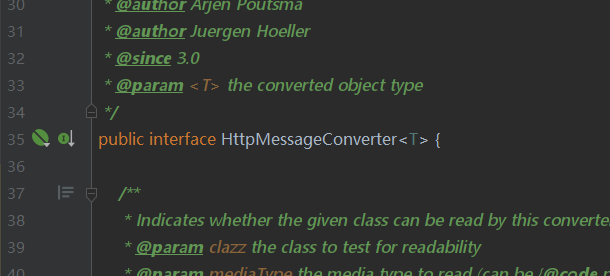
HandlerMethodReturnValueHandler 接口在 Spring Web 中扮演了关键角色，负责将控制器方法的返回值处理成适合 HTTP 响应的格式。通过实现该接口，开发者可以灵活地扩展 Spring 的返回值处理能力，以满足特定需求。

## HttpMessageConverter

### 基本概念

HttpMessageConverter 接口在 Spring Web 中的主要作用是将 HTTP 请求和响应体与 Java 对象之间进行转换。它支持多种数据格式，如 JSON、XML 和表单数据，方便开发者处理不同类型的数据。

### 源码



### 主要功能

#### 1、****序列化和反序列化****

将 Java 对象转换为 HTTP 响应中的数据格式（如 JSON），以及将请求体中的数据转换为 Java 对象。

#### ****2、内容类型处理****

根据请求或响应的 Content-Type 和 Accept 头部，选择合适的转换器来处理数据。

#### **3、支持多种格式**

内置多种转换器，如 MappingJackson2HttpMessageConverter（用于 JSON）、XmlAwareHttpMessageConverter（用于 XML）等

#### **4、灵活性**

开发者可以自定义自己的 HttpMessageConverter 实现，以支持特定的数据格式或业务需求。

### 工作流程

#### 1、请求处理

当收到 HTTP 请求时，Spring 根据请求的 Content-Type 选择合适的 HttpMessageConverter，并使用它将请求体转换为对应的 Java 对象。

#### 2、响应处理

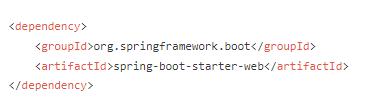
在控制器方法返回 Java 对象时，Spring 使用合适的 HttpMessageConverter 将对象序列化为响应体格式，发送给客户端。

### 示例

下面是一个使用 Spring Web 的 HttpMessageConverter 的代码示例，展示了如何将 Java 对象转换为 JSON 格式，并处理 HTTP 请求和响应。

#### 1、添加maven依赖

首先，确保在 pom.xml 中添加以下依赖：



#### 2、创建Java对象

创建一个简单的 Java 类，例如 User：



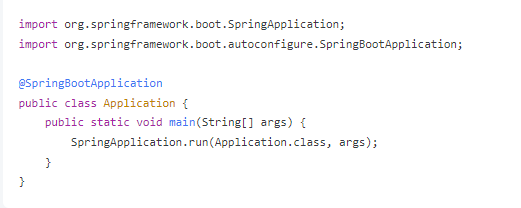
#### 3、创建Controller

创建一个控制器，处理 HTTP 请求：



#### 4、运行Spring Boot 应用

确保有一个主类来运行 Spring Boot 应用：

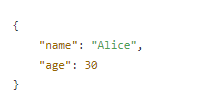


#### 5、测试API

POST 请求（创建用户）：

URL: http://localhost:8080/users

Body (JSON)



GET 请求（获取用户）：

URL: http://localhost:8080/users/Alice

返回的 JSON：



#### 6、结果

当发送 POST 请求时，HttpMessageConverter 会将请求体的 JSON 转换为 User 对象；当发送 GET 请求时，返回的 User 对象会被自动转换为 JSON 格式。这就是 HttpMessageConverter 在 Spring Web 中的工作原理。

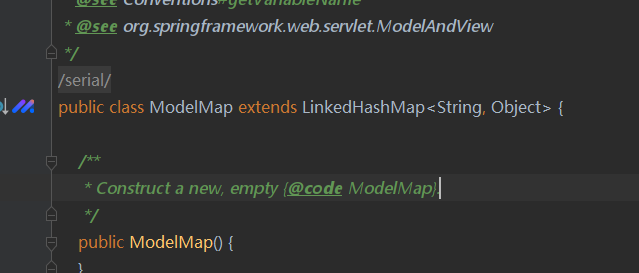
# 类

## ModelMap

### 基本概念

ModelMap 是 Spring 框架中的一个类，主要用于在 MVC 模式中处理模型数据。它是一个用于传递数据到视图（例如 JSP、Thymeleaf 等）中的容器。ModelMap 继承自 LinkedHashMap类，提供了方便的方法来存储和管理模型属性。

### 源码



### 主要作用

#### 1、存储模型数据

ModelMap 允许开发者将多个属性存储到一个容器中，这些属性会被传递到视图层进行渲染。

#### 2、简化代码

使用 ModelMap 可以使得控制器中的代码更简洁，开发者无需创建复杂的模型对象，可以直接将属性以键值对的形式存入 ModelMap 中。

#### 3、与视图解耦

ModelMap 把控制器和视图的耦合降到最低，控制器只负责填充数据，而视图负责展示数据，促进了代码的分离。

### 使用方法

#### 1、****作为控制器方法的参数****

在控制器方法中，可以直接使用 ModelMap 作为参数，Spring 会自动注入该对象。



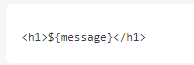
#### ****2、添加属性****

可以通过 addAttribute 方法将数据添加到 ModelMap 中。



#### 3、****从**** ModelMap ****中获取属性****

在视图中，可以通过属性名获取模型数据



#### 4、与 ModelAndView 一起使用

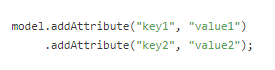
ModelMap 也可以与 ModelAndView 一起使用，以便同时传递模型数据和视图名称。



### 其他特性

#### 1、链式调用

ModelMap 支持链式调用，可以连续调用 addAttribute 方法，简化代码。



#### 2、合并多个模型

ModelMap 可以与其他 Map 类型的模型合并。

### 总结

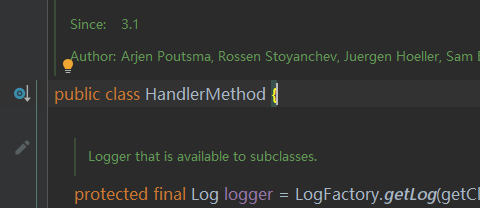
ModelMap 是 Spring MVC 中一个非常实用的类，它简化了数据的传递过程，促进了代码的清晰性与可维护性。在实际开发中，合理使用 ModelMap 能够提高开发效率，使得视图和控制器之间的数据交互更加流畅。

## HandlerMethod

### 基本概念

HandlerMethod是Spring Framework中的一个类，主要用于处理 Spring MVC 的请求。它封装了对请求处理方法的描述，提供了在 Spring MVC 的处理请求流程中所需的所有信息。这使得 Spring 能够灵活地调用和管理请求处理方法。HandlerMethod 是 Spring MVC 中的一个核心类，通常由 RequestMappingHandlerMapping 组件生成，并由DispatcherServlet 在请求处理过程中使用。它代表了一个请求处理方法及其相关的元数据（如类和方法参数等）。

### 源码



### 主要作用

#### 1、封装请求处理方法

HandlerMethod 包含了处理 HTTP 请求的方法（通常是控制器中的方法）以及相关的类信息。

#### 2、提供元数据

它提供了与处理方法相关的各种元数据，例如：

1、方法的参数类型

2、方法的返回类型

3、请求方法（GET POST等）

4、请求路径

5、处理方法所在的控制器类等。

#### 3、方法调用的抽象

HandlerMethod 允许在运行时通过反射调用控制器方法，从而处理请求。

### 重要的属性

#### Method method

表示被调用的处理方法

#### Object bean

表示处理方法所在的控制器对象

### 关键方法

#### getBean()

返回处理方法所在的控制器实例

#### getMethod()

返回实际的处理方法

#### getMethodParameters()

返回处理方法的参数信息

#### invoke(Object... args)

调用该处理方法，传入相应的参数。

#### getReturnType()

获取处理方法的返回类型。

### 与其他类的关系

HandlerMethod 通常与以下类一起工作：

RequestMappingHandlerMapping：负责将请求映射到相应的处理方法。它根据请求的信息（如路径、HTTP 方法等）创建 HandlerMethod 实例。

DispatcherServlet：核心的请求分发器，负责调用合适的 HandlerMethod 进行请求处理。

HandlerMethodArgumentResolver：用于解析方法参数。Spring MVC 会根据请求中的参数类型和注解，使用不同的解析器将请求参数映射到处理方法的参数上。

### 示例

在实际使用中，HandlerMethod 通常不会直接被开发者使用，但理解其工作机制对于深入了解 Spring MVC 的工作原理是非常重要的。下面是一个简化的示例，展示了 HandlerMethod 的基本使用：



在这个例子中，Spring 会为 greet 方法生成一个 HandlerMethod 实例，以封装该方法及其相关信息。

### 总结

HandlerMethod 是 Spring MVC 中一个非常重要的类，提供了处理请求方法的封装和元数据支持。它通过与其他 Spring 组件的协作，支持灵活的请求处理机制，使得开发者能够专注于业务逻辑的实现，而不必关心底层的请求处理细节。理解 HandlerMethod 的工作原理有助于深入掌握 Spring MVC 的请求处理流程和设计理念。

## HandlerExecutionChain

### 基本概念

在 Spring MVC 中，当一个 HTTP 请求到达时，框架会根据 URL 映射找到对应的处理器（通常是控制器）。HandlerExecutionChain 类则负责将请求处理器和相关的拦截器组合在一起，以便在执行请求时提供更灵活的处理方式。

### 主要组成

Handler：处理器本身，通常是一个控制器实例（例如，使用 @Controller 注解的类），负责处理请求并返回响应。

Interceptors：一个拦截器数组，拦截器是在请求处理之前和之后执行的一些处理逻辑，可以用于权限检查、日志记录、性能监控等。

### 作用

#### 1、管理处理器和拦截器

将请求处理器和拦截器组织在一起，形成一个完整的处理链。

#### 2、执行拦截器

在请求处理前后，依次调用拦截器中的方法，从而实现一些横切关注点的功能，例如：

preHandle()：在请求到达处理器之前执行。

postHandle()：在处理器执行之后，但在视图渲染之前执行。

afterCompletion()：在请求处理完成后执行。

#### 3、提供链式执行机制

通过 HandlerExecutionChain，可以方便地实现多个拦截器的链式调用，使得请求处理更加灵活。

### 使用场景

### 示例

### 总结

## MethodParameter

### 基本概念

## RequestMappingHandlerMapping

## CommonsMultipartFile

### 基本概念

CommonsMultipartFile 是一个旧版本的 Spring 类，从 Spring 5.0 开始已经不建议使用了。正确的方式是使用 MockMultipartFile 类。

## MockMultipartFile

### 基本概念

可以使用 MockMultipartFile 类来模拟文件上传。

### 方法

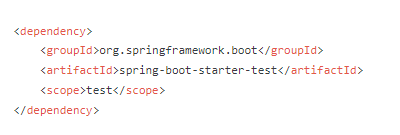
#### public MockMultipartFile(String name, InputStream contentStream)

通过给定的内容创建MockMultipartFile对象

### 操作步骤

1、导入所需的依赖项

你可以添加以下测试依赖：



2、创建 MockMultipartFile 对象

在测试类中，你需要创建一个 MockMultipartFile 对象，来模拟上传的文件。你可以使用 MockMultipartFile 的构造函数来指定文件名、内容以及其他文件相关信息。示例代码如下：



3、执行文件上传的测试

在测试方法中，你可以执行针对文件上传的测试逻辑。例如，你可以调用相应的控制器方法，传入 MockMultipartFile 对象，然后验证上传文件的处理结果是否符合预期。

4、断言和验证

在测试方法中，你可以使用断言来验证上传文件的处理结果是否正确。例如，你可以断言某个返回值、某个状态码或者验证数据库中的存储情况等。

### 总结

MockMultipartFile 类允许你在 Spring Web 的单元测试中模拟文件上传。通过创建 MockMultipartFile 对象，你可以指定文件的各种属性，并将其用作单元测试中的输入。这样，你可以测试文件上传相关的逻辑，而无需实际上传真实的文件。

## RequestMappingHandlerAdapter

## MappingJackson2HttpMessageConverter

## HttpServletRequestWrapper

### 基本概念

## DefaultHandlerExceptionResolver