Java EE技术

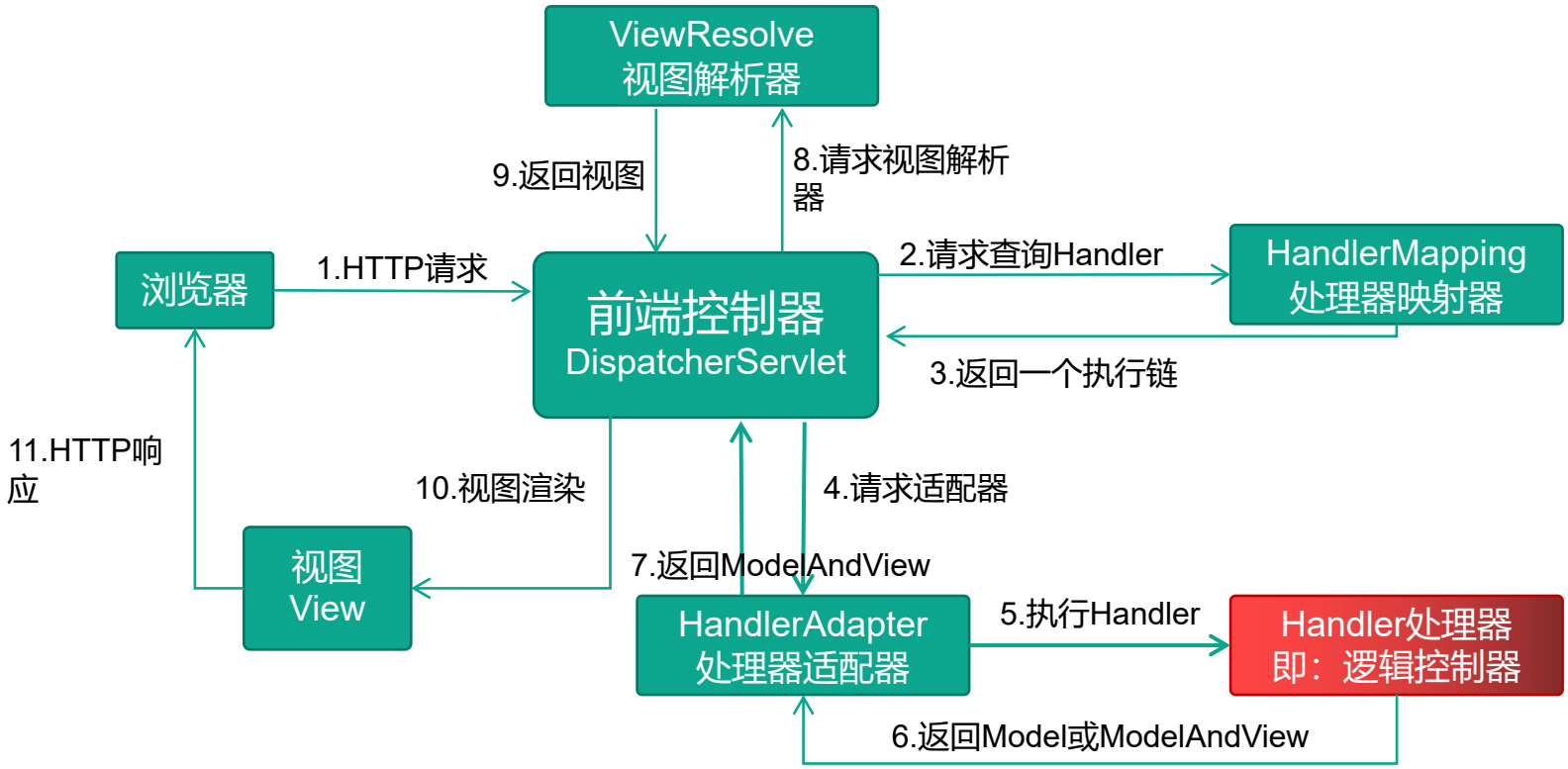
Spring MVC

内容

概述

Spring MVC 是 Spring 提供的一个基于 MVC 设计模式的轻量级 Web 开发框架。在 Spring MVC 框架中，Controller（控制器） 替换 Servlet 来担负控制器的职责，用于接收请求，调用相应的 Model 进行处理，处理器完成业务处理后返回处理结果。Controller 调用相应的 View 并对处理结果进行视图渲染，最终客户端得到响应信息。

Spring MVC工作原理



Spring MVC工作原理

➢DispatcherServlet

DispatcherServlet 是前端控制器，从图 1 可以看出，Spring MVC 的所有请求都要经过 DispatcherServlet 来统一分发。DispatcherServlet 相当于一个转发器或中央处理器，控制整个流程的执行，对各个组件进行统一调度，以降低组件之间的耦合性，有利于组件之间的拓展。

➢HandlerMapping

HandlerMapping 是处理器映射器，其作用是根据请求的 URL 路径，通过注解或者 XML 配置，寻找匹配的处理器（Handler）信息。

Spring MVC工作原理

➢HandlerAdapter

HandlerAdapter 是处理器适配器，其作用是根据映射器找到的处理器（Handler）信息，按照特定规则执行相关的处理器（Handler）。

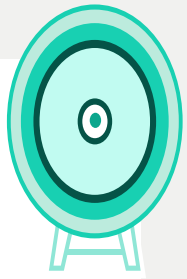
➢ Handler

逻辑控制器，其作用是执行相关的请求处理逻辑，并返回相应的数据和视图信息，将其封装至 ModelAndView 对象中。

➢ View Resolver

View Resolver 是视图解析器，其作用是进行解析操作，通过 ModelAndView 对象中的 View 信息将逻辑视图名解析成真正的视图 View（如通过一个 JSP 路径返回一个真正的 JSP 页面）。

Spring MVC工作原理



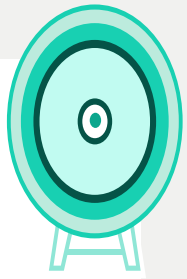
- **View**

  视图，返回给客户端的数据或页面。Spring MVC支持多种视图，默认为JSP。

Spring MVC Maven依赖

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>x.x.x</version>
</dependency>
```

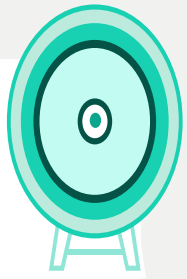
Spring MVC控制器配置



启用Spring MVC，在web.xml中添加：

```
<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-config.xml</param-value><!--配置文
件-->
    </init-param>
  </servlet>
```

Spring MVC控制器配置



在web.xml中添加：

```
<servlet-mapping>
    <servlet-name>springMVC</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

Spring MVC框架配置

首先，在Spring配置文件的beans中添加命名空间：

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

在xsi:schemaLocation添加：

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

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

Spring MVC框架配置

➢注解驱动配置

添加注解驱动的目的是加载处理注解的映射器和适配器。代码如下：

```
<mvc:annotation-driven />
```

➢组件扫描

功能和spring一样，通过扫描自动注册通过注解注册的bean。

```
<context:component-scan base-package="包名" />
```

Spring MVC框架配置

➢视图解析器配置

用于指定视图解析器的实现类及视图所在位置。

```
<bean
class="org.springframework.web.servlet.view.InternalResourc
eViewResolver">
    <!--后缀，一般指视图文件的扩展名-->
    <property name="suffix" value=".jsp" />
    <!--前缀，一般指视图的存放位置-->
    <property name="prefix" value="/WEB-INF/views/" />
</bean>
```

Spring MVC框架配置

➢静态资源配置

默认Spring MVC的控制器会对所有的请求进行处理；但静态资源，如：CSS文件、js文件、图片等不需要服务器处理的资源，可通知控制器对这些请求直接放行。配置方法：

```
<mvc:resources mapping="URL/**" location="本地路径" />
```

其中：location的起点为工程的“webapp”目录。

控制器

- Spring MVC基于方法来处理客户端的请求，在同一个控制器中定义多个方法（Handler）来处理多种请求。在Spring MVC中只需要在类上通过@Controller进行注解，即可生成一个控制器，而在该控制器的方法上，通过@RequestMapping来注解一个处理器（Handler）。

控制器

➢@Controller

注解类为一个控制器。在类名上进行注解，并放到Spring的扫描路径上。

➢@RequestMapping

注解控制器的方法为一个处理器（Handler），并指明一个请求路径（url）来处理客户端的一个请求。格式：

@RequestMapping（“url”）

@RequestMapping（ {“url1”，“url2”...} ）

控制器--示例

```
@Controller
public Demo{
        @RequestMapping(“/index”)
        public String index(){
                .....
                return “index”;                //返回一个逻辑视图
        }
}
```

控制器

➢@RequestMapping

另外，@RequestMapping也可以注解在类上；当注解在类上时，注解的“url”相当于一个目录，而该类中被注解的方法相当于文件。如：

```
@Controller
@RequestMapping(“/user”)
public UserController{
    @RequestMapping(“/add”)
    public String useradd(User user){....}
}
```

对useradd方法的访问路径为：

/user/add

控制器

➢获取客户端参数

Spring MVC通过处理方法的方法参数既可以获取客户端参数。

1、单独参数获取

要求客户端参数名和方法参数名一致即可；另外，控制器在获取参数前会对客户端参数按方法参数的类型进行转换，因此注意客户端参数的类型和格式。例：

```
public String index(String uname, Integer age){}
```

控制器

➢获取客户端参数

Spring MVC通过处理方法的方法参数既可以获取客户端参数。

2、复合参数获取

要求方法参数为JavaBean，客户端参数名与Bean属性名一直即可。若没有对应的客户端参数，Bean的属性值将为默认值。

public String index(User user){}

控制器

➢ 向视图传值及选择视图

■ModelAndVeiw

ModelAndVeiw用于向视图传值并指明视图路径。使用ModelAndVeiw向视图传值，要求处理方法的返回值必须为ModelAndVeiw。常用方法：

addObject（String attributeName，Object value）：向视图传值；

setViewName（String path）：指明视图路径

注意：ModelAndVeiw必须在处理方法中创建、赋值并返回。

控制器

➢ 向视图传值及选择视图

■Model

Model只用于向视图传值。使用Model向视图传值，要求处理方法的返回值为String，即：视图路径。使用Model时，Model对象为处理方法的参数传入，即Model对象由Spring容器创建。常用方法：

addAttribute(String name,Object value)，

addAttributes(Map<String,?> map)：　向视图传值；

控制器

➢页面重定向

场景：若处理方法在处理过程中出现错误，或不允许浏览相应视图，则需跳转到错误页面；处理方法：进行页面跳转。

只需设置视图路径为：

redirect:新URL

控制器

➢向处理方法注入Servlet对象

Spring MVC对Servlet做了封装，在一般操作中不需要使用Servlet的对象，但在一下特殊情况下需要直接操作Servlet对象（Request、Session）才能完成。需要使用这些对象时，只要将需要的对象作为处理方法的参数传入即可。如：

```
public String index(String uname, HttpSession session){}
```

其中session对象即servlet的会话对象。

文件上传

文件上传是Web应用常用的一个功能。Spring MVC对文件上传做了封装，使文件上传变得简单。Spring MVC的文件上传功能依赖于Apache的Commons FileUpload组件，因此在pom中必须添加依赖：

```
<dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
        <version>x.x.x</version>
</dependency>
```

文件上传

另外需要Spring MVC的配置文件中添加“MultipartResolver”解析器。代码：

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize"
value="5120000000" />
    <property name="defaultEncoding" value="utf-8" />
</bean>
```

文件上传

- **其中：**

  maxUploadSize：指定上传附件的大小上限；

  defaultEncoding：请求编码格式，应与应用的编码一致；

  注意：id必须为：multipartResolver

文件上传

- **上传文件表单：**

```
<form action="/upload" method="post"
enctype="multipart/form-data">
    <input name="file" type="file"/>
    <button type="submit">Upload</button>
</form>
```

注：method必须为post，enctype必须为"multipart/form-data"

文件上传

- **服务器端处理方法：**

  public String upload(MultipartFile file,....){}

  MultipartFile对象常用方法：

  String getContentType()：获取文件MIME类型，即文件类型；如：“image/jpeg”

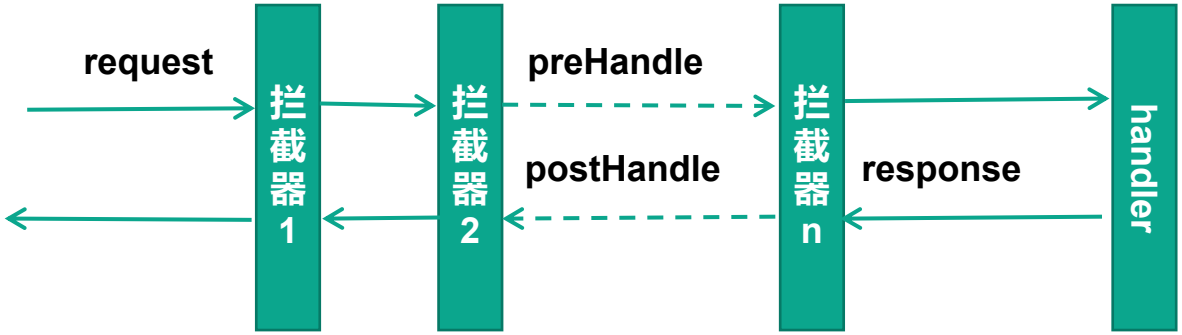
  InputStream getInputStream()：获取上传文件流；

  String getOriginalFilename()：获取上传文件原名；

  void transferTo(File dest)：将上传文件保存的指定文件；

拦截器

Spring MVC中存在大量的拦截器，主要完成对用户请求进行预处理和拦截及后期处理。如：类型转换、请求参数注入等。用户请求在到达处理Handler之前需经过一系列的拦截器，拦截器工作原理：



拦截器实现

实现一个拦截器只需实现HandlerInterceptor接口即可；其接口方法有：

boolean preHandle(HttpServletRequest req, HttpServletResponse res, Object handler)：在请求处理之前被调用；返回true时继续后续连接器及处理方法的调用，若返回false则表示请求结束，后续的拦截器及处理方法则不再执行。注意：若在该方法中拦截了请求（返回false），则必须进行请求转发或重定向。

拦截器实现

void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)

只有处理方法被执行后，该方法才会被执行；其功能用于视图处理。

其中：

ModelAndView： 为处理方法返回的视图及数据。

handler：被调用的处理方法。

拦截器实现

void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)

只有在相应的拦截器的prehandler返回true时才会被执行。

拦截器注册

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="拦截路径"/>
        <bean class="实现类" />
    </mvc:interceptor>
</mvc:interceptors>
```

注：拦截路径一般需要使用通配符“*”；如：/* 表示对所有非静态资源进行拦截。

参考资料

- http://c.biancheng.net/spring_mvc/mvc.html