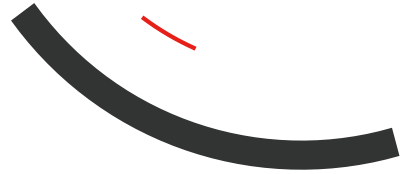


黑马程序员™
www.itheima.com

传智播客旗下
高端IT教育品牌



SpringMVC入门

目录 Contents

- ◆ Spring与Web环境集成
- ◆ SpringMVC的简介
- ◆ SpringMVC的组件解析

1. Spring集成web环境

1.1 ApplicationContext应用上下文获取方式

应用上下文对象是通过`new ClasspathXmlApplicationContext(spring配置文件)`方式获取的，但是每次从容器中获得Bean时都要编写`new ClasspathXmlApplicationContext(spring配置文件)`，这样的弊端是配置文件加载多次，应用上下文对象创建多次。

在Web项目中，可以使用`ServletContextListener`监听Web应用的启动，我们可以在Web应用启动时，就加载Spring的配置文件，创建应用上下文对象`ApplicationContext`，在将其存储到最大的域`ServletContext`域中，这样就可以在任意位置从域中获得应用上下文`ApplicationContext`对象了。

1. Spring集成web环境

1.2 Spring提供获取应用上下文的工具

上面的分析不用手动实现，Spring提供了一个监听器**ContextLoaderListener**就是对上述功能的封装，该监听器内部加载Spring配置文件，创建应用上下文对象，并存储到**ServletContext**域中，提供了一个客户端工具**WebApplicationContextUtils**供使用者获得应用上下文对象。

所以我们需要做的只有两件事：

- ① 在**web.xml**中配置**ContextLoaderListener**监听器（导入spring-web坐标）
- ② 使用**WebApplicationContextUtils**获得应用上下文对象**ApplicationContext**

1. Spring集成web环境

1.3 导入Spring集成web的坐标

```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-web</artifactId>  
  <version>5.0.5.RELEASE</version>  
</dependency>
```

1. Spring集成web环境

1.4 配置ContextLoaderListener监听器

```
<!--全局参数-->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:applicationContext.xml</param-value>
</context-param>
<!--Spring的监听器-->
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

1. Spring集成web环境

1.5 通过工具获得应用上下文对象

```
ApplicationContext applicationContext =  
    WebApplicationContextUtils.getWebApplicationContext(servletContext);  
Object obj = applicationContext.getBean("id");
```

1. Spring集成web环境

1.5 知识要点

Spring集成web环境步骤

- ① 配置ContextLoaderListener监听器
- ② 使用WebApplicationContextUtils获得应用上下文

目 录 Contents

- ◆ Spring与Web环境集成
- ◆ SpringMVC的简介
- ◆ SpringMVC的组件解析

2. SpringMVC 简介

2.1 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。

2. SpringMVC 简介

2.3 SpringMVC快速入门

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DispatchcerServlet
- 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

2. SpringMVC 简介

2.2 SpringMVC快速入门

① 导入Spring和SpringMVC的坐标

```
<!--Spring坐标-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.0.5.RELEASE</version>
</dependency>
<!--SpringMVC坐标-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>5.0.5.RELEASE</version>
</dependency>
```

2. SpringMVC 简介

2.2 SpringMVC快速入门

① 导入Servlet和Jsp的坐标

```
<!--Servlet坐标-->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
</dependency>
<!--Jsp坐标-->
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.0</version>
</dependency>
```

2. SpringMVC 简介

2.2 SpringMVC快速入门

② 在web.xml配置SpringMVC的核心控制器

```
<servlet>
  <servlet-name>DispatcherServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring-mvc.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>DispatcherServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

2. SpringMVC 简介

2.2 SpringMVC快速入门

③ 创建Controller和业务方法

```
public class QuickController {  
    public String quickMethod() {  
        System.out.println("quickMethod running....");  
        return "index";  
    }  
}
```

2. SpringMVC 简介

2.2 SpringMVC快速入门

③ 创建视图页面index.jsp

```
<html>
<body>
    <h2>Hello SpringMVC!</h2>
</body>
</html>
```


2. SpringMVC 简介

2.2 SpringMVC快速入门

④ 配置注解

```
@Controller
public class QuickController {
    @RequestMapping("/quick")
    public String quickMethod(){
        System.out.println("quickMethod running....");
        return "index";
    }
}
```

2. SpringMVC 简介

2.2 SpringMVC快速入门

⑤ 创建spring-mvc.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">
    <!--配置注解扫描-->
    <context:component-scan base-package="com.itheima"/>
</beans>
```

2. SpringMVC 简介

2.2 SpringMVC快速入门

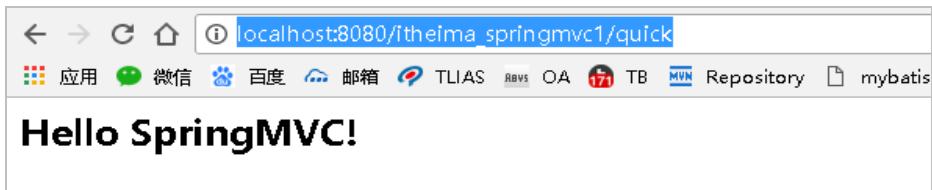
⑥ 访问测试地址

```
http://localhost:8080/itheima_springmvc1/quick
```

控制台打印

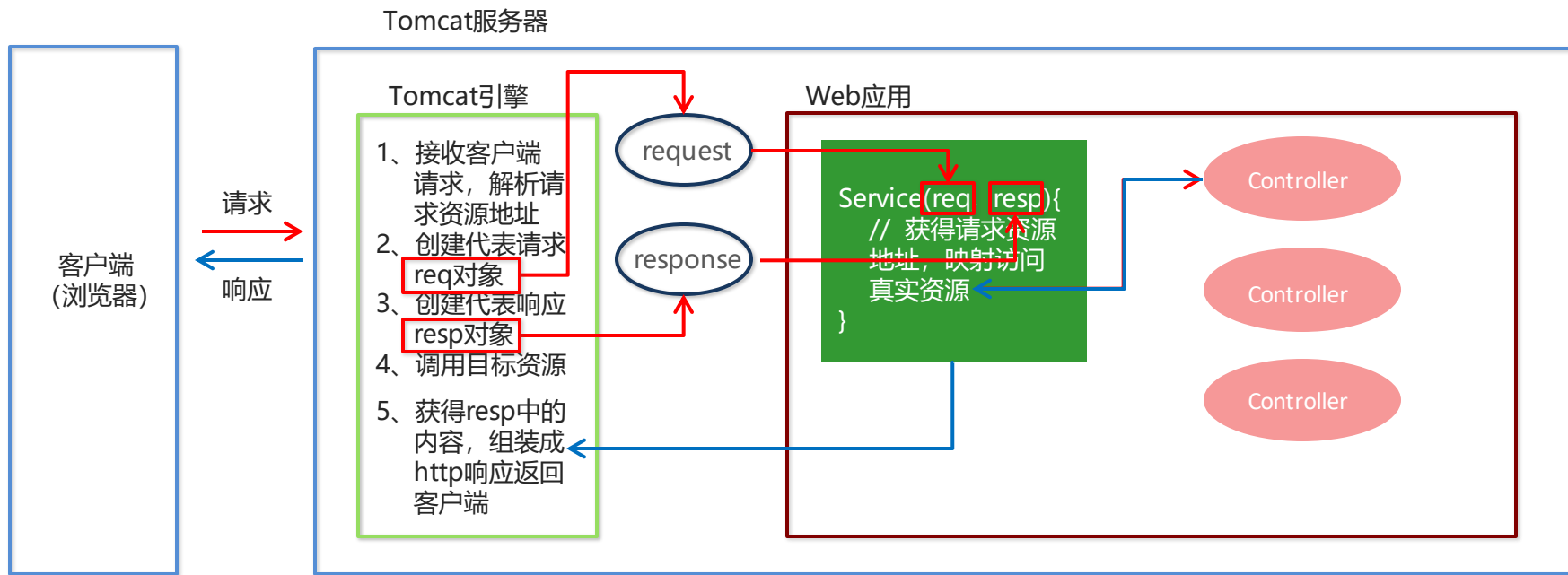
```
quickMethod running.....
```

页面显示



2. SpringMVC 简介

2.3 SpringMVC流程图示



2. SpringMVC 简介

2.4 知识要点

SpringMVC的开发步骤

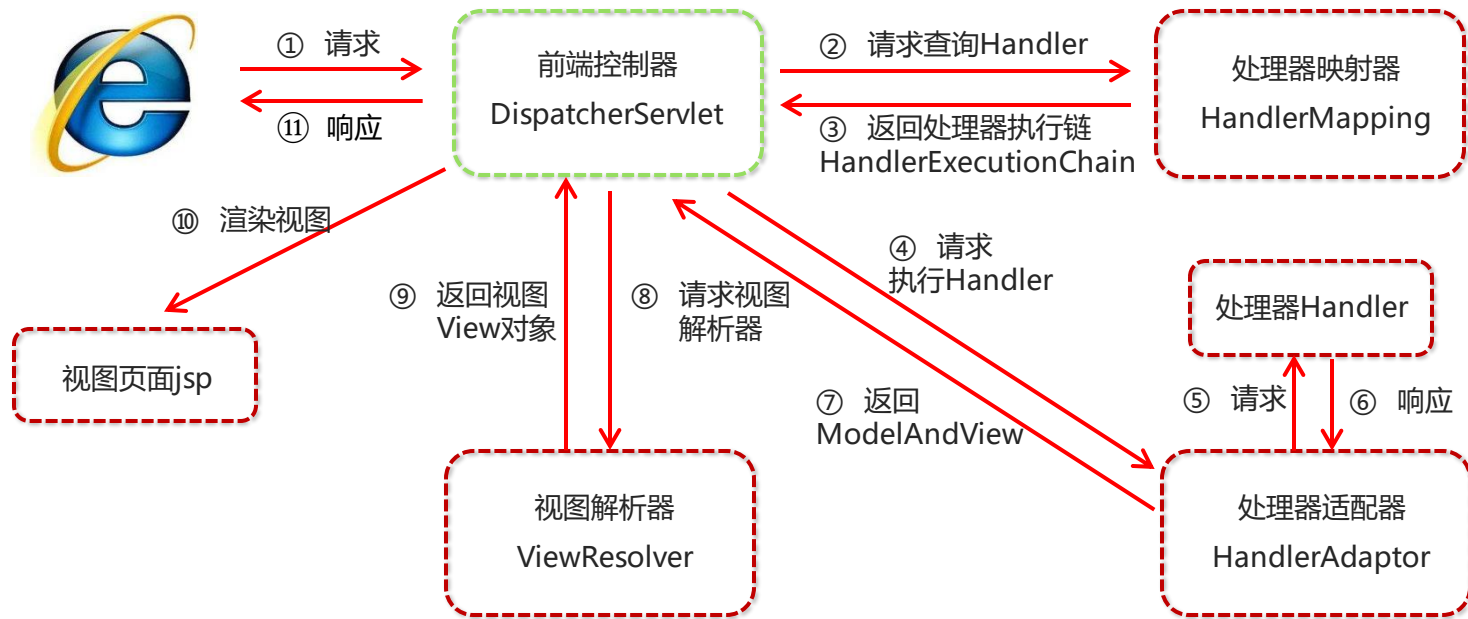
- ① 导入SpringMVC相关坐标
- ② 配置SpringMVC核心控制器DispatchcerServlet
- ③ 创建Controller类和视图页面
- ④ 使用注解配置Controller类中业务方法的映射地址
- ⑤ 配置SpringMVC核心文件 spring-mvc.xml
- ⑥ 客户端发起请求测试

目录 Contents

- ◆ Spring与Web环境集成
- ◆ SpringMVC的简介
- ◆ SpringMVC的组件解析

3. SpringMVC 组件解析

3.1 SpringMVC的执行流程



3. SpringMVC 组件解析

3.1 SpringMVC的执行流程

- ① 用户发送请求至前端控制器DispatcherServlet。
- ② DispatcherServlet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。
- 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(Controller, 也叫后端控制器)。
- ⑥ Controller执行完成返回ModelAndView。
- ⑦ HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet。
- ⑧ DispatcherServlet将ModelAndView传给ViewResolver视图解析器。
- ⑨ ViewResolver解析后返回具体View。
- ⑩ DispatcherServlet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。

3. SpringMVC 组件解析

3.2 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。

3. SpringMVC 组件解析

3.2 SpringMVC组件解析

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

3. SpringMVC 组件解析

3.3 SpringMVC注解解析

@RequestMapping

作用：用于建立请求 URL 和处理请求方法之间的对应关系

位置：

- 类上，请求URL 的第一级访问目录。此处不写的话，就相当于应用的根目录
- 方法上，请求 URL 的第二级访问目录，与类上的使用@RequestMapping标注的一级目录一起组成访问虚拟路径

属性：

- **value**：用于指定请求的URL。它和path属性的作用是一样的
- **method**：用于指定请求的方式
- **params**：用于指定限制请求参数的条件。它支持简单的表达式。要求请求参数的key和value必须和配置的一模一样

例如：

- **params = {"accountName"}**，表示请求参数必须有accountName
- **params = {"money!100"}**，表示请求参数中money不能是100

3. SpringMVC 组件解析

3.3 SpringMVC注解解析

1. mvc命名空间引入

命名空间: `xmlns:context="http://www.springframework.org/schema/context"`

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

约束地址: `http://www.springframework.org/schema/context`

`http://www.springframework.org/schema/context/spring-context.xsd`

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

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

2. 组件扫描

SpringMVC基于Spring容器，所以在进行SpringMVC操作时，需要将Controller存储到Spring容器中，如果使用@Controller注解标注的话，就需要使用<context:component-scan base-

package="com.itheima.controller"/>进行组件扫描。

3. SpringMVC 组件解析

3.4 SpringMVC的XML配置解析

1. 视图解析器

SpringMVC有默认组件配置，默认组件都是`DispatcherServlet.properties`配置文件中配置的，该配置文件地址 `org/springframework/web/servlet/DispatcherServlet.properties`，该文件中配置了默认的视图解析器，如下：

```
org.springframework.web.servlet.ViewResolver=org.springframework.web.servlet.view.I  
nternalResourceViewResolver
```

翻看该解析器源码，可以看到该解析器的默认设置，如下：

```
REDIRECT_URL_PREFIX = "redirect:"    --重定向前缀  
FORWARD_URL_PREFIX  = "forward:"     --转发前缀（默认值）  
prefix = "";           --视图名称前缀  
suffix = "";          --视图名称后缀
```

3. SpringMVC 组件解析

3.4 SpringMVC的XML配置解析

1. 视图解析器

我们可以通过属性注入的方式修改视图的前后缀

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

3. SpringMVC 组件解析

3.5 知识要点

SpringMVC的相关组件

- 前端控制器: DispatcherServlet
- 处理器映射器: HandlerMapping
- 处理器适配器: HandlerAdapter
- 处理器: Handler
- 视图解析器: View Resolver
- 视图: View

SpringMVC的注解和配置

- 请求映射注解: @RequestMapping
- 视图解析器配置:

```
REDIRECT_URL_PREFIX = "redirect:"
```

```
FORWARD_URL_PREFIX = "forward:"
```

```
prefix = "";
```

```
suffix = "";
```



传智播客旗下高端IT教育品牌