第35卷第3期 合肥工业大学学报(自然科学版) Vol. 35 No. 3
2012年3月 JOURNAL OF HEFEI UNIVERSITY OF TECHNOLOGY Mar. 2012

Doi:10.3969/j.issn.1003-5060.2012.03.012

基于Spring MVC框架的Web研究与应用

薛 峰, 梁 锋, 徐书勋, 王彪任

(合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:MVC是软件工程中的一种经典软件架构模式。文章在分析MVC架构基本组成和原理的基础上,详细介绍了Spring MVC架构的组成原理、应用配置;最后,以一个应用案例,详细说明了Web请求在Spring MVC框架中的流转生命周期。

关键词:Spring MVC框架;MVC模式;控制器;视图

中图分类号:TP393 **文献标识码**:A **文章编号**:1003-5060(2012)03-0337-04

Research on Spring MVC framework based Web and its application

XUE Feng, LIANG Feng, XU Shu-xun, WANG Biao-ren

(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: The MVC pattern is a classic software architecture pattern in software engineering. In this paper, based on the analysis of the basic components of the MVC architecture and its principles, the composition and application configuration of the Spring MVC architecture are presented. Then the full life cycle of a Web request in the Spring MVC framework is introduced by using a concrete application example.

Key words: Spring MVC framework; MVC pattern; controller; view

MVC模式(Model View Controller,简称MVC)是软件工程中的一种软件架构模式,把软件系统分为模型(Model)、视图(View)和控制器(Controller)[1]3个部分。Model对象包含数据;View对象负责显示有模型包含的数据,用于与用户交互;Controller对象是介于Model与View之间的桥梁,它可以分发和处理用户的请求,选择适当的视图用于显示模型包含的数据返回给用户。

1 MVC模式基本思想及原理

1.1 MVC模式

Spring框架提供了构建Web应用程序的全功能MVC模块[2],是一种高度可配置的MVC框架,可以定制本地化和主题解析,并提供多种视图技术,实现了控制器、模型对象、分派器以及处理程序对象的多角色分离[3],这种分离让它们更容易进行定制。

MVC是一种著名的设计模式,从本质上讲是对于GOF 23种设计模式中一些基本模式的集合和优化[4]。MVC模式是目前交互式系统中应用最广的一种分层架构,可以很好地隔离用户界面层和业务处理层,对代码进行模块化的划分,将一个系统中的各个功能部分之间进行解耦。

MVC模型强制性地将应用程序的输入、处理和输出分开。在MVC架构中,通过把系统分为3个基本部分,使应用系统结构更清晰,升级、维护更方便。

1.2 MVC模式的结构

MVC模式的结构如图1所示,其包括的模型、视图和控制器分别对应应用的内部数据、数

收稿日期:2011-09-28;修回日期:2011-12-23

基金项目:安徽省自然科学基金资助项目(090412059);北京航天航空大学国家重点实验室开放课题资助项目(BUAA-VR-10KF-05)

作者简介:薛 峰(1978—),男,安徽舒城人,博士,合肥工业大学副教授,硕士生导师.

据视图和输入输出控制部分[5]。

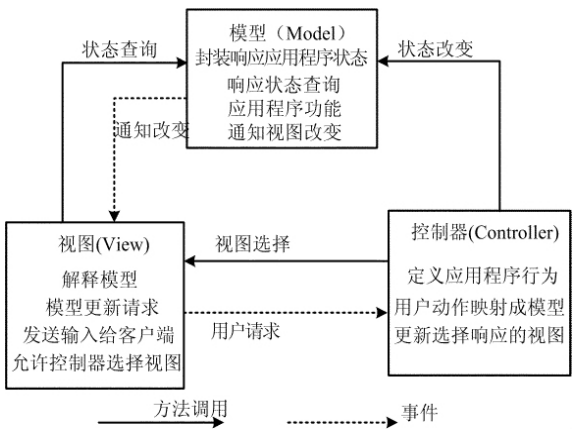

图 1 MVC 结构

1.2.1 模型

模型是与问题相关数据的逻辑抽象,代表对象的内在属性,是整个模型的核心。模型是真正完成任务的代码,接受视图请求的数据,并返回最终的处理结果。

1.2.2 视图

视图是模型的外在表现,也就是界面,提供了模型的表示。视图具有与外界交互的功能,是应用系统与外界的接口,一方面它为外界提供输入手段,并触发应用逻辑运行,另一方面它又将逻辑运行的结果以某种形式显示给外界,但它并不进行任何实际的业务处理。

1.2.3 控制器

控制器是模型与视图的联系纽带,控制器的任务是从用户接受请求,将模型与视图匹配。控制器提取通过视图传输进来的外部信息,并将用户与 View 的交互转换为基于应用程序行为的标准业务事件,再将标准业务事件解析为 Model 对应的动作。同时,模型的更新和修改也将通过控制器来通知视图,从而保持各个视图与模型的一致性。

2 Spring MVC 原理及配置

2.1 Spring MVC 基本组件和处理流程

Spring MVC 实现了 MVC 的核心概念,它为控制器和处理程序提供了大量与此模式相关的功能。并且当向 MVC 添加反转控制时,它使应用程序高度解耦,提供了通过简单的配置更改即可动态更改组件的灵活性。Spring MVC 在接受一个请求时的处理流程,如图 2 所示。

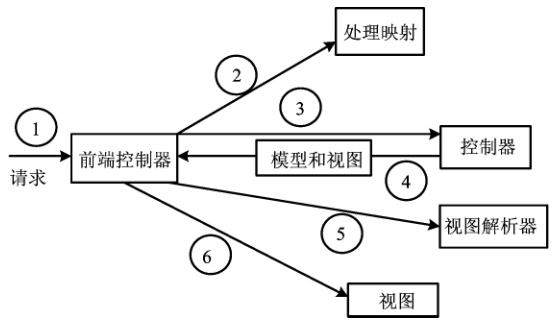

图 2 Spring MVC 处理请求流程

2.1.1 前端控制器(DispatcherServlet)

给处理程序分派请求,执行视图解析。它根据用户的请求将其转发给特定的控制器,然后由该控制器负责处理具体的请求并返回模型和视图对象,分发器再对模型和视图进行解析[6]。

2.1.2 控制器(Controuer)

控制器是实际承担 Web 业务逻辑处理的组件,其再调用具体的业务 Service 完成相应的逻辑请求。缺省的处理器是一个简单的控制器接口,可以实现接口生成应用的控制器。

2.1.3 模型和视图(ModelAndView)

控制器会返回与请求相关的模型和视图,并传递给前端控制器,分发器接收到模型和视图后,将解析分离模型和视图,并将模型加载到视图上渲染给客户端。

2.1.4 视图解析器(ViewResolver)

Spring 提供了视图解析器在浏览器显示模型数据。大多数情况下,模型和视图对象中封装的是逻辑视图名,就需要通过视图解析器根据逻辑视图名,解析成具体的 View 资源,然后才渲染给客户端[7]。Spring 内置了对 JSP、Velocity 模版和 XSLT 视图的支持[8]。

2.2 Spring MVC 应用配置

Spring MVC 的配置主要包括控制器、视图解析器和映射请求。

2.2.1 配置 DispatcherServlet

DispatcherServlet 配置格式如下:

```
〈servlet〉
    〈servlet-name〉Spring MVC Dispatcher Servlet〈/servlet-name〉
    〈servlet-class〉org.springframework.web.servlet.DispatcherServlet〈/servlet-class〉
    〈init-param〉
        〈param-name〉contextConfigLocation〈/param-name〉
        〈param-value〉/WEB-INF/spring/*.xml〈/pa-
```

```
        ram-value〉
    〈/init-param〉
    〈load-on-startup〉1〈/load-on-startup〉
〈/servlet〉
```

2.2.2　配置 ViewResolver

ViewResolver 提供了从视图名称到实际视图的映射，即在模型视图名称添加前后缀。View 处理请求的准备工作，并将该请求提交给某种具体的视图技术，默认的为 jsp 视图。ViewResolver 的配置格式如下：

```
〈bean class =" org. springframework. web. servlet. view. InternalResourceViewResolver"〉
    〈property name =" prefix"  value =" /WEB - INF/views"〉
    〈proprty name =" suffix"  value =" jsp" 〉
〈/bean〉
```

2.2.3　使用@Controller 定义一个控制器

Spring MVC 自 2.5 版本开始提供基于注解的 Controller，@Controller 注解将任意的 POJO 类标注成控制器类，被标注的控制器类不需要实现特定的框架 Controller 接口，也不必继承特定的框架基类。

2.2.4　使用@RequestMapping 映射请求

@RequestMapping 注解可以标注在类定义处，将 Controller 和特定请求关联起来；还可以在方法签名处，以便进一步对请求进行分流。在类声明处标注的 @RequestMapping 相当于让 POJO 实现了 Controller 接口，而在方法定义处相当于让 POJO 扩展 Spring 预定义的 Controller（如 SimpleFormController 等）。

2.3　Spring MVC 框架中 XML 配置的缺点

Spring 框架的核心是一个 Bean 容器，结合 AOP 技术，把事务管理融合进去，就构建了一个无状态会话 Bean EJB 容器相对应的轻量级容器。SpringMVC 的很多组件都是通过 XML 进行配置，但存在如下缺点：

（1）必须结合配置文件才能弄清楚 java 对象间的关系，代码的可读性变差。

（2）由于依赖关系通过 XML 文件描述，所有这种关系设置的正确性，只能在运行时期检验，而在程序编译阶段无法判别。

（3）由于引入 XML 文件描述和装配 JavaBean，当系统的 JavaBean 达到一定数量时，系统很难维护，而且 XML 越多，可读性和可跟踪性就越差。建议可以使用注释配置完成部分 XML 配置的功能，从而有效减少 XML 配置的工作量，提高程序的内聚性。

2.4　Spring MVC 请求流程时序

SpringMVC 请求流程的时序，如图 3 所示。

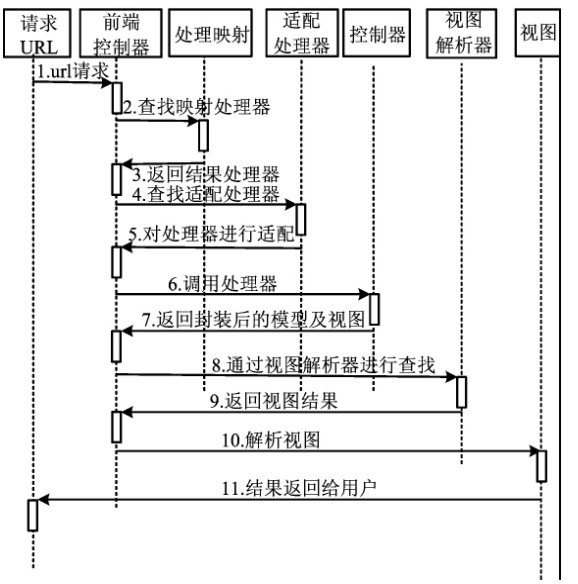

图 3　Spring MVC 请求流程时序

3　Spring MVC 的简单应用

采用应用案例详细说明 Spring MVC 的实现过程，其基本功能是实现个人信息的基本管理，如姓名、性别、出生信息、工作经历等。下面以修改用户基本信息的请求响应，详细介绍 Spring MVC 的处理流程。

（1）进入用户基本信息修改界面，如图 4 所示。

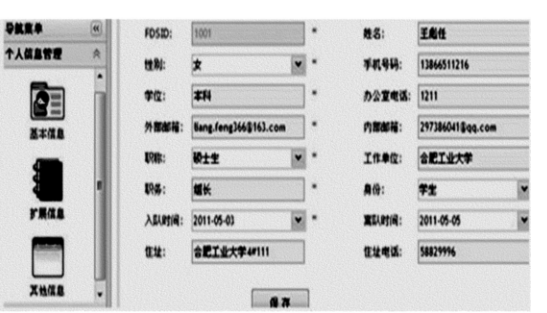

图 4　用户基本信息修改界面

填写相应的信息后，点击“保存”按钮，向服务器发送 post 请求。

（2）分发器得到请求，通过控制器映射找到负责具体业务逻辑处理的控制器，并将请求转发给该控制器。在应用中该控制器为 MemberController，对应的处理方法为：

```
@RequestMapping(" baseinfo/member/update")
public void updateMember(Member member, ModelMap model){
    memberService. updateMember (member);
    model. put(" result"," success");
}
```

(3) 控制器调用 MemberServiceImpl 保存用户信息,其处理方法如下:

```
@Override
public void updateMember(Member member){
    getSqlMapClientTemplate. update(" updateMember", member);
    User user = new User();
    user. setFdsId(member. getFdsId());
    user. setName(memer. getName());
    user. setStatus(-1);
    model. put(" result"," success");
    getSqlMapClientTemplate. update(" updateUser", user);
}
```

(4) MemberServiceImpl 调用 DAO 或者持久层 iBATIS 更新数据库里用户的基本信息。

(5) 控制器创建 ModelAndView 对象,并将业务逻辑产生的数据模型和相关视图封装到 ModelAndView 对象中,返回分发器。

(6) 分发器得到并解析 ModelAndView 对象。在应用中,ModelAndView 中没有存放任何 View 的信息,则对应的视图仍为用户请求所在的当前视图。

(7) 前台取到了放在 ModelMap 中的"success",则显示表示本次修改用户信息成功,如图5所示。

图5 用户更新信息成功

4 结束语

Spring 主要思想是利用 IoC 依赖注入机制,实现面向接口编程,以达到降低耦合度、减少代码量的目的。Spring MVC 清楚地分离了 Web 应用中的所有职责,框架提供依赖注入机制,降低组件的耦合度。本文重点研究了 Spring MVC 的结构组成,详细介绍了 Web 请求在 Spring MVC 框架中的流转。此外,Spring MVC 中可通过配置拦截器,实现基于 AOP 的权限控制,如何在 Spring MVC 基本流程基础上,设计粒度更细的权限管理技术是进一步研究的方向。

[参 考 文 献]

[1] 任中方,张 华. MVC 模式研究的综述[J]. 计算机应用研究,2004,21(10):1-4.

[2] 张 宇,王映辉,张翔南. 基于 Spring 的 MVC 框架设计与实现[J]. 计算机工程,2010,36(4):59-62.

[3] 庄少炖. 基于 Spring 的轻量级 Web 框架研究与实现[D]. 成都:电子科技大学,2009.

[4] 邹存洁. 基于 MVC 模式的 Spring 框架的应用与研究[D]. 大连:大连海事大学,2006

[5] 陈 平. 基于 Spring 的轻量级 Web 框架的研究与设计[D]. 江苏镇江:江苏大学,2005.

[6] 符培炯,杜忠军. Spring 在实现 MVC 构架中的应用[J]. 计算机技术与发展,2006,16(6):236-238.

[7] 张 利,吴传胜,崔 雷,等. 应用 MVC 模式构建 Web 信息系统框架研究[J]. 合肥工业大学学报:自然科学版,2007,30(7):829-832.

[8] 王承冠,陆金桂. 基于 SPRING 和 VELOCITY 的 WEB 开发模式及其应用[J]. 微计算机信息,2006,22(30):230-232.

(责任编辑 吕 杰)