文章编号：2096-1472(2017)-01-05-03

基于Spring MVC+JDBCTemplate的Web系统的研究与应用

赵 璘[1]，王红霞[2]
（1.大连东软信息学院软件工程系，辽宁 大连 116023;
2.大连市公安局网安支队，辽宁 大连 116011）

摘 要：Spring MVC以其松散耦合的特性在业内被广泛使用，JDBCTemplate对数据库的操作在JDBC层面做了深层次的封装，简化了项目中繁琐的JDBC操作。Spring MVC与JDBCTemplate的有机结合，优化了软件开发的过程。在分析Spring MVC与JDBCTemplate的技术要点、工作原理的基础上，通过实例，展示了Spring MVC+JDBCTemplate的整合应用给项目带来的便利。

关键词：SpringMVC；JDBCTemplate；封装；整合
中图分类号：TP311.5 **文献标识码**：A

Research and Application of the Web System Based on Spring MVC+JDBCTemplate

ZHAO Lin[1],WANG Hongxia[2]
(1.*Dalian Neusoft University of Information*,*Department of Software Engineering*,*Dalian* 116023,*China*;
2.*Dalian Municipal Public Security Bureau Network Security Detachment*,*Dalian* 116011,*China*)

Abstract: Spring MVC is widely used in the industry because of its loose coupling characteristics.JDBCTemplate makes the database operations to be encapsulated at the JDBC level and simplifies the tedious JDBC operation in projects. It optimizes the software development process to integrate Spring MVC and JDBCT Template.Based on the analysis of the technical points and working principles of Spring MVC and JDBCTemplate,the study implements some experiments to present the convenience brought by the integration of Spring MVC and JDBCTemplate to the project.

Keywords: Spring MVC;JDBCTemplate;encapsulation;integration

1 引言(Introduction)

在Web应用中，表现层是Web应用不可忽略的重要组成部分，Spring为表现层提供了一个十分优秀的Web框架——Spring MVC。与大多数其他的Web框架一样的设计理念，都是遵循MVC的思想架构。Spring MVC通过注解的方式，让POJO成为处理请求的控制器，无需实现任何的接口[1]。另外，Spring MVC最大的优点就是松散耦合，更具有灵活性和可扩展性[2]。

JDBC已经能够满足大部分用户最基本的对数据库的需求，但是在使用JDBC时，应用必须自己来管理数据库资源。JDBCTemplate正是为了减少JDBC繁琐的代码而设计出来的。Spring对数据库操作需求提供了很好的支持，并在原始JDBC基础上，构建了一个抽象层，提供了许多使用JDBC的模板和驱动模块，为Spring应用操作关系数据库提供了更大的便利[3-6]。Spring封装好的模板，封装了数据库存取的基本过程，方便使用。

2 Spring MVC的技术要点(Technical points of Spring MVC)

2.1 Spring MVC的组件

(1)DispatherServlet：前置控制器，负责接收并处理所有的web请求，根据HandlerMapping找到具体的Controller，由Controller完成具体的处理逻辑。

(2)HandlerMapping：负责处理web请求和具体的Controller之间的映射关系匹配。

(3)Controller：DispatherServlet的次级控制器，web请求的具体处理者。DispatherServlet获得HandlerMapping的返回结果后，调用Controller的方法处理前端发出的请求，处理结果通过ModelAndView对象返回。

(4)ViewResolver：用来处理视图名与具体的View实例之间的映射对应关系。根据ModelAndView中的视图名查找相应的View实现类，然后将查找的结果返回给DispatcherServlet，DispatcherServlet最终会将ModelAndView中的模型数据交给返回的View处理最终的视

图渲染工作。

(5)View：为支持多种视图技术而存在，统一抽象视图的生成策略，根据模型数据输出具体的视图。

2.2 Spring MVC的体系架构

Spring MVC是基于Model 2实现的，Model 2是MVC模型在Java Web应用中的一个变体。Spring MVC的底层机制是MVC，是利用处理器分离模型、视图和控制达到松散耦合的效果。Spring MVC的框架模型如图1所示。

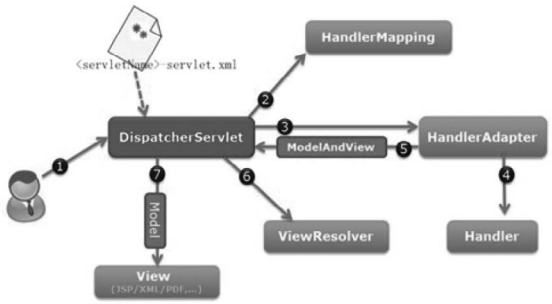

图1 Spring MVC的框架模型图

Fig.1 The framework model diagram of Spring MVC

从接收请求开始，到返回响应为止，Spring MVC框架中的各个组件各司其职，通力合作，有序配合地完成各自的工作。在整个Spring MVC框架中，DispatcherServlet处于核心领导地位。它负责组织和协调不同的组件完成从请求到响应的任务。其处理请求的整体过程如下：

(1)整个过程从客户端发出一个HTTP请求开始，Web应用服务器接收请求，如果与DispatcherServlet的请求路径相匹配，Web容器就将该请求转发给DispatcherServlet处理。

(2)DispatcherServlet接收到这个请求后，将根据请求的信息及HandlerMapping的配置找到处理请求的处理器，即Handler。注意：Spring MVC中并没有定义Handler接口。

(3)当DispatcherServlet根据HandlerMapping得到对应当前请求的Handler后，通过HandlerAdapter对Handler进行封装。再通过统一的适配器接口调用Handler(HandlerAdapter是Spring MVC的框架级接口)。

(4)处理器完成业务处理后，返回一个ModelAndView给DispatcherServlet，ModelAndView包含视图逻辑的名称和模型数据的信息。

(5)ModelAndView中包含的是逻辑视图名，不是真正意义的视图对象。DispatcherServlet通过ViewResolver完成从逻辑视图名到真实视图对象的解析。

(6)当得到真正的视图对象后(View)，DispatcherServlet就使用这个View对象对ModelAndView中的模型数据进行视图渲染。

(7)最后客户端得到的响应信息可能是一个页面(HTML、JSP)，也可能是XML、JSON串或其他不同的媒体形式。

2.3 Spring MVC配置

(1)配置DispatcherServlet

DispatcherServlet在web.xml中进行配置，它让Spring MVC能够生龙活虎。基本代码如下所示：

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

3 JDBCTemplate的应用(The application of JDBCTemplate)

由于JDBC API过于底层，直接进行JDBC编写数据库程序，需要完成数据库代码(SQL)、还需要编写获取JDBC连接、异常处理、释放连接资源等代码，实在是过于繁琐。Spring JDBC通过模板和回调机制简化了JDBC编写数据库程序的复杂度。借助JDBCTemplate，程序员只需要编写处理业务的核心SQL语句即可。

3.1 JDBCTemplate配置

在Spring的配置文件ApplicationContext.xml中进行数据源的配置。以下代码以mariadb数据库为例，代码如下：

```
<bean id="dataSource"class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close"p:driverClassName="org.mariadb.jdbc.Driver"
p:url="dbc:mysql://localhost:3306/asms?useUnicode=true&characterEncoding=UTF-8"
p:username="asms"
p:password="asms"/>
<bean id="jt"class="org.springframework.jdbc.core.JdbcTemplate"p:dataSource-ref="dataSource"/>
```

<bean id="transactionManager"class="org.springframework.jdbc.datasource.DataSourceTransactionManager"p:dataSource-ref="dataSource"/>

3.2 JDBCTemplate中的常用方法

(1)update()方法用于执行数据库表的新增、修改、删除的SQL语句。

(2)batchUpdate()方法用于执行批处理相关的SQL语句。

(3)query()方法及queryForXXX()方法：用于执行查询相关的SQL语句。其中queryForXXX()根据返回对象类型进行方法的选择(queryForObject,queryForList等)。

(4)call()方法用于执行存储过程、函数相关的SQL语句。

3.3 JDBCTemplate实例

在代码中，以向表nps_classes中插入数据为例。首先定义SQL语句，然后使用JDBCTemplate执行该SQL。

```
privatestaticfinal String SQL_ADD_CLASS="insert into nps_classes(classid,classname,classdesc,batid,apid) values(?,?,?,?,?)";
    publicvoidaddClass(Map map) {
        Object[]params=MapUtil.getObjectArrayFromMap(map, "classid,classname,classdesc,batid,apid");
        jt.update(SQL_ADD_CLASS,params);
    }
```

4 应用实例(Application instance)

运用实际案例介绍Spring MVC在项目中的实现过程。基本功能说明：毕业生首次登录系统要完成自主注册功能，该功能分为两步完成：第一步，通过姓名和身份证号在毕业生信息中进行校验，校验成功后进入第二步，校验失败重新填写姓名和身份证号进行校验。第二步：在第一步校验成功后，需要填写姓名拼音、电子邮箱、验证码、密码、确认密码提交至系统保存，保存成功即毕业生注册成功。

(1)在注册第二步，填写相应信息后，点击“下一步”按钮，向服务端发送post请求，如图2所示。

图2 详细信息界面

Fig.2 The detail page

(2)分发器得到客户端请求后，通过控制器映射匹配到负责业务逻辑处理的控制器，并将请求转发给该控制器。该控制器所在的Java类为StuRegisterController.java，代码如下：

```
@RequestMapping(value="/stu/registerasmsstep2.do")
public Map<String,Object>registerasmsstep2(String studentno,String stupwd,Stringstudentnameen)
        throwsIOException {
        Map<String,Object>jsonrslt=new HashMap<String,Object>();
        boolean flag=false;
        flag=this.registerService.updateregisterasmsstep2(studentno,stupwd,studentnameen);
        jsonrslt.put("flag",flag);
        returnjsonrslt;
    }
```

(3)控制器调用RegisterServiceImpl处理注册第二步的业务逻辑，其处理方法如下：

```
@Override
publicboolean updateregisterasmsstep2(String studentno,Stringstupwd,Stringstudentnameen) {
    boolean flag=false;
    Map<String,String>rmParam=new HashMap<String,String>();
    rmParam.put("studentno",studentno);
    rmParam.put("stupwd",stupwd);
    rmParam.put("studentnameen",studentnameen);
    int i=jdbctemplate.update(sql.updateregisterasmsstep2(),rmParam);
        if(i==1){
            flag=true;
        }
        return flag;
    }
```

(4)由于业务要求第一步注册校验时已将毕业生信息插入到数据表中，所以第二步注册仅仅是更新部分数据信息，RegisterServiceImpl中的JDBCTemplate更新数据表的信息，调用SQL如下：

```
public String updateregisterasmsstep2() {
        return"update ASMS_STUREGINFO set
```

stupwd=:stupwd,studentnameen=:studentnameen where studentno=:studentno";

}

(5)响应返回至客户端获取flag数据，则显示“注册成功，请使用学号和密码进行系统登录”信息，如图3所示。

图3 注册成功界面

Fig.3 The successful registration page

5 结论(Conclusion)

本文分析了Spring MVC的技术要点和JDBCTemplate的应用。Spring MVC技术要点包括组件、框架模型和配置，JDBCTemplate的应用包括配置和常用方法。最后通过实际项目，介绍了web请求在Spring MVC中的应用，以及JDBCTemplate如何操作数据表。

综上得出：Spring MVC大大简化了程序开发的繁琐度，JDBCTemplate降低了编写数据库程序的复杂度。Spring MVC+JDBCTemplate的组合值得在Web应用软件开发中广泛使用。

参考文献(References)

[1] 张文宇,许明健,薛昱.论spring的零配置与XML配置[J].计算机系统应用,2015,24(2):270-275.

[2] 周燕玲.Spring MVC框架开发WEB应用程序的探索与研究[J].科技广场,2016(6):25-28.

[3] 叶雯.基于Spring MVC框架的Web登录模块的设计与实现[J].电脑知识与技术,2013(35):7983-7984.

[4] Zhang Chao,Zhao Ping,He Jing.Design and Implementation of the Control System Software Based on MVC Model[J].High Power Laser and Particle Beams,2013(S1):91-95.

[5] TianPengfei,Tian Di,Yang Guang.Design and Implementation of LIBS Software Based on MVC Architecture[J].Journal of Jilin University(Engineering and Technology Edition),2016(1):242-245.

[6] Lin HC,et al.Development of a Real-Time Clinical Decision Support System upon the Web MVC-Based Architecture for Prostate Cancer Treatment[J].BMC Med Inform DecisMak,2011(926):3306-3309.

作者简介：

赵 璘(1981-)，男，硕士，助教.研究领域：软件工程.

王红霞(1982-)，女，硕士，工程师.研究领域：计算机软件与理论.

(上接第14页)

数目和开关大小，增加了开关的复杂度从而相应地增加路由延迟和开关延迟，增加虚拟通道就会相应的加大缓冲器的大小，而缓冲器是路由器功耗最大、产热最多、成本最高的模块，所以增加虚拟通道的数目反而会影响网络的性能以及增大成本与功耗。因此，新提出来的WRD在保证网络性能没有下降，甚至在某些通讯模式下有所提升的情况下，大大降低了虚拟通道的使用数目，这是一个很大的改进。

参考文献(References)

[1] W.J.Dally.Performance Analysis of K-ary N-cube Interconnection Networks[J].IEEE Transactions on Computers,1990,39(6):775-785.

[2] J. Duato.A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks[J].IEEE Transactions on Parallel and Distributed Systems,1993,4(12):1320-1331.

[3] J.Upadhyay,V.Varavithya,P.Mohapatra.A Traffic Balanced Adaptive Wormhole Routing Scheme for Two-Dimensional Meshes[J].IEEE Transactions on Computers,1997(5):190-197.

[4] A.Agarwal,et al.Johnson.The Mit Alewife Machine:Architecture and Performance[J].In 25 years of the International Symposia on Computer Architecture,ACM Press,1998.

[5] W.Dally and B.Towles.Principles and Practices of Interconnection Networks[J].Morgan Kaufmann,2004.

[6] H.Sullivan and T.R.Bashkow.A Large Scale,Homogeneous,Fully Distributed Parallel Machine[J].In Proceedings of the 4th Annual Aymposium on Computer Architecture,ACM Press,1977.

[7] 张立,戴一奇.随机Oblivious路由算法中随机性与时间代价的研究[J].计算机学报,1996(5):388-397.

作者简介：

任一曼(1992-)，女，硕士生.研究领域：片上网络.