



设计模式与系统架构分析报告



系（院）：_____ 计算机科学学院 _____

课 程：_____ 设计模式与系统架构 _____

指导教师：_____ 向华 _____

专业班级：_____ 计科 11402 _____

姓 名：_____ 陈昌杭 _____

学 号：_____ 201403613 _____

设计时间：_____ 2017.6.12 - 2017.6.25 _____

目录

- 1.观察者模式 1
 - 1.1.定义 1
 - 1.2.UML 图 1
 - 1.3.理解 2
 - 1.4.工程项目 2
- 2.Spring MVC 4
 - 2.1. Spring MVC 架构流程 4
 - 2.2. Spring MVC 架构组件 5
 - 2.3. Spring MVC 架构开发 6
 - 2.4. Spring MVC 架构项目 8
 - 2. 5. Spring MVC 架构拓展 8

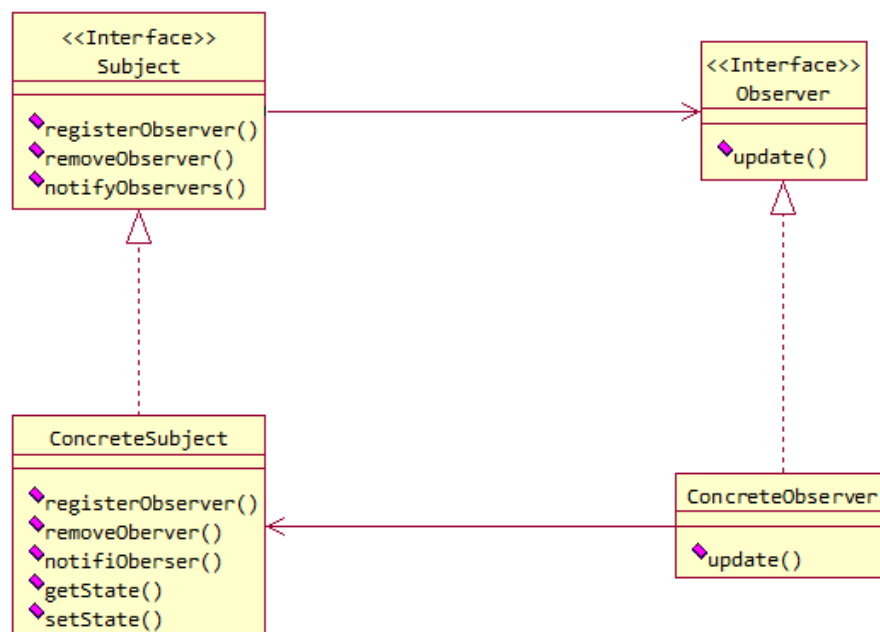
<https://github.com/chenchanghang>

1.观察者模式

1.1.定义

定义了对对象之间的一对多的依赖，当一个对象变化状态时，它对所有依赖者都会收到通知并更新。

1.2.UML 图



对与上面的 UML 图的说明：

1.Subject(目标)

目标知道它的观察者。可以有任意多个观察者观察同一个目标。提供注册和删除观察者对象的接口。

2.Observer(观察者)

为那些在目标发生改变时需获得通知的对象定义一个更新接口。

3.ConcreteSubject(具体目标)

将有关状态存入各 **ConcreteObserver** 对象。当它的状态发生改变时，向它的各个观察者发出通知。

4.ConcreteObserver(具体观察者)

维护一个指定 ConcreteSubject 对象的引用。存储有关状态，应与目标的状态保持一致。实现 Observer 的更新接口以使自身状态与目标的状态保持一致。

1.3.理解

1.3.1.观察者模式思维与常规思维对比

我们的常规思维方式会认为：所有的观察者会在一定的反应时间内发现被观察者对象的变化。这两种思维方式存在观察者与被观察者谁发现变化？谁主动？谁被动？的差异。

下面我举例说明：假设对象 A 的观察者有对象 B、C 和 D；当对象 A 发生变化时。使用常规思维方式，对象 A 的所有观察者主动地发现对象 A 的变化，这时观察者是主动的，而被观察者是被动的，观察者发现被观察者的变化。使用观察者模式的思维方式，对象 A 变化后，主动地告知其所有的观察者自己的变化，这时被观察者是主动的，而观察者是被动的，被观察者告知观察者自己的变化。

1.3.2.类比中断与轮询

这两种思维方式的对比让我联想到在计算机内部的实现机制。在计算机组成原理课程中提到，CPU 对于外部 I/O 事件的两种不同的程序实现方式——查询方式和中断方式。

在这个过程中，I/O 类似于被观察者，CPU 类似观察者，查询方式相当于常规思维，这时，CPU 不断地查询 I/O 设备的状态；如果 I/O 设备一发生变化，CPU 立马可以察觉发现，并根据变化执行不同的操作。而中断方式就完全是观察者模式的体现，如果 I/O 设备发生变化，I/O 设备会产生中断信告知 CPU，CPU 接收到 I/O 后执行相应操作；其他时间 CPU 保持空闲状态或执行其他操作，这样可以节省 CPU 资源，这个类比到程序员开发中也有同样的益处。

1.4.工程项目

1.4.1.github

这部分的讲解是引用 github 上的工程项目，作者原仓库链接地址为：
<https://github.com/zhanglei-workspace/shopping-management-system.git>。

这个仓库是一个 Java 学习的工程项目的仓库，里面有四个项目，前三个是 Java 学习到不同的程度，使用不同的技术的实现同样的功能——商品管理系统。最后一个是个结合之前所学的所有知识，并加以运用的综合项目，这个项目由于我能力有限，我没弄明白这个项目具体结果。这次使用的是第二个项目——基于 JSP 实现的商品管理系统。

由于基于 JSP 实现的商品管理系统使用 Oracle 数据库作为后台数据库,再加上使用了一些 JavaWeb 框架。在本机调试时存在一些环境配置问题,更无法对程序进一步的修改。

1.4.2.需求分析和功能分析

这个工程项目是商品管理系统,实现了对商品的买卖操作。在工程项目中,只有两个实体模型类:售货员类和商品类;实现的功能有售货员注册登录退出系统和售货员对商品进货和出售的操作。这个工程项目的核心在于实现对商品的各种操作,由于商品信息都在数据库中;所以,本质就是实现对数据库的增删改查操作。

1.4.3.观察者设计模式思想

一方面,工程项目中使用了 MVC 模式(model-view-controller),这个模式本来就是观察者模式思想的体现;model 是被观察者,view 和 controller 都是观察者;当 model 发生变化时,model 会自己发消息通知 view 和 controller。其中原理上面的理解我就不再重复。

另一方面,工程项目中只有两个类:商品类和售货员类;其中,售货员类是观察者,商品类是被观察者。当商品数量达到最高上限或最低下限时,会自动通知售货员停止进货和及时补货操作。

举例说明:在 shopping-management-system\1\#Java 菜鸟项目
\\lyons.eaby\src\lyons\goods\BuyGoods.java 中的 127 至 137 行之间的这段代码

```
int commodity_balance = -1;
System.out.println(Integer.parseInt(goods[4]));
commodity_balance = Integer.parseInt(goods[4])-1; //目前是默认每次修改一个
System.out.println(commodity_balance);
if (commodity_balance >= 0)
{
    pstmtCommodity.setInt(1,commodity_balance);
}else
{
    String failNumber = "数据库中商品不足";
    messShopping(request,response,failNumber);
}
```

补充说明:由于被观察者是商品类,其本质是数据库里的数据;所有对于观察者注册和删除被观察者过程就会比较特殊,体现为连接和关闭数据的过程,

实现过程参考代码: shopping-management-system\1\#Java 菜鸟项目
\\lyons.eaby\src\lyons\db 目录下的 Dbconn.java 和 Dbclose.java 这个两个类的实现代码。

2.Spring MVC

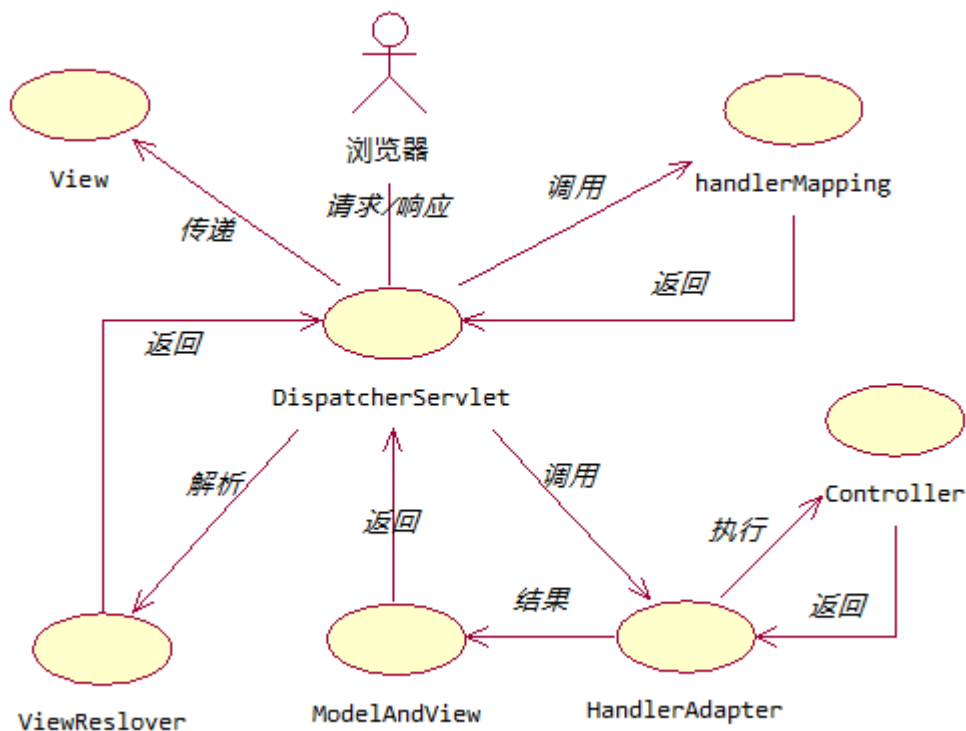
对于 Spring MVC 框架的学习来源于传智播客的公开课视频，由于时间比较仓促，只能根据自己的接受能力，讲述自己的理解。对于使用的项目也是直接使用的是公开课中的教学源码，已上传到我的仓库：

https://github.com/chenchanghang/Design_Patterns/tree/master/Design_Patterns_05/springmvcfirst1208。

2.1. Spring MVC 架构流程

Spring MVC 是 Spring 框架中的重要 MVC 架构。

Spring Web MVC 是一种基于 Java 的实现了 Web MVC 设计模式的请求驱动类型的轻量级 Web 框架，即使用了 MVC 架构模式的思想，将 web 层进行职责解耦，基于请求驱动指的就是使用请求-响应模型。



Browser/Server 模式下的流程：

1. 浏览器发送请求 Request 给前端控制器 DispatcherServlet。
2. 前端控制器 DispatcherServlet 调用处理器映射器 handlerMapping 返回后端处理器 Controller。
3. 前端控制器 DispatcherServlet 通过处理器适配器 HandlerAdapter 执行后端控制器 Controller 返回模型视图 ModelAndView。
4. 前端控制器 DispatcherServlet 将模型视图 ModelAndView 传给视图解析器 ViewResolver 解析返回视图 View。

- 5.前端控制器 DispatcherServlet 渲染视图 View。
6. 前端控制器 DispatcherServlet 返回响应 Respond 给浏览器。

2.2. Spring MVC 架构组件

2.2.1.前端控制器 DispatcherServlet

接收到浏览器请求 Request 后的前端控制器 DispatcherServlet 作为整个流程中的核心，处理各项事宜，最后将结果返回响应 Respond。

2.2.2.处理器映射器 handlerMapping

处理器映射器 handlerMapping 根据请求 Request 寻找后端处理器 Controller 并返回给前端控制器 DispatcherServlet。

2.2.3.处理器适配器 HandlerAdapter

前端控制器 DispatcherServlet 通过处理器适配器 HandlerAdapter 执行后端处理器 Controller。

2.2.4.后端处理器 Controller

在前端控制器 DispatcherServlet 控制下执行后端处理器 Controller，对浏览器的请求 Request 进行处理。

2.2.5.模型视图 ModelAndView

执行后端处理器 Controller 获得结果模型视图 ModelAndView，并返回给前端控制器 DispatcherServlet。

2.2.6.视图解析器 ViewResolver

将前端控制器 DispatcherServlet 传递过来的模型视图 ModelAndView 解析。
















2.2.7.视图 View

由视图解析器 ViewResolver 解析而得视图 View，最后用前端控制器 DispatcherServlet 渲染，返回响应 Respond 给浏览器。

2.3. Spring MVC 架构开发

2.3.1. 开发环境

- 1.系统: Windows 10
- 2.Java: jdk1.7
- 3.IDE: MyEclipse2014
- 4.服务器: Tomcat7
- 5.数据库: MySQL5.5
- 6.jar 包:

-  commons-logging-1.1.1.jar
-  jstl-1.2.jar
-  spring-aop-3.2.0.RELEASE.jar
-  spring-aspects-3.2.0.RELEASE.jar
-  spring-beans-3.2.0.RELEASE.jar
-  spring-context-3.2.0.RELEASE.jar
-  spring-context-support-3.2.0.RELEASE.jar
-  spring-core-3.2.0.RELEASE.jar
-  spring-expression-3.2.0.RELEASE.jar
-  spring-jdbc-3.2.0.RELEASE.jar
-  spring-orm-3.2.0.RELEASE.jar
-  spring-test-3.2.0.RELEASE.jar
-  spring-tx-3.2.0.RELEASE.jar
-  spring-web-3.2.0.RELEASE.jar
-  spring-webmvc-3.2.0.RELEASE.jar

2.3.2. 配置前端控制器 DispatcherServlet

在 web.xml 中配置前端控制器 DispatcherServlet。


```

<!-- springmvc前端控制器 -->
<servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- contextConfigLocation配置springmvc加载的配置文件（配置处理器映射器、适配器等等）
    如果不配置contextConfigLocation，默认加载的是/WEB-INF/servlet名称-servlet.xml（springmvc-servlet.xml）
    -->
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:springmvc.xml</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <!--
    第一种：*.action，访问以.action结尾 由DispatcherServlet进行解析
    第二种：/，所以访问的地址都由DispatcherServlet进行解析，对于静态文件的解析需要配置不让DispatcherServlet进行解析
    使用此种方式可以实现 RESTful风格的url
    第三种：/*，这样配置不对，使用这种配置，最终要转发到一个jsp页面时，
    仍然会由DispatcherServlet解析jsp地址，不能根据jsp页面找到handler，会报错。
    -->
    <url-pattern>*.action</url-pattern>
</servlet-mapping>

```

2.3.3.配置处理器映射器 handlerMapping

在 springmvc.xml 中配置处理器映射器 handlerMapping。

```

<!--注解映射器 -->
<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>

```

2.3.4.配置处理器适配器 HandlerAdapter

```

<!-- 处理器适配器 所有处理器适配器都实现 HandlerAdapter接口 -->
<bean
    class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />

```

2.3.5.编写和配置后端处理器 Controller

继承接口 Controller，实现对浏览器的请求 Request 进行处理，返回模型视图 ModelAndView。

```

public class ItemsController1 implements Controller {

    @Override
    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws Exception {

```

在 springmvc.xml 中配置后端处理器 Controller。

```

<bean id="itemsController1" name="/queryItems_test.action" class="cn.itcast.ssm.controller.ItemsController1" />

```

2.3.5.编写视图 View

使用 Browser/Server 模式，对于 JSP 视图。

查询条件:

<input type="text"/>				
商品列表:				
商品名称	商品价格	生产日期	商品描述	操作
联想笔记本	6000.0		ThinkPad T430 联想笔记本电脑!	修改
苹果手机	5000.0		iphone6苹果手机!	修改

2.3.6.配置视图解析器 ViewResolver

在 springmvc.xml 中配置解析器 ViewResolver。

```
<!-- 视图解析器
解析jsp解析，默认使用jstl标签，classpath下的得有jstl的包
-->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <!-- 配置jsp路径的前缀 -->
    <property name="prefix" value="/WEB-INF/jsp/" />
    <!-- 配置jsp路径的后缀 -->
    <property name="suffix" value=".jsp" />
</bean>
```

2.4. Spring MVC 架构项目

这个项目是结合 MyBatis 和 Spring MVC 的综合项目。

MyBatis是一个优秀的持久层框架，Mybatis通过xml或注解的方式将要执行的各种statement（statement、preparedStatemnt、CallableStatement）配置起来，并通过java对象和statement中的sql进行映射生成最终执行的sql语句，最后由mybatis框架执行sql并将结果映射成java对象并返回。

项目实现的是商品订单的查询功能。在之前使用Mybatid框架的项目的基础上，进一步开发，使用Spring MVC架构实现的。

2. 5. Spring MVC 架构拓展

Spring MVC 的基本开发过程中的配置过程，上面花了大量篇幅讲解；其实这只是基础，Spring MVC 还有另一种模式——注解开发。通常情况下我们会在 xml 配置文件中 action、service、dao 等层的声明，然后并告知框架我们想要的注入方式，然后在类中声明要组合类的 get、set 方法。而通过 Spring 框架中注解的运用也就主要是解决这类问题的。

关于 Spring MVC 开发相关的，除了前面介绍的 Mybatis 外，还有 JSTL、JSON、RESTful。下面分别简单介绍：

JSTL (JSP Standard Tag Library), JSP 标准标签库, 是一个实现 Web 应用程序中常见的通用功能的定制标记库集, 这些功能包括迭代和条件判断、数据管理格式化、XML 操作以及数据库访问。

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 ECMAScript 的一个子集, 采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。易于人阅读和编写, 同时也易于机器解析和生成, 并有效地提升网络传输效率。

RESTful 是一种互联网软件架构。网站即互联网软件, 采用客户端/服务器模式, 建立在分布式体系上, 通过互联网通信, 具有高延时、高并发等特点。网站开发, 完全可以采用软件开发的模式。它结构清晰、符合标准、易于理解、扩展方便, 所以正得到越来越多网站的采用。