轻量级 J2EE 框架应用

E 1　A Simple Controller

学号：SA18225170
姓名：李朝喜

报告撰写时间：2018/11/25

1. 主题概述

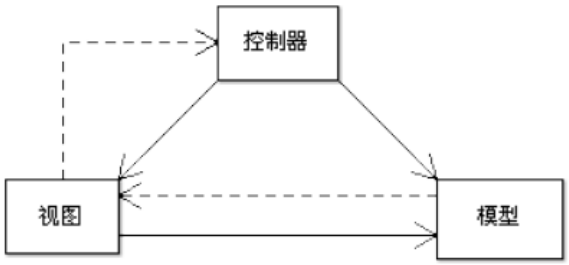
1.1 MVC 模式

MVC 模式（Model－View－Controller）最早由 Trygve Reenskaug 在 1978 年提出，是施乐帕罗奥多研究中心（Xerox PARC）在 20 世纪 80 年代为程序语言 Smalltalk 发明的一种软件架构模式，其把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。

- 控制器（Controller）：负责转发请求，对请求进行处理；
- 视图（View）：界面设计人员进行图形界面设计；
- 模型（Model）：程序员编写程序应有的功能（实现算法等等）、数据库专家进行数据管理和数据库设计（可以实现具体的功能）；

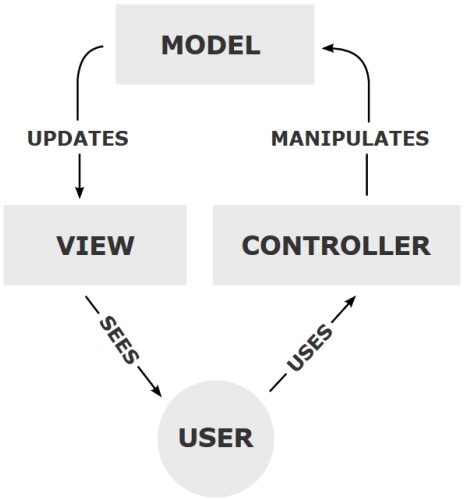
MVC 模式的目的是实现一种动态的程序设计，使后期对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。此外，该模式通过对复杂度的简化，使程序结构更加直观。软件系统通过对自身基本部分分离的同时也赋予了各个基本部分应有的功能，使专业人员可以根据自身的专长分组。

MVC 模式示图如下：



MVC 模式图

（图源：https://commons.wikimedia.org/w/index.php?curid=1281188）



MVC 组件间合作示例图

（图源：https://commons.wikimedia.org/w/index.php?curid=10298177）

1.2 控制器（Controller）

1.2.1 作用

控制器（Controller）：负责转发请求，对请求进行处理。

1.2.2 实现

- 定义 SimpleController 类，继承 javax.servlet.http.HttpServle 类；
- 根据实际业务重写 doGet(req, resp)、doPost(req, resp)方法；
- 从 req 中可获得实际请求内容：
  - SimpleController 可将请求转发给其他 Controller；
  - SimpleController 可将请求任务分配给 service 对象（模型层）处理；
  - SimpleController 也可以在本类中处理简单业务；
- 将处理后的结果传给 resp，以完成此次交互；

2. 假设

2.1 知识背景

在进行实验前，请确保具备以下知识背景：

- 具备基本的 HTML 编程能力，如：会写简单的 HTML 页面；
- 理解 Java 基础知识，并且具备基本的 Java 编程能力，如：理解 Java 中对象封装、继承等概念，以及具备 JavaSE 基本开发能力；
- 对相关术语有概念，知道它们的用途，如：Eclipse、Tomcat、Maven、Servlet。

2.2 环境背景

本次实验是在 Windows 10 系统下进行的，部分操作可能对其他系统并不适用。此外，请确保你的电脑至少有 2G 的内存，否则在同时运行多款软件时，可能会出现卡顿现象。

具体要求：

- 已安装并且配置好 Java 环境：Java SE Development Kit 11.0.1
- 已安装好 Eclipse：eclipse-jee-2018-09-win32-x86_64.zip
- 已安装好 Tomcat：https://tomcat.apache.org/download-90.cgi
- 已安装好 Maven：https://maven.apache.org/download.cgi

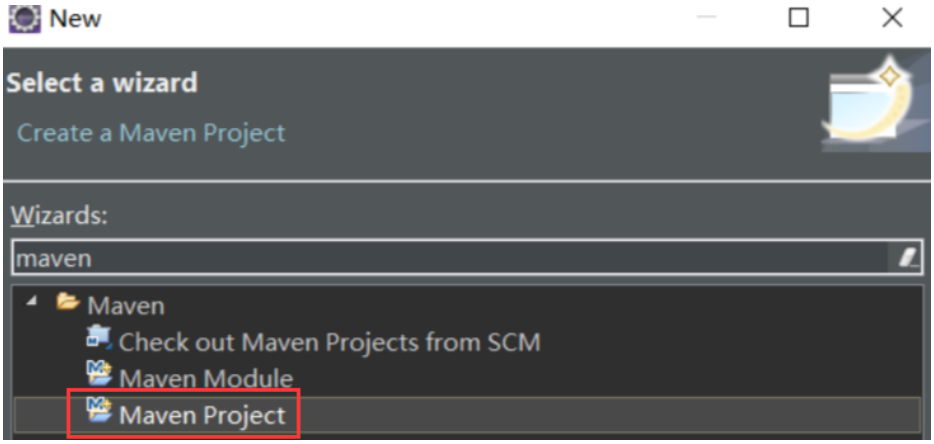
注意：软件安装以及相关配置不在本文档涉及范围内，对于如：Java 环境变量配置、Maven 修改镜像源、Eclipse 中配置 Maven 等问题，请读者自行查阅资料。

3. 实现或证明

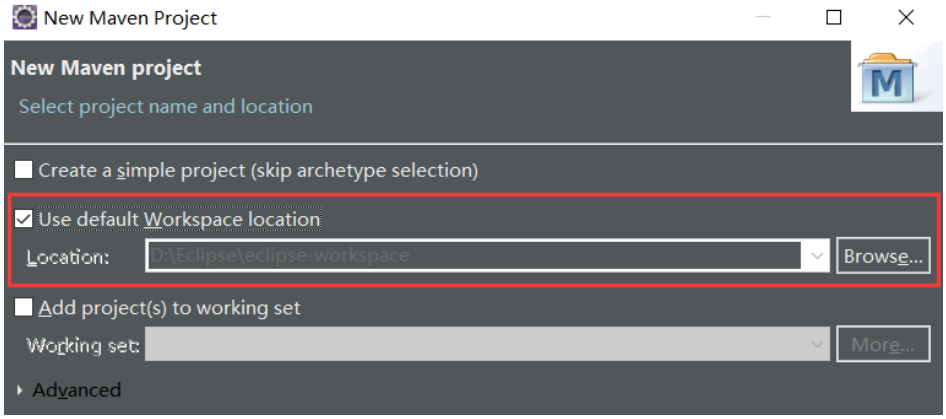
3.1 SimpleController 实现

3.1.1 项目创建

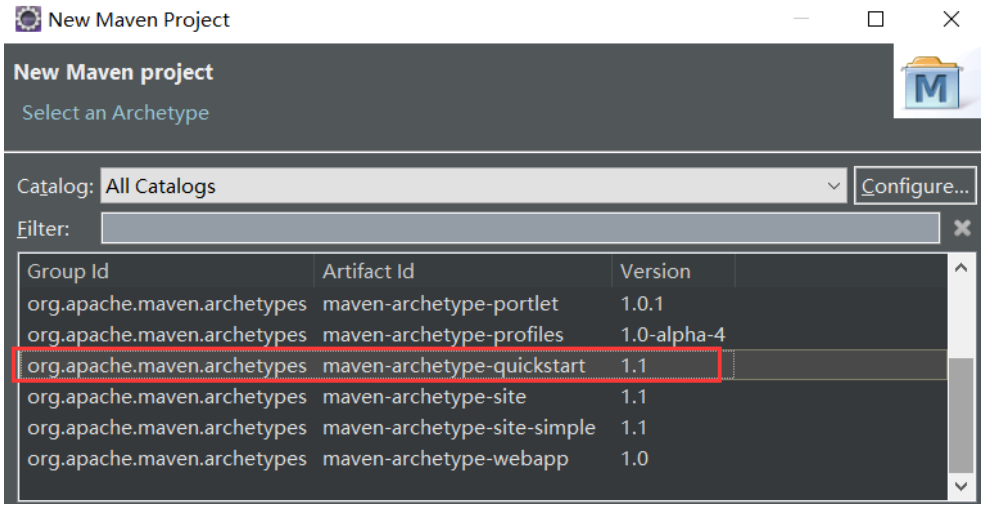
1）运行 Eclipse，左上角选择：File -> New -> Other，选择创建 Maven Prject



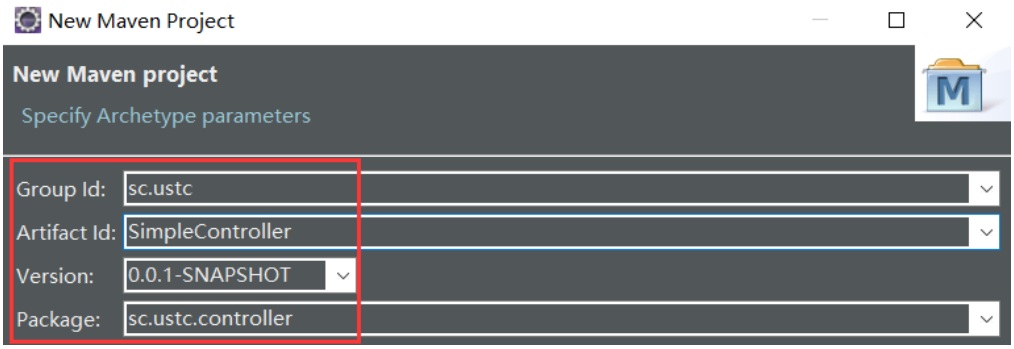
2）选择项目存放地址，如本例中为：D:\Eclipse\eclipse-workspace



3）选择：maven-archetype-quickstart 1.1



4）填写项目信息：Group Id、Artifact Id、Version、Package



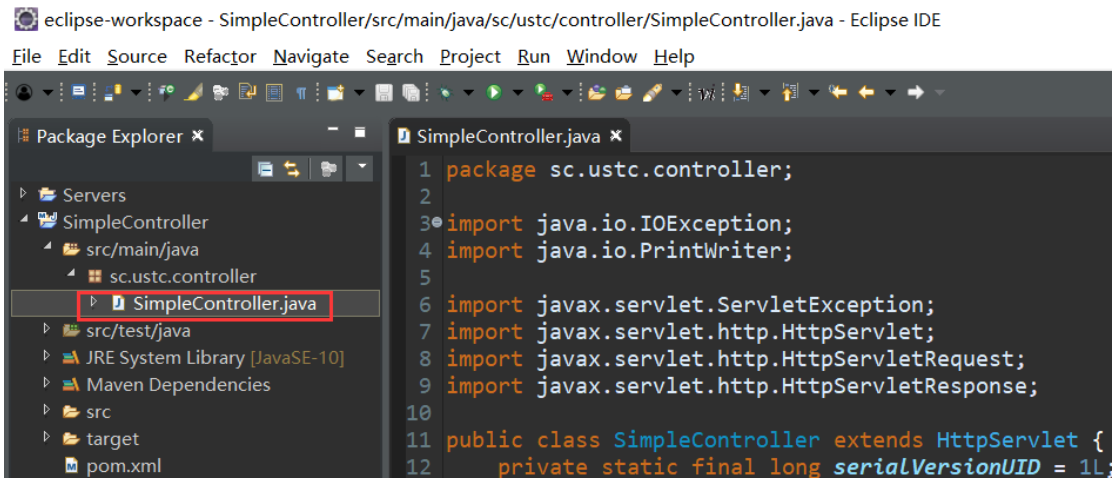
3.1.2 添加依赖包（javax.servlet，javax.servlet.jsp）

在 pom.xml 添加如下配置：

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>javax.servlet.jsp-api</artifactId>
  <version>2.3.1</version>
  <scope>provided</scope>
</dependency>
```

3.1.3 编写 Controller（SimpleController.java）

在 sc.ustc.controller 下创建 SimpleController.java 文件，编写如下代码：



```
package sc.ustc.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SimpleController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        doPost(req, resp);
    }

    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        resp.setCharacterEncoding("UTF-8");
        PrintWriter out = resp.getWriter();
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=utf-8\" />");
        out.println("<title>SimpleController</title>");
        out.println("</head>");
        out.println("<body>欢迎使用 SimpleController!</body>");
        out.println("</html>");
    }
}
```

3.1.4 Maven 发布 jar 包（simple-controller-0.0.1.jar）

1）修改 pom.xml 中 groupId、artifactId、version、packaging 等信息：

```
<groupId>sc.ustc</groupId>
<artifactId>simple-controller</artifactId>
<version>0.0.1</version>
<packaging>jar</packaging>
```

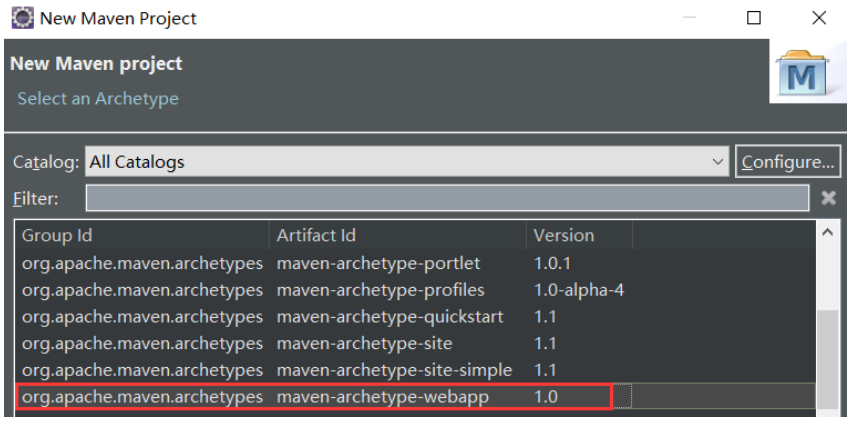
2）在 pom.xml 页面，右键 Run As -> Maven install，BUILD SUCCESS 后就可以在仓库中看到发布的 jar 包，如下图：

3）注：在执行 Maven install 时，可能会出现问题：No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?具体解决办法，参考第 5 部分（参考文献）中相关网址内容。

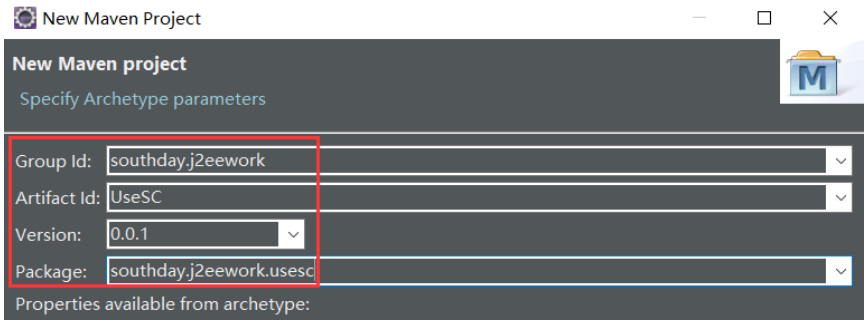
3.2 UseSC 实现

3.2.1 项目创建

1）同 3.1.1 中步骤类似，也是创建 Maven Project，区别是这次要选择：maven-archetype-webapp 1.0，如图：



2）填写项目信息：Group Id、Artifact Id、Version、Package

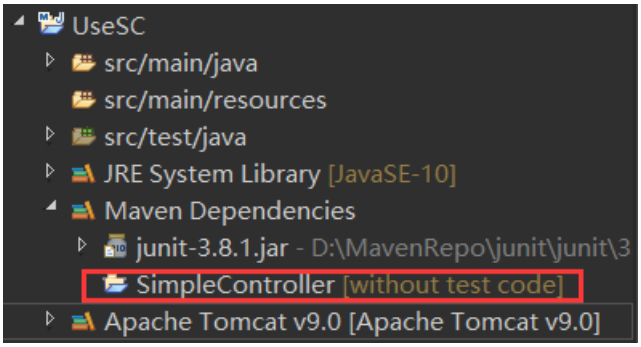


3.2.2 引入依赖包（simple-controller-0.0.1.jar）

1）在 pom.xml（UseSC 项目的 pom.xml）中添加配置：

```
<dependency>
  <groupId>sc.ustc</groupId>
  <artifactId>simple-controller</artifactId>
  <version>0.0.1</version>
</dependency>
```

2）引入后就可以在 Package Explorer 中看到，如图：



3.2.3 web.xml 中配置 servlet

在 src/main/webapp/WEB-INF/下打开 web.xml 文件，并在<web-app></web-app>标签之间添加如下配置：

```
<servlet>
   <servlet-name>sc</servlet-name>
   <servlet-class>sc.ustc.controller.SimpleController</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>sc</servlet-name>
   <url-pattern>/*</url-pattern>
</servlet-mapping>
```

3.2.4 设置 welcome.html 欢迎页面

1）在 src/main/webapp/目录下新建 welcome.html 文件，编写如下代码：

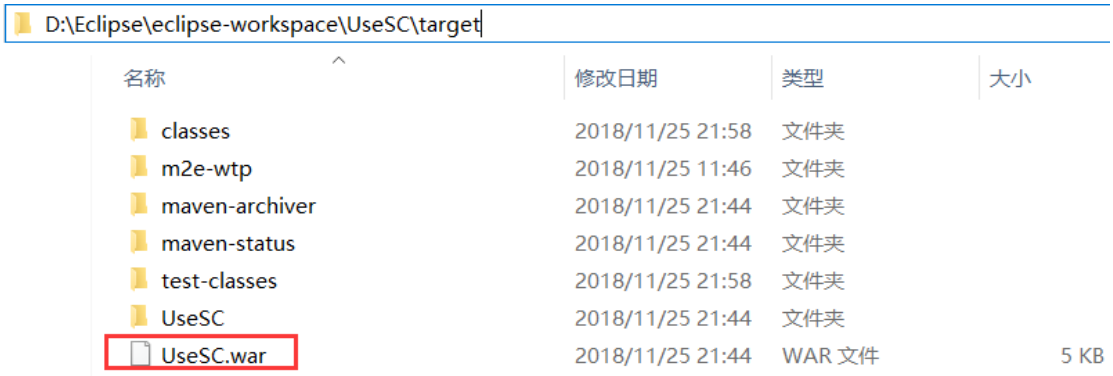
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>UseSC - southday</title>
</head>
<body>
    <p style="text-align: center;">
        <font size="5">Welcome to UseSC!</font>
    </p>
</body>
</html>
```

2）在 web.xml 中配置默认欢迎页面，在<web-app></web-app>标签之间添加如下配置：

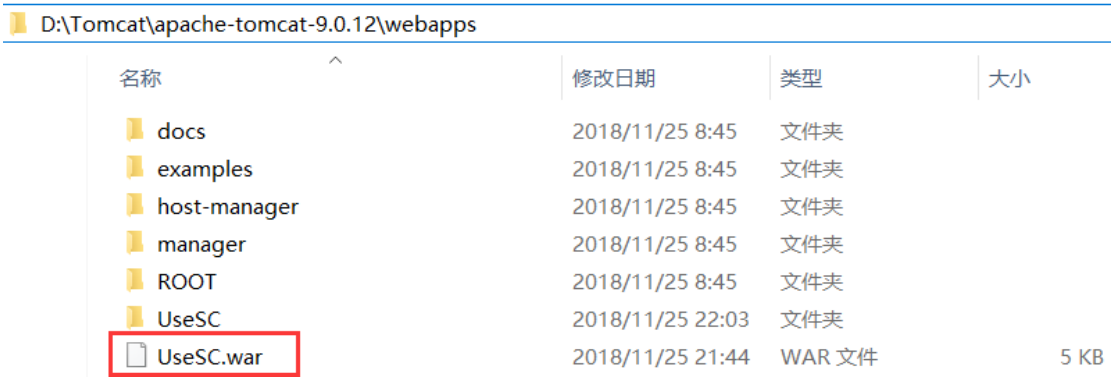
```
<welcome-file-list>
    <welcome-file>welcome.html</welcome-file>
</welcome-file-list>
```

3.3 Controller 测试

1）选中 pom.xml 页面，右键 Run As -> Maven Install，BUILD SUCCESS 后就可在 target 目录下看到发布的 war 包，如图：



2）将 UseSC.war 包拷贝到 Tomcat 安装目录下的 webapp 目录中，如下：

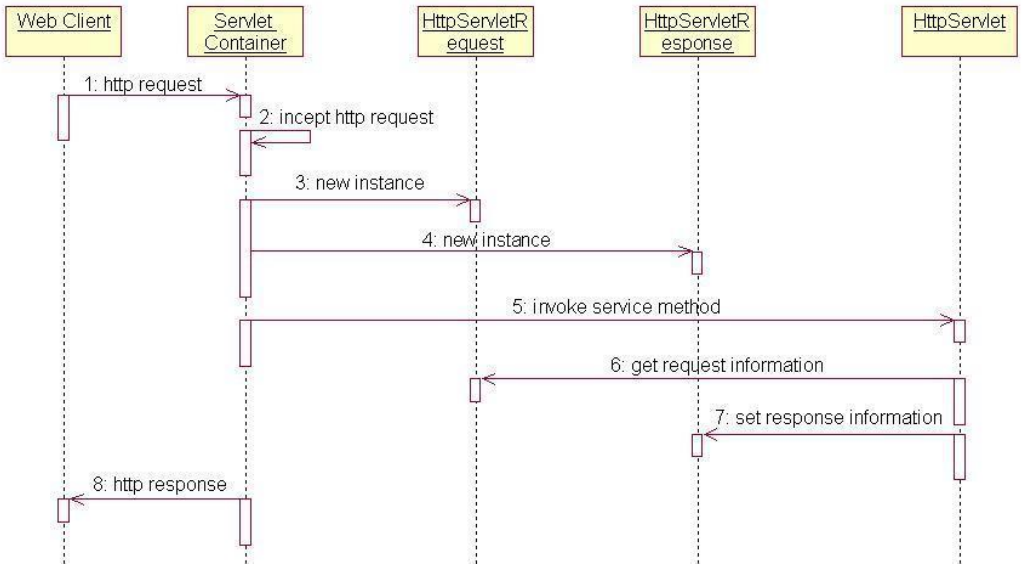


3）运行程序：D:\Tomcat\apache-tomcat-9.0.12\bin\startup.sh 即可启动 tomcat9 服务，启动服务后，在浏览器中输入：http://localhost:8080/UseSC/，看到如下页面：

4）运行程序：D:\Tomcat\apache-tomcat-9.0.12\bin\shutdown.sh 即可停止服务。

3.4 Http request 在 Web Container 中的处理流程

- Web Client 向 Servlet 容器（Tomcat）发出 Http 请求；
- Servlet 容器接收 Web Client 的请求；
- Servlet 容器创建一个 HttpRequest 对象，将 Web Client 请求的信息封装到这个对象中；
- Servlet 容器创建一个 HttpResponse 对象；
- Servlet 容器调用 HttpServlet 对象的 service 方法，把 HttpRequest 对象与 HttpResponse 对象作为参数传给 HttpServlet（Controller）对象；
- Controller 调用 HttpRequest 对象的相关方法，获取 Http 请求信息；
- Controller 将请求转发给其他 Controller，或者分配给 service 对象处理；
- Controller 调用 HttpResponse 对象的相关方法，包装处理结果，生成响应数据；
- Servlet 容器把 HttpServlet（Controller）的响应结果传给 Web Client；



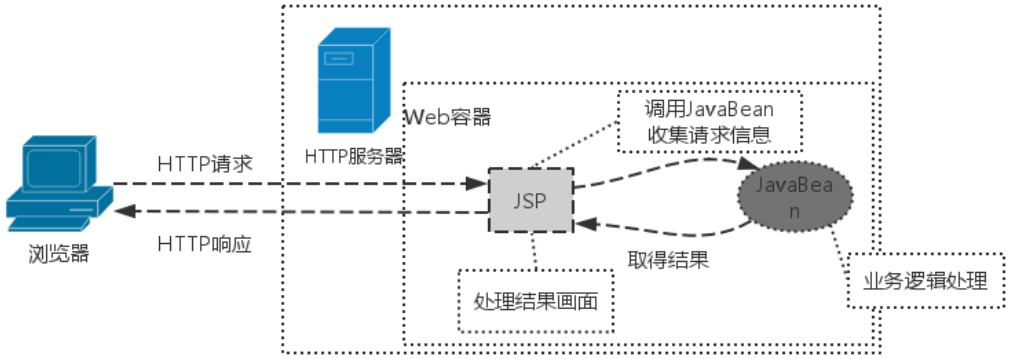
Servlet 工作原理时序图

（图源：https://upload-images.jianshu.io/upload_images/2062729-14290e2fa96e4d4a.jpg?imageMogr2/auto-orient/strip%7CimageView2/2/w/894）

3.5 JSP Model 1 与 Model 2 架构的优缺点及适用场景

3.5.1 JSP Model 1

在 Model 1 架构上，用户直接请求某个 JSP 页面，JSP 获取请求参数并调用 JavaBean 来处理请求。业务逻辑的部分封装在 JavaBean 中，JavaBean 处理完成请求后，JSP 会从 JavaBean 中提取结果，进行画面的呈现处理。



JSP Model 1 架构图

（图源：https://img-blog.csdn.net/20160809164151223）

优点：模型层次相较于 Model 2 要少（少了 Controller），容易上手，适合快速开发；

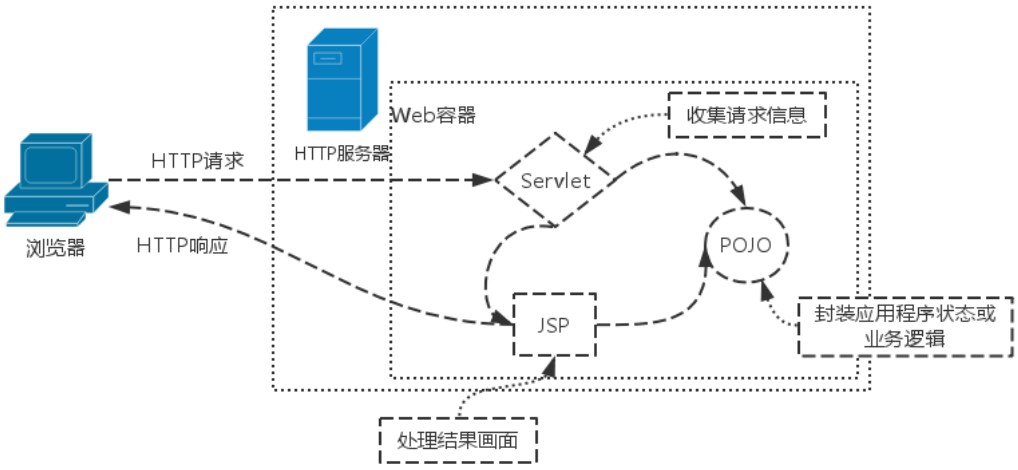
缺点：

- 模块间紧耦合，难以划分职责、合作开发；
- 页面渲染与业务处理的代码相互交织，增加了开发与维护难度；
- 当请求过多，业务处理逻辑较为复杂时，JSP 负载增加，可能不堪重负；

适用场景：适用于一些小型的、需要快速迭代开发的项目，并且项目中只有很少的、简单的业务逻辑处理需求。

3.5.2 JSP Model 2

在 Model 2 架构中，请求处理、业务逻辑处理以及画面呈现被区分为三个不同的角色职责，在分工合作开发时，使用 Model 2 架构可以理清职责界限，网页设计者与 Java 代码或请求处理以及业务模型原件的设计之间不用相互考虑太多，分工较为明确。



JSP Model 2 架构图

（图源：https://img-blog.csdn.net/20160809161434045）

优点：

- 模块间松耦合，职责划分明确，适合多人合作开发，且相互间干扰很少；
- 由于模块间松耦合，所以系统具有良好的可扩展性和灵活性；
- 使用 Controller 来处理请求，减轻了 JSP 的压力；

缺点：

- 系统模块层次较多，入门门槛相对于 Model 1 要高一些；
- 考虑到各模块间交互，会使程序设计成本和代码调试成本增加；

适用场景：适用于大规模系统的开发，包括具备较多复杂业务的系统。这类系统通常有良好的设计与软件开发流程，可以协同多人进行开发，并且相互间干扰较少。

4. 结论

之前做过 Java Web 开发，但基本都是用框架，并没有去了解里面的机制。通过此次实验，我实现了一个简易的 Controller，并且自建 Web 项目使用了该 Controller，这让我对 MVC 模式中的控制器有了更深的理解。

实验过程中也遇到了很多问题，如：Maven install 报错、HTML 中文乱码、Eclipse 配置 Maven 和 Tomcat 等，这些问题都可以通过百度、谷歌搜索快速获得解决方案。但更重要的问题是对技术知识本质理解的不足，如：何为 Servlet？何为 Web Container？等等，这些知识是需要静下心来去思考、去摸索、去实践，才能有更深的体会，而不能急于求成。

此外，实验中的 SimpleController 类的 doPost()方法，使用硬编码来返回欢迎页面（即：控制器 C 同时也承担了视图 V 的职责），这显然不是一种好的设计。因为一旦修改需求，比如：在欢迎页面中添加项目名称，就需要修改 SimpleController 中的代码，然后打包、发布、部署到服务器上。而如果 SimpleController 类中又有其他处理逻辑，就容易让修改变得混乱，代码难以维护。更好的设计是使用 MVC 进行分层，基于配置文件（xml）、注解来进行开发，而这些内容将会在接下来的实验中得以体现，拭目以待吧。

5. 参考文献

[1] MVC 维基百科：https://zh.wikipedia.org/wiki/MVC
[2] No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK? ：https://stackoverflow.com/questions/19655184/no-compiler-is-provided-in-this-environment-perhaps-you-are-running-on-a-jre-ra
[3] 如何在 Eclipse 配置 Tomcat 服务器：https://jingyan.baidu.com/article/3065b3b6efa9d7becff8a4c6.html
[4] web 开发中 web 容器的作用（如 tomcat）：https://www.jianshu.com/p/99f34a91aefe
[5] WEB 请求处理三：Servlet 容器请求处理：https://www.jianshu.com/p/571c474279af
[6] 谈谈 Tomcat 请求处理流程：http://www.importnew.com/27729.html
[7] JavaWeb 中的 Model 1 与 MVC/Model 2 架构比较：https://blog.csdn.net/zhuxinquan61/article/details/52163070
[8] About the Model 2 Versus Model 1 Architecture：https://download.oracle.com/otn_hosted_doc/jdeveloper/1012/developing_mvc_applications/adf_aboutmvc2.html