《JavaEE 实验手册》

Java 教研室

版 本 1.3

文 档 提 供：Java 教 研 室　孙 丽 萍

修 改 记 录

修改时间	修改人	修改内容
2012.8.15	王伟	文档创建
2016.3.7	孙丽萍	修改

目录

Web 程序结构与部署

一、内容概述

本章的教学内容是了解 JavaWeb 应用程序的目录组成以及各目录的作用，Web 应用的结构以及部署 Web 应用需要掌握的技术。

Tomcat 的应用程序结构：

```
Tomcat 安装目录
  ├── bin
  ├── conf
  ├── ...
  ├── webapps
      ├── appl
      ├── mysite
          ├── *.html, *.gif, *.js, *.css
          ├── *.jsp
          ├── WEB-INF
              ├── classes
              │   ├── onest
              │       ├── dev
              │           ├── MyServlet.class
              ├── lib
              │   ├── *.jar
              ├── web.xml
```

二、实验一 手工编写 helloworld

2.1 实验目的

手工编写与配置 Servlet 程序，熟悉 Web 应用的组成结构。

2.2 准备

在进行开发之前，要保证开发工具的正确安装与配置。

2.3 实验步骤

步骤一：

编写 Servlet 源程序

打开文本编辑工具(如 Notepad)，输入以下代码

```
package onest.dev;

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

import javax.servlet.Servlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class HelloServlet implements Servlet {

    public void destroy() {}
    public ServletConfig getServletConfig() {
        return null;
    }
    public String getServletInfo() {
        return null;
    }
    public void init(ServletConfig arg0) throws ServletException {}
    public void service(ServletRequest request, ServletResponse
response) throws ServletException, IOException {
        PrintWriter writer = response.getWriter();
        writer.println("Hello,Welcome goto your first Servlet");
    }
}
```

保存该文本文件到 D 盘根目录下的 HelloServlet.java 文件。

步骤二：

编译 Servlet 代码，生成.class 字节码文件

因为编写的是 Servlet 程序，用到了 Servlet 的 Jar 包(Tomcat 提供)，所以在编译时要指明 classpath，如，在命窗口输入以下内容：

```
javac -cp D:\opt\apache-tomcat-6.0.18\lib\servlet-api.jar HelloServlet.java
```

生成.class 文件，如*图 2-1*

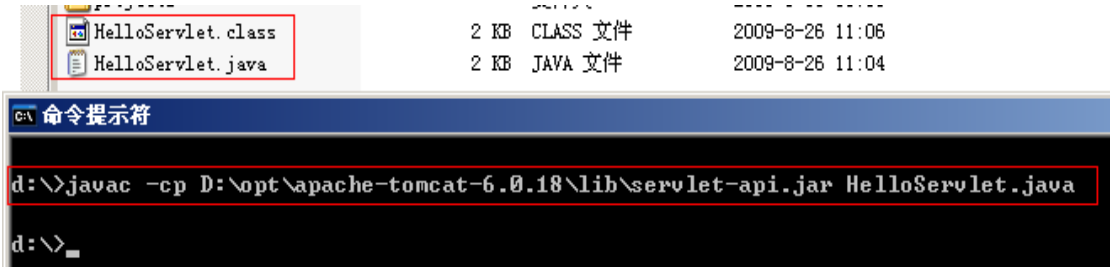


图 2-1

步骤三：

创建 JavaWeb 应用的目录

在“%Tomcat 安装目录%\wabapps\”目录下创建站点文件夹“first”。并且需要在 first 站点文件夹中创建标准 JavaWeb 站点目录“WEB-INF\classes\”，在该目录下根据编写的 HelloServlet 类的包名创建文件夹“onest\dev\”，并将 HelloServlet.class 复制到该文件夹中。如*图 2-2*

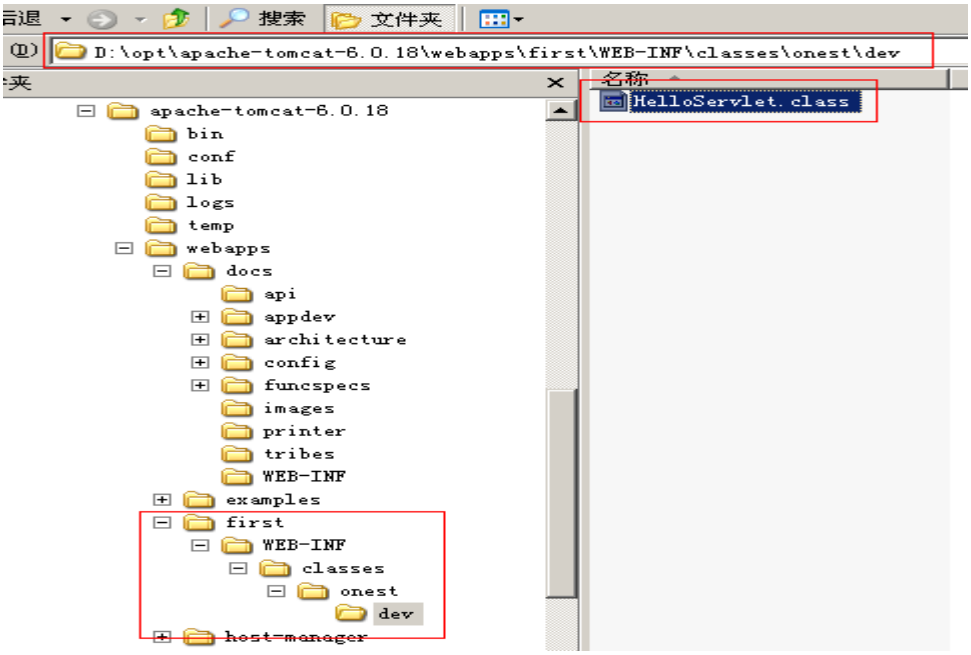


图 2-2

步骤四：

配置 Servlet 的访问 URL

在“%Tomcat 安装目录%\wabapps\ first\WEB-INF\”目录下新建文本文件，输入以下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
        http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>onest.dev.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

将该文件保存为 web.xml。

步骤五：

启动 Tomcat，用浏览器测试 Servlet

在浏览器中输入“http://localhost:8080/first/hello”，如*图 2-3*

图 2-3

2.4 实验结论

通过试验，熟悉 JavaWeb 项目目录结构及开发 Servlet 的过程。

三、实验二 获取请求

3.1 实验目的

针对请求(Request)的实验。

3.2 准备

保证开发工具的正确安装与配置。如果 Tomcat 正在运行，需要停止它。

3.3 实验步骤

步骤一：

，新建项目“15_request”，使用 Eclipse 创建 Servlet 类，类名称为 RequestDemo，如*图 3-1* 所示

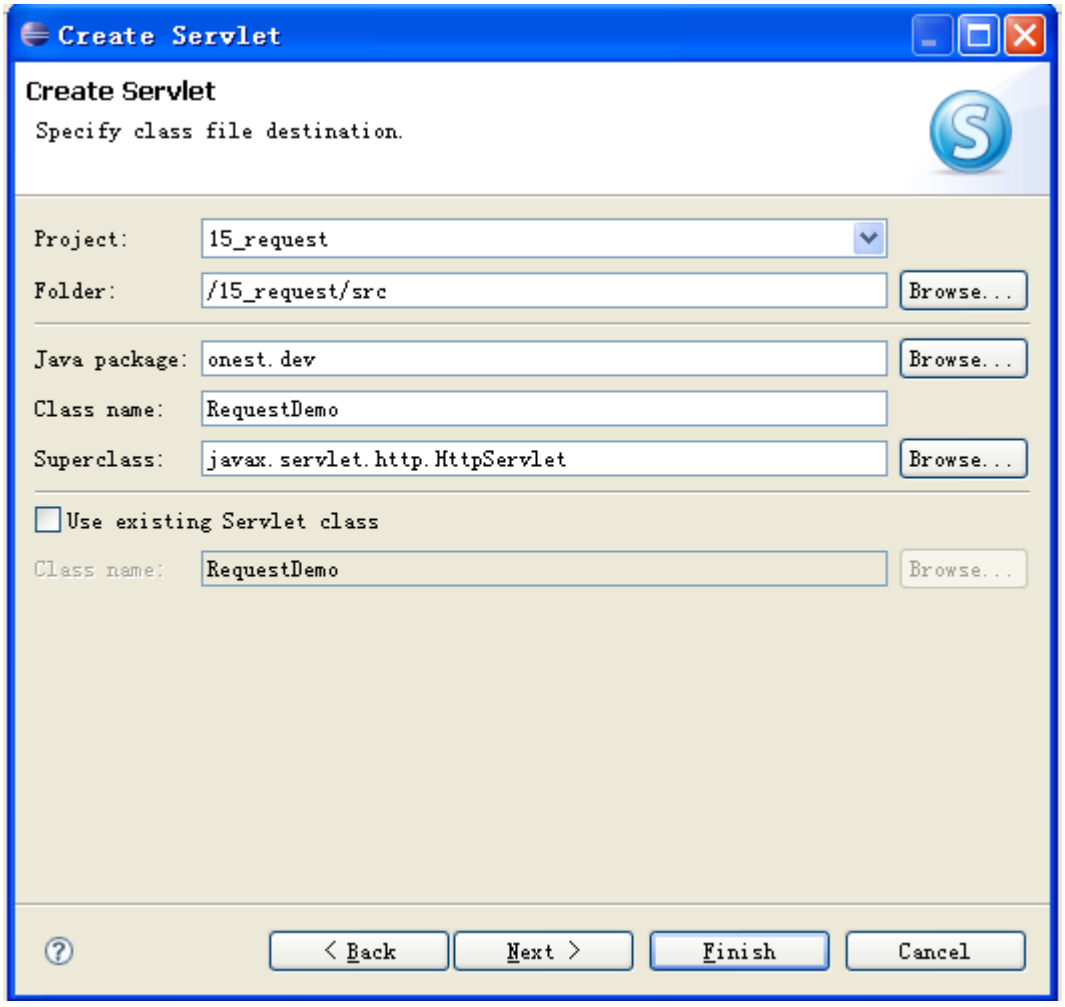


图 3-1

步骤二：

将 index.html 修改成如下内容，注意链接与 action 的指向地址：

```
<a href="RequestDemo">
  goto RequestDemo Servlet with GET
</a>
<form action=" RequestDemo" method="get">
  please input your nick:<input type="text" name="nick" />
  submit a GET request.<input type="submit" />
</form>
<form action=" RequestDemo" method="post">
  please input your nick:<input type="text" name="nick"_/>
  submit a POST request.<input type="submit" />
</form>
```

步骤三：

修改 Servlet 类 RequestDemo 的 doGet 和 doPost 方法如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    String nick = request.getParameter("nick");
    System.out.println("You send a GET request");
    System.out.println("Your nick is : " + nick);

}

protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    String nick = request.getParameter("nick");
    System.out.println("You send a POST request");
    System.out.println("Your nick is : " + nick);
}
```

注意 request.getParameter 方法的使用。

步骤四：

使用 Eclipse 启动 Tomcat，用浏览器访问地址 http://localhost:8080/15_request ，显示如*图 3-2*

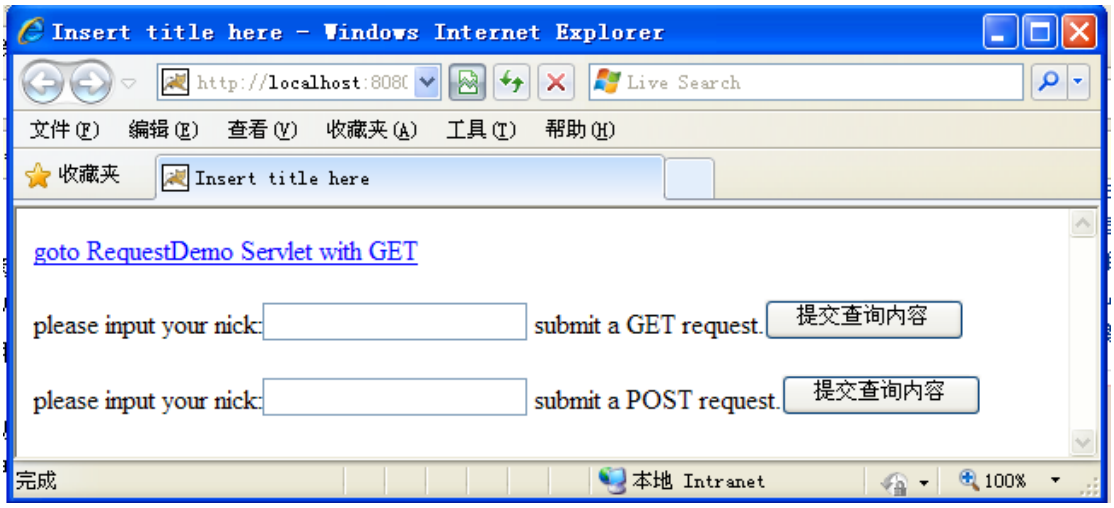


图 3-2

步骤五：

分别点击链接、GET 提交按钮与 POST 提交按钮，注意观察各种请求方式下浏览器地址栏的变化以及 Eclipse 控制台输出内容的不同，如*图 3-3*，*图 3-4*，*图 3-5 所示*

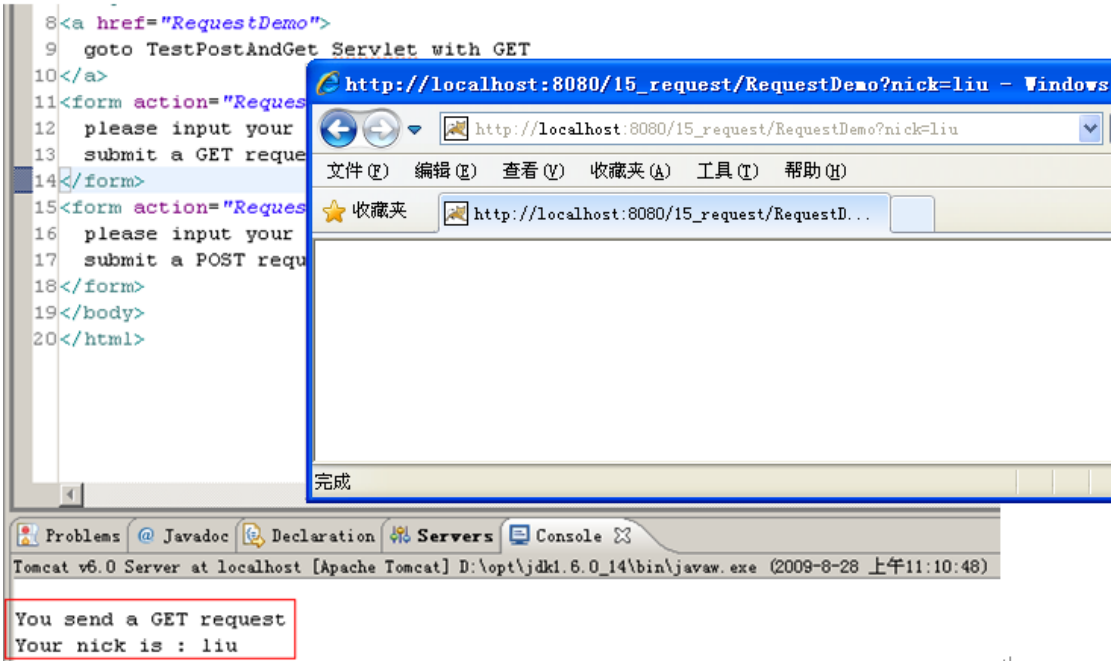


图 3-3

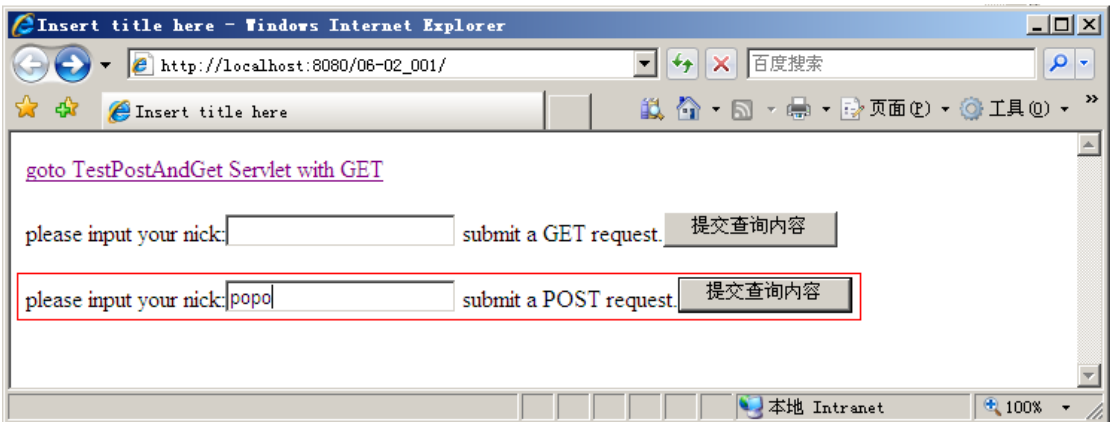


图 3-4

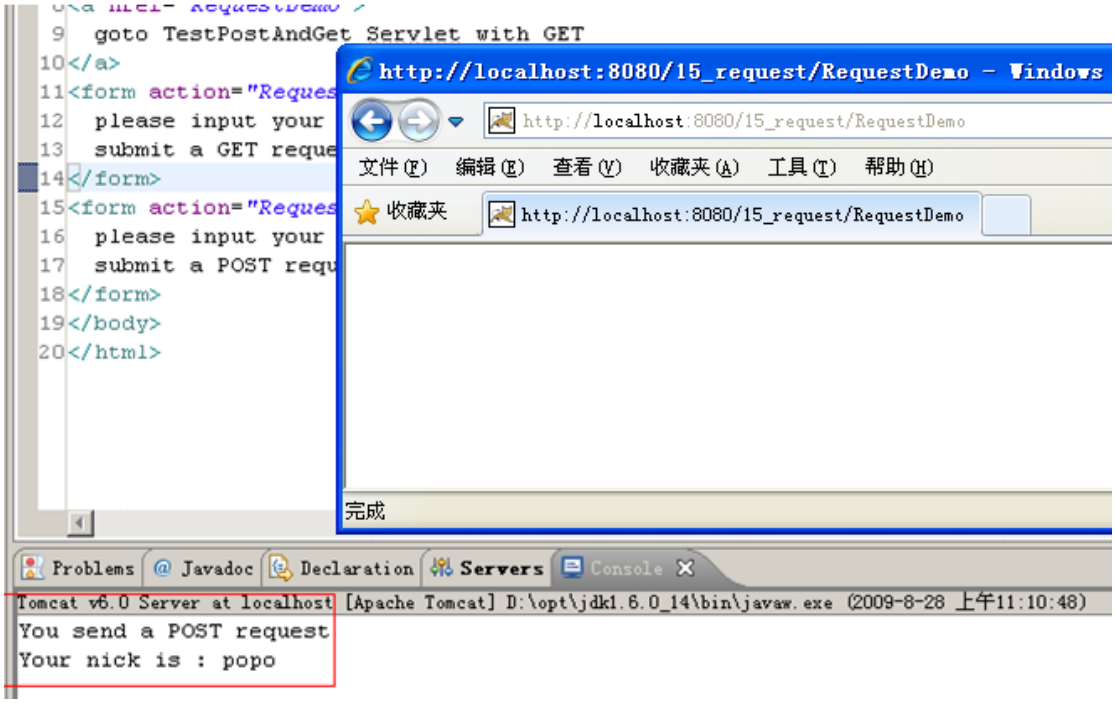


图 3-5

与 Request 相关的实验完成。

3.4 实验结论

通过该实验，进一步理解 Request 的 GET 与 POST 两种请求方式的不同。掌握由客户端浏览器向服务器端程序传递数据是通过 request.getParameter()方法取得客户端 URL 参数或表单域中的值。

四、实验三 response

4.1 实验目的

针对响应(Response)的实验。

4.2 准备

保证开发工具的正确安装与配置。如果 Tomcat 正在运行，需要停止它。

4.3 实验步骤

步骤一：

新建项目“15_response”，使用 Eclipse 创建 Servlet 类，类名称为

ResponseDemo，如*图 4-1*

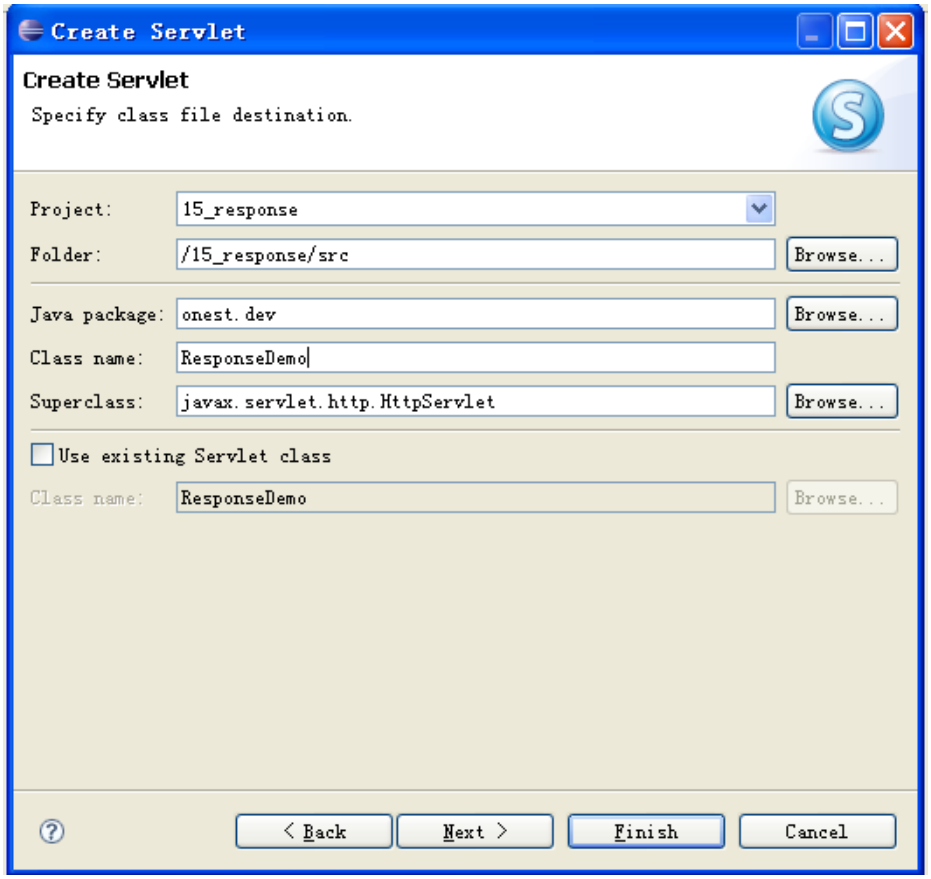


图 4-1

步骤二：

修改 ResponseDemo 的 doGet 和 doPost 方法如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    PrintWriter writer = response.getWriter();
    writer.println("You send a GET request");

}

protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    PrintWriter writer = response.getWriter();
    writer.println("You send a POST request");
}
```

注意 response.getWriter 方法，以及 writer.println 和 writer.writer 等输出方法。

步骤三：

修改 index.html 中链接与 action 地址为 ResponseDemo

```
<a href="ResponseDemo">
  goto TestPostAndGet Servlet with GET
</a>
<form action="ResponseDemo" method="get">
  please input your nick:<input type="text" name="nick" />
  submit a GET request.<input type="submit" />
</form>
<form action="ResponseDemo" method="post">
  please input your nick:<input type="text" name="nick" />
  submit a POST request.<input type="submit" />
</form>
```

步骤四：

运行调试该项目，观察浏览器的地址栏以及浏览器内容的变化，如*图 4-2*，*图 4-3* 所示

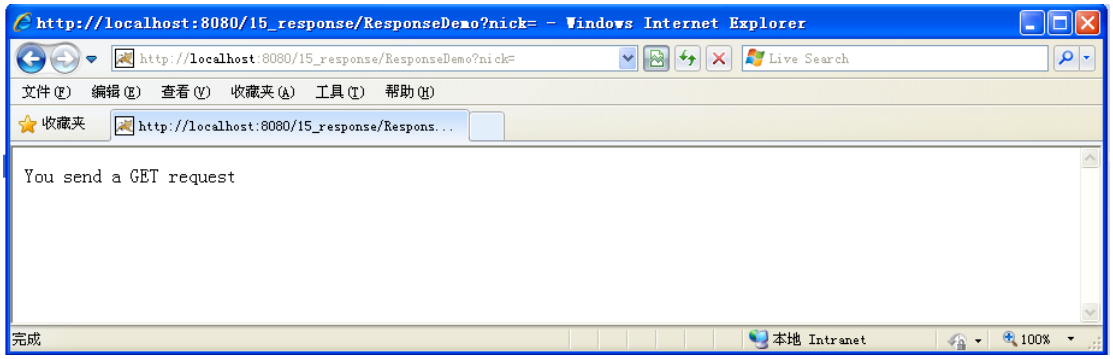


图 4-2

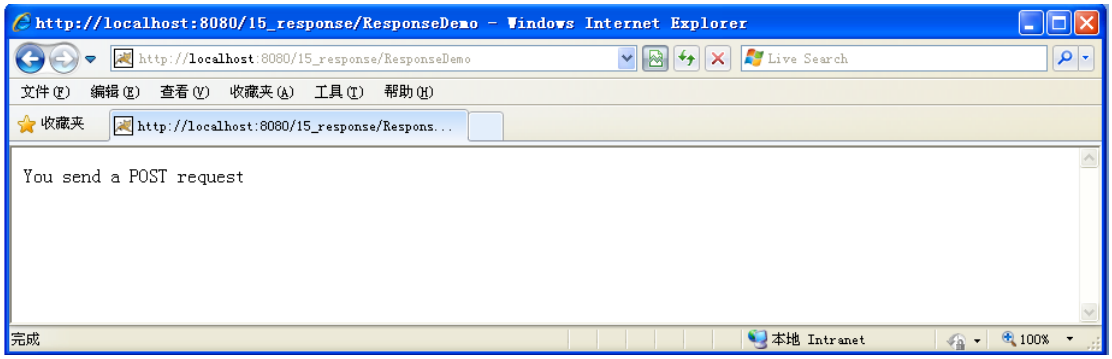


图 4-3

4.4 实验结论

通过实验掌握 Response 的使用，需要注意输出内容是输出到流里，与前面课程中 I/O 部分的知识结合理解。

4.5 扩展

使用 response.getWriter()方法会获得一个输出流(PrintWriter)，PrintWriter 有 print()、println()和 write()等方法，分别使用这些方法向页面输出内容，观察客户端得到的源码，区分这些方法的不同。

五、实验四 2 种跳转的区别

5.1 实验目的

针对页面跳转、请求转发与请求数据的传递

5.2 准备

保证开发工具的正确安装与配置。如果 Tomcat 正在运行，需要停止它。

5.3 实验步骤

步骤一：

使用 Eclipse 分别创建三个 Servlet 类：

- RedirectDemo：展示页面跳转的功能
- ForwardDemo：展示请求转发功能
- TargetServlet：页面跳转与请求转发的目标地址

步骤二：

RediredtDemo 的关键代码如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    this.doPost(request, response);
}

protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    request.setAttribute("request.Attribute", "liuzhanhong");
    response.sendRedirect("TargetServlet");
}
```

步骤三：

ForwardDemo 关键代码如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    doPost(request, response);
}

protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    request.setAttribute("request.Attribute", "liuzhanhong");
    request.getRequestDispatcher("TargetServlet").forward(request,
response);
}
```

步骤四：

TargetServlet 关键代码如下：

```
protected void doGet(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    this.doPost(request, response);
}

protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    Object attribute = request.getAttribute("request.Attribute");
    PrintWriter writer = response.getWriter();
    writer.println("hi,Welcome<br />");
    writer.println("request.Attribute is : " + attribute);
}
```

步骤五：

运行该项目，并测试 ForwarDemo，如*图 5-1* 所示

图 5-1

观察地址栏与输出页面的结果。如*图 5-2*

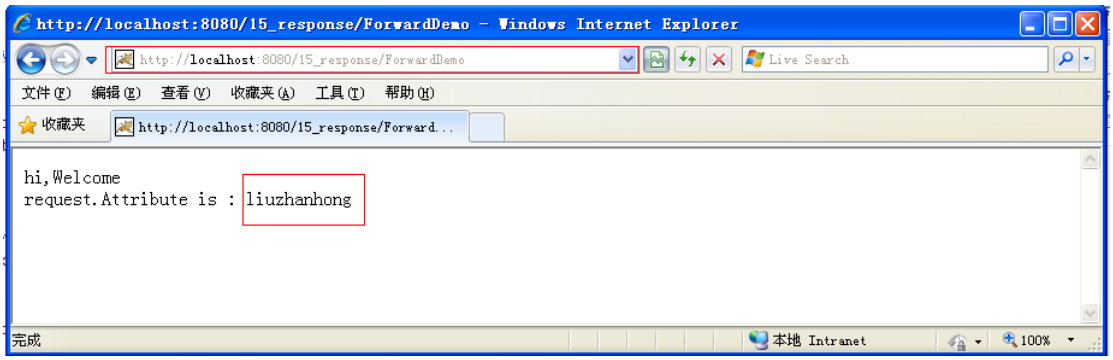


图 5-2

步骤六：

测试 RedirectDemo，如*图 5-3*

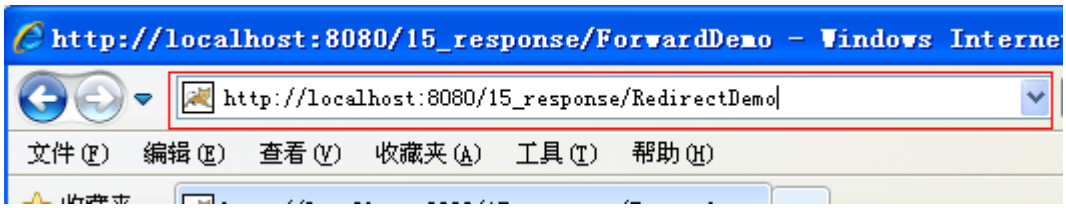


图 5-3

观察地址栏与输出页面的结果。如*图 5-4*

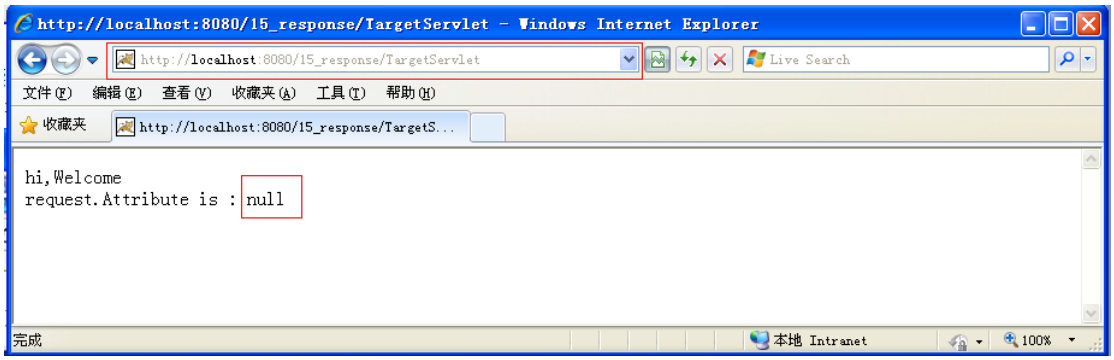


图 5-4

5.4 实验结论

页面跳转(SendRedirect)无法携带 Attribute 属性，但请求转发可以，但将请求进行转发时地址栏的 URL 是没有变化的。

六、实验五 登录

6.1 实验目的

使用 Servlet 技术，实现简单的页面登录的功能。

6.2 准备

功能需求：

功能一

客户端访问站点首页面，在首页面可以让用户输入用户名和密码，并通过提交按钮进行登录操作。

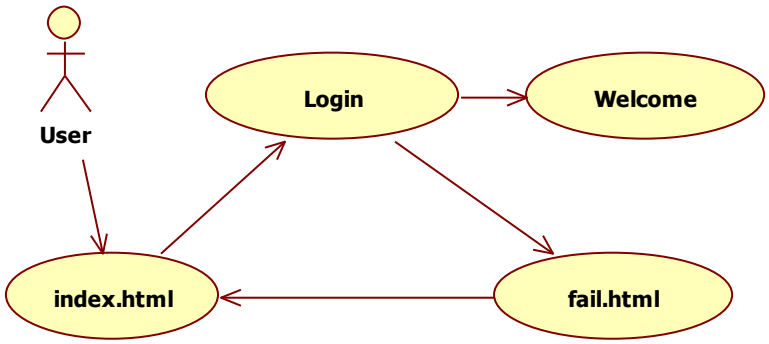
功能二

检查用户输入的密码(对用户名不检查)，如果密码为 123，表示登录成功，转向欢迎页面，且欢迎页面显示用户输入的名称。

功能三

如果用户输入的用户名不密码不与上一功能需求相符，页面跳转到失败页面，失败页面提供转向首页面的链接

用例图：



概要设计：

设计一

需要有一个静态页面(index.html)作为首页面，并提供两个文本输入框和一个提交按钮。

设计二

需要有一个 Servlet(LoginServlet)来实现进行登录时的验证功能。

设计三

在 LoginServlet 中，如果验证通过，取得客户输入的用户名并将请求转发给欢迎页面。

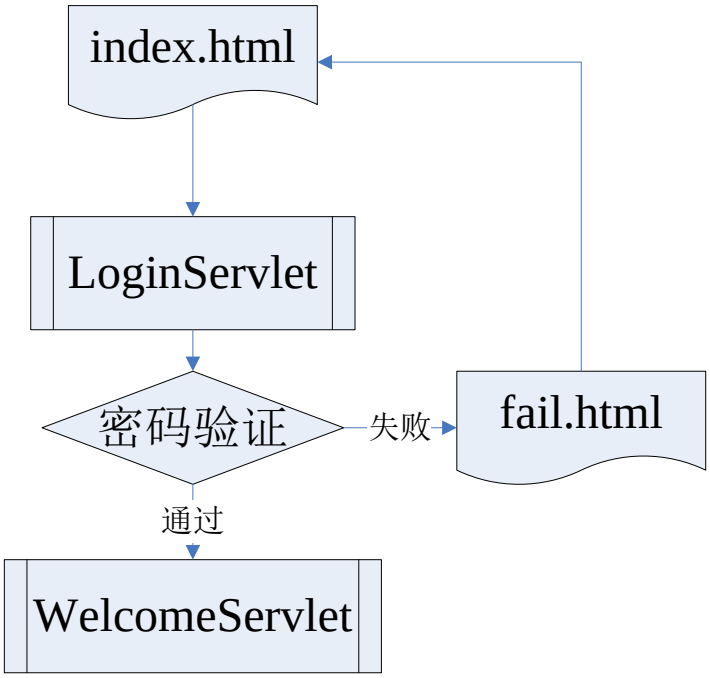
设计四

欢迎页面要展示动态信息(用户名)，所以也为一个 Servlet(WelcomeServlet),用来向客户端展示欢迎信息及用户名。

设计五

在 LoginServlet 中，如果验证失败，转向失败页面(静态页面 fail.html)，该页面有一个返回首页面的链接

流程图：



6.3 编码实现

步骤一：

使用 Eclipse 创建动态网站项目，并起名称。

[由学生完成]

步骤二：

创建静态页面 index.html。

[由学生完成]

步骤三：

创建 Servlet 类：LoginServlet，并实现验证的逻辑代码。

[由学生完成]

步骤四：

创建 Servlet 类：WelcomeServlet，并实现显示用户名的功能。

[由学生完成]

步骤五：

创建静态页面 fail.html。

[由学生完成]

6.4 实验结论

通过此实验，让同学掌握 Servlet 开发动态网站的相关技术与实现过程，了解开发动态网站的步骤，并进一步熟悉 Eclipse 工具的使用。