

request 请求对象

今日内容介绍

- ◆ 案例一：完成用户注册的案例
- ◆ 案例二：完善用户登录，显示错误信息

今日内容学习目标

- ◆ 使用 request 获得表单请求参数
- ◆ 使用 request 完成请求转发，并成功在一次请求中传递数据

第1章 案例：完成用户注册的功能：

1.1 需求：

网站首页上点击注册的链接，跳转到注册页面，在注册页面中输入信息，完成注册（将数据保存到数据库中）。

会员注册 USER REGISTER

用户名

密码

确认密码

Email

姓名

性别 男 女

出生日期

验证码 LOCK

注册

1.2 相关知识点：

1.2.1 HttpServletRequest 对象

在 Servlet API 中，定义了一个 `HttpServletRequest` 接口，它继承自 `ServletRequest` 接口，专门用来封装 HTTP 请求消息。由于 HTTP 请求消息分为请求行、请求消息头和请求消息体三部分，因此，在 `HttpServletRequest` 接口中定义了获取请求行、请求头和请求消息体的相关方法，接下来，本节将针对这些方法进行详细地讲解。

1.2.1.1 获取请求行信息的相关方法

当访问 Servlet 时，会在请求消息的请求行中，包含请求方法、请求资源名、请求路径等信息，为了获取这些信息，在 `HttpServletRequest` 接口中，定义了一系列用于获取请求行的方法，如表 4-2 所示。

表1-1 获取请求行的相关方法

方法声明	功能描述
<code>String getMethod()</code>	该方法用于获取 HTTP 请求消息中的请求方式(如 GET、POST 等)
<code>String getRequestURI()</code>	该方法用于获取请求行中资源名称部分，即位于 URL 的主机和端口之后、参数部分之前的部分
<code>String getQueryString()</code>	该方法用于获取请求行中的参数部分，也就是资源路径后面问号 (?) 以后的所有内容
<code>String getProtocol()</code>	该方法用于获取请求行中的协议名和版本，例如，HTTP/1.0 或 HTTP/1.1
<code>String getContextPath()</code>	该方法用于获取请求 URL 中属于 WEB 应用程序的路径，这个路径以 “/” 开头，表示相对于整个 WEB 站点的根目录，路径结尾不含 “/”。如果请求 URL 属于 WEB 站点的根目录，那么返回结果为空字符串 (“”)
<code>String getServletPath()</code>	该方法用于获取 Servlet 的名称或 Servlet 所映射的路径
<code>String getRemoteAddr()</code>	该方法用于获取请求客户端的 IP 地址，其格式类似于 “192.168.0.3”
<code>String getRemoteHost()</code>	该方法用于获取请求客户端的完整主机名，其格式类似于 “pc1.itcast.cn”。需要注意的是，如果无法解析出客户机的完整主机名，该方法将会返回客户端的 IP 地址
<code>int getRemotePort()</code>	该方法用于获取请求客户端网络连接的端口号
<code>String getLocalAddr()</code>	该方法用于获取 Web 服务器上接收当前请求网络连接的 IP 地址
<code>String getLocalName()</code>	该方法用于获取 Web 服务器上接收当前网络连接 IP 所对应的主机名
<code>int getLocalPort()</code>	该方法用于获取 Web 服务器上接收当前网络连接的端口号
<code>String getServerName()</code>	该方法用于获取当前请求所指向的主机名，即 HTTP 请求消息

	中 Host 头字段所对应的主机名部分
int getServerPort()	该方法用于获取当前请求所连接的服务器端口号，即如果 HTTP 请求消息中 Host 头字段所对应的端口号部分
String getScheme()	该方法用于获取请求的协议名，例如 http、https 或 ftp
StringBuffer getRequestURL()	该方法用于获取客户端发出请求时的完整 URL，包括协议、服务器名、端口号、资源路径等信息，但不包括后面的查询参数部分。注意，getRequestURL()方法返回的结果是 StringBuffer 类型，而不是 String 类型，这样更便于对结果进行修改

在表 4-2 中，列出了一系列用于获取请求消息行信息的方法，为了使读者更好地理解这些方法，接下来，通过一个案例来演示这些方法的使用。

在 chapter04 项目的 src 目录下，新建一个名称为 cn.itcast.chapter04.request 的包，在包中编写一个名为 RequestLineServlet 的类，该类中编写了用于获取请求行中相关信息的方法，如文件 4-10 所示。

文件 4-1 RequestLineServlet.java

```

1 package cn.itcast.chapter04.request;
2 import java.io.*;
3 import javax.servlet.*;
4 import javax.servlet.http.*;
5 public class RequestLineServlet extends HttpServlet {
6     public void doGet(HttpServletRequest request,
7         HttpServletResponse response) throws ServletException, IOException {
8         response.setContentType("text/html;charset=utf-8");
9         PrintWriter out = response.getWriter();
10        // 获取请求行的相关信息
11        out.println("getMethod : " + request.getMethod() + "<br>");
12        out.println("getRequestURI : " + request.getRequestURI() + "<br>");
13        out.println("getQueryString:" + request.getQueryString() + "<br>");
14        out.println("getProtocol : " + request.getProtocol() + "<br>");
15        out.println("getContextPath:" + request.getContextPath() + "<br>");
16        out.println("getPathInfo : " + request.getPathInfo() + "<br>");
17        out.println("getPathTranslated : "
18            + request.getPathTranslated() + "<br>");
```

```

31     public void doPost(HttpServletRequest request,
32             HttpServletResponse response) throws ServletException, IOException {
33         doGet(request, response);
34     }
35 }

```

在 web.xml 中配置完 RequestLineServlet 的映射后，启动 Tomcat 服务器，在浏览器的地址栏中输入地址“<http://localhost:8080/chapter04/RequestLineServlet>”访问 RequestLineServlet，浏览器的显示结果如图 4-14 所示。

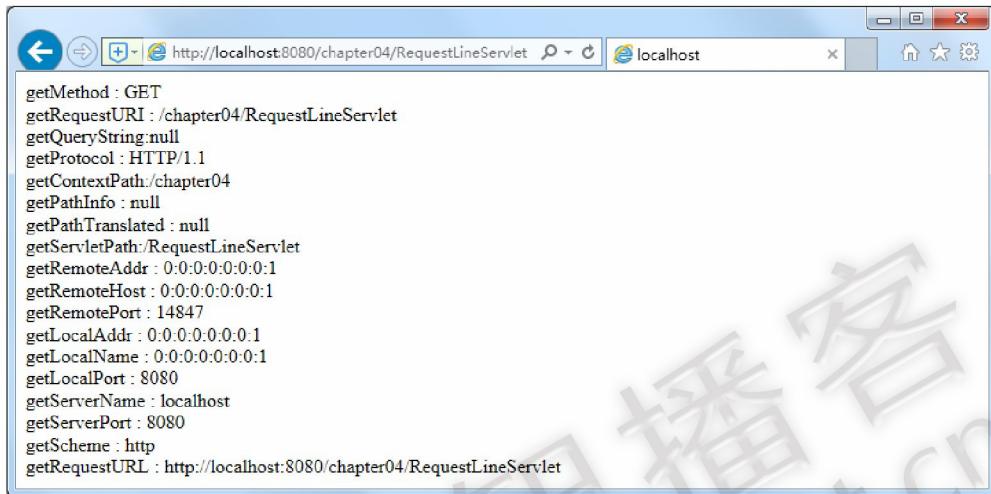


图3-1 运行结果

从图 4-14 中可以看出，浏览器显示出了请求 RequestLineServlet 时，发送的请求行信息。由此可见，通过 HttpServletRequest 对象可以很方便的获取到请求行的相关信息。

1.2.1.2 获取请求消息头的相关方法

当请求 Servlet 时，需要通过请求头向服务器传递附加信息，例如，客户端可以接收的数据类型，压缩方式，语言等等。为此，在 HttpServletRequest 接口中，定义了一系列用于获取 HTTP 请求头字段的方法，如表 4-3 所示。

表1-2 获取请求消息头的方法

方法声明	功能描述
String getHeader(String name)	该方法用于获取一个指定头字段的值，如果请求消息中没有包含指定的头字段，getHeader()方法返回 null；如果请求消息中包含有多个指定名称的头字段，getHeader()方法返回其中第一个头字段的值
Enumeration getHeaders(String name)	该方法返回一个 Enumeration 集合对象，该集合对象由请求消息中出现的某个指定名称的所有头字段值组成。在多数情况下，一个头字段名在请求消息中只出现一次，但有时候可能会出现多次
Enumeration getHeaderNames()	该方法用于获取一个包含所有请求头字段的 Enumeration 对象

int getIntHeader(String name)	该方法用于获取指定名称的头字段，并将其值转为 int 类型。需要注意的是，如果指定名称的头字段不存在，返回值为 -1；如果获取到的头字段的值不能转为 int 类型，将发生 NumberFormatException 异常
Long getDateHeader(String name)	该方法用于获取指定头字段的值，并将其按 GMT 时间格式转换成一个代表日期/时间的长整数，这个长整数是自 1970 年 1 月 1 日 0 点 0 分 0 秒算起的以毫秒为单位的时间值
String getContentType()	该方法用于获取 Content-Type 头字段的值，结果为 String 类型
int getContentLength()	该方法用于获取 Content-Length 头字段的值，结果为 int 类型
String getCharacterEncoding()	该方法用于返回请求消息的实体部分的字符集编码，通常是从 Content-Type 头字段中进行提取，结果为 String 类型

在表 4-3 中，列出了一系列用于读取 HTTP 请求消息头字段的方法，为了更好地掌握这些方法，接下来通过一个案例来学习这些方法的使用。

在 cn.itcast.chapter04.request 包中编写一个名为 RequestHeadersServlet 的类，该类使用 getHeaderNames() 方法来获取请求消息头信息，如文件 4-11 所示。

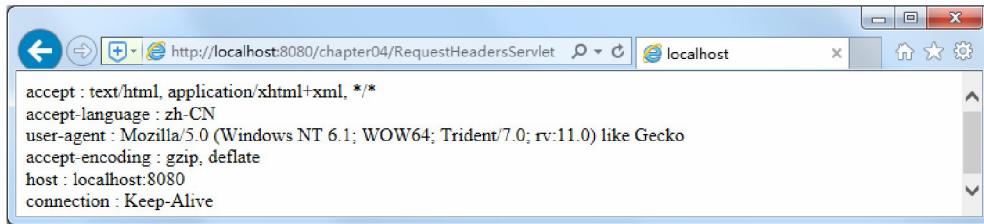
文件 4-2 RequestHeadersServlet.java

```

36 package cn.itcast.chapter04.request;
37 import java.io.IOException;
38 import java.io.PrintWriter;
39 import java.util.Enumeration;
40 import javax.servlet.*;
41 import javax.servlet.http.*;
42 public class RequestHeadersServlet extends HttpServlet {
43     public void doGet(HttpServletRequest request,
44         HttpServletResponse response) throws ServletException, IOException {
45         response.setContentType("text/html;charset=utf-8");
46         PrintWriter out = response.getWriter();
47         // 获取请求消息中所有头字段
48         Enumeration headerNames = request.getHeaderNames();
49         // 使用循环遍历所有请求头，并通过 getHeader() 方法获取一个指定名称的头字段
50         while (headerNames.hasMoreElements()) {
51             String headerName = (String) headerNames.nextElement();
52             out.print(headerName + " : "
53                     + request.getHeader(headerName) + "<br>");
```

60 }

在 web.xml 中配置完 RequestHeadersServlet 映射后，启动 Tomcat 服务器，在浏览器的地址栏中输入地址 “<http://localhost:8080/chapter04/RequestHeadersServlet>” 访问 RequestHeadersServlet，浏览器的显示结果如图 4-15 所示。



1.2.1.3 获取请求参数

在实际开发中，经常需要获取用户提交的表单数据，例如，用户名、密码、电子邮件等，为了方便获取表单中的请求参数，在 HttpServletRequest 接口中，定义了一系列获取请求参数的方法，如表 4-4 所示。

表 1-3 获取请求参数的方法

方法声明	功能描述
String getParameter(String name)	该方法用于获取某个指定名称的参数值，如果请求消息中没有包含指定名称的参数，getParameter()方法返回 null；如果指定名称的参数存在但没有设置值，则返回一个空串；如果请求消息中含有多个该指定名称的参数，getParameter()方法返回第一个出现的参数值
String[] getParameterValues(String name)	HTTP 请求消息中可以有多个相同名称的参数（通常由一个包含有多个同名的字段元素的 FORM 表单生成），如果要获得 HTTP 请求消息中的同一个参数名所对应的所有参数值，那么就应该使用getParameterValues()方法，该方法用于返回一个 String 类型的数组
Enumeration getParameterNames()	getParameterNames()方法用于返回一个包含请求消息中所有参数名的 Enumeration 对象，在此基础上，可以对请求消息中的所有参数进行遍历处理
Map getParameterMap()	getParameterMap()方法用于将请求消息中的所有参数名和值装入进一个 Map 对象中返回

表 4-4 中，列出了 HttpServletRequest 获取请求参数的一系列方法。其中，getParameter()方法用于获取某个指定的参数，而 getParameterValues()方法用于获取多个同名的参数。接下来，通过一个具体的案例，分步骤讲解这两个方法的使用，具体如下：

(1) 在 chapter04 项目的 WebContent 根目录下编写一个表单文件 form.html，如文件 4-13 所示。

文件 4-3 form.html

```
61 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
62           "http://www.w3.org/TR/html4/loose.dtd">
63 <html>
```

```

64 <head>
65 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
66 <title>Insert title here</title>
67 </head>
68 <body>
69     <form action="/chapter04/RequestParamsServlet" method="POST">
70         用户名: <input type="text" name="username"><br>
71         密 &nbsp;&nbsp;&nbsp;码: <input type="password" name="password">
72         <br>
73         爱好:
74         <input type="checkbox" name="hobby" value="sing">唱歌
75         <input type="checkbox" name="hobby" value="dance">跳舞
76         <input type="checkbox" name="hobby" value="football">足球<br>
77         <input type="submit" value="提交">
78     </form>
79 </body>
80 </html>

```

(2) 在 cn.itcast.chapter04.request 包中编写一个名称为 RequestParamsServlet 的 Servlet 类, 使用该 Servlet 获取请求参数, 如文件 4-14 所示。

文件4-4 RequestParamsServlet.java

```

81 package cn.itcast.chapter04.request;
82 import java.io.*;
83 import javax.servlet.*;
84 import javax.servlet.http.*;
85 public class RequestParamsServlet extends HttpServlet {
86     public void doGet(HttpServletRequest request,
87             HttpServletResponse response) throws ServletException, IOException {
88         String name = request.getParameter("username");
89         String password = request.getParameter("password");
90         System.out.println("用户名:" + name);
91         System.out.println("密 码:" + password);
92         // 获取参数名为“hobby”的值
93         String[] hobbies = request.getParameterValues("hobby");
94         System.out.print("爱好:");
95         for (int i = 0; i < hobbies.length; i++) {
96             System.out.print(hobbies[i] + ", ");
97         }
98     }
99     public void doPost(HttpServletRequest request,
100             HttpServletResponse response) throws ServletException, IOException {
101         doGet(request, response);
102     }
103}

```

在文件 4-14 中，由于参数名为“hobby”的值可能有多个，因此，需要使用 `getParameterValues()` 方法，获取多个同名参数的值，返回一个 `String` 类型的数组，通过遍历数组，打印出每个“hobby”参数对应的值。

(3) 在 `web.xml` 中配置完 `RequestParamsServlet` 映射后，启动 Tomcat 服务器，在浏览器的地址栏中输入地址“`http://localhost:8080/chapter04/form.html`”访问 `form.html` 页面，并填写表单相关信息，填写后的页面如图 4-18 所示。

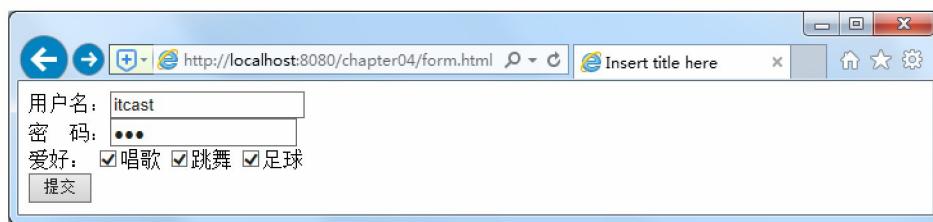


图3-2 运行结果

(4) 单击图 4-18 所示的“提交”按钮，在 Eclipse 的控制台打印出了每个参数的信息，如图 4-19 所示。

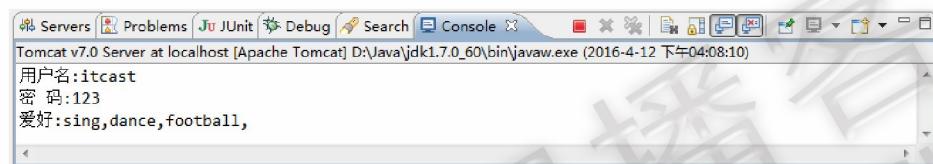


图3-3 运行结果

1.3 代码实现

```
package com.itheima.e_regist;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * 用户注册的 Servlet
 */
public class RegistServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        try{
            request.setCharacterEncoding("UTF-8");
            // 1.接收参数
            String username = request.getParameter("username");
            String password = request.getParameter("password");
        }
    }
}
```

```
String email = request.getParameter("email");
String name = request.getParameter("name");
String sex = request.getParameter("sex");
String telephone = request.getParameter("telephone");
// 2.封装数据
User user = new User();
user.setUsername(username);
user.setPassword(password);
user.setEmail(email);
user.setName(name);
user.setSex(sex);
user.setTelephone(telephone);

// 3.调用业务层处理数据
UserService userService = new UserService();
userService.regist(user);
// 4.页面跳转

response.sendRedirect(request.getContextPath() + "/demo2-regist/login.htm");
} catch (Exception e) {
    e.printStackTrace();
}

}

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

}
```

1.4 总结:

1.4.1 解决请求参数的中文乱码问题

在填写表单数据时，难免会输入中文，如姓名、公司名称等。在文件 4-13 中，由于 HTML 设置了浏览器在传递请求参数时，采用的编码方式是 UTF-8，但在解码时采用的是默认的 ISO8859-1，因此会导致乱码的出现。在浏览器的地址栏中输入地址“<http://localhost:8080/chapter04/form.html>”再次访问 form.html 页面，输入用户名为“传智播客”以及相关表单信息，如图 4-20 所示。



图3-4 运行结果

单击图 4-20 中的“提交”按钮，这时，控制台打印出了每个参数的值，具体如图 4-21 所示。

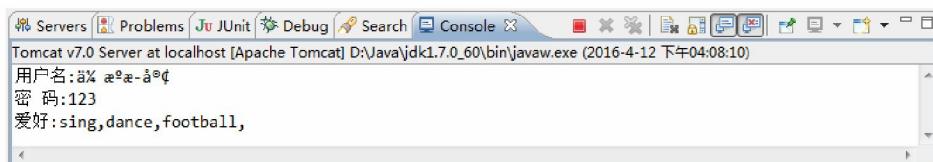


图4-21 运行结果

从图 4-21 可以看出，当输入的用户名为中文时，出现了乱码问题。通过本任务的学习，读者将学会如何处理请求参数的中文乱码问题。

【实现步骤】

1. 设置编码方式

在 HttpServletRequest 接口中，提供了一个 setCharacterEncoding()方法，该方法用于设置 request 对象的解码方式，接下来，对文件 4-14 进行修改，修改后的代码如文件 4-15 所示。该方法用于返回请求消息的实体部分的字符集编码。

文件4-5 RequestParamsServlet.java

```
package cn.itcast.chapter04.request;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class RequestParamsServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
        //设置 request 对象的解码方式
        request.setCharacterEncoding("utf-8");
        String name = request.getParameter("username");
        String password = request.getParameter("password");
        System.out.println("用户名: " + name);
        System.out.println("密 码: " + password);
        String[] hobbys = request.getParameterValues("hobby");
        System.out.print("爱好: ");
        for (int i = 0; i < hobbys.length; i++) {
            System.out.print(hobbys[i] + ", ");
        }
    }
    public void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

```
}
```

2. 查看运行结果

启动 Tomcat 服务器，再次访问 form.html 页面，输入中文用户名“传智播客”以及相关表单信息，控制台打印的结果如图 4-22 所示。

```
Tomcat v7.0 Server at localhost [Apache Tomcat] D:\Java\jdk1.7.0_60\bin\javaw.exe (2016-4-12 下午04:19:01)
用户名:传智播客
密码:123
爱好:sing,dance,football,
```

图3-6 运行结果

从图 4-22 可以看出，控制台输出的参数信息没有出现乱码。需要注意的是，这种解决乱码的方式只对 POST 方式有效，而对 GET 方式无效。为了验证 GET 方式的演示效果，接下来，将 form.html 文件中 method 属性的值改为“GET”。重新访问 form.html 页面并填写中文信息，控制台的打印结果如图 4-23 所示。

```
Tomcat v7.0 Server at localhost [Apache Tomcat] D:\Java\jdk1.7.0_60\bin\javaw.exe (2016-4-12 下午04:19:01)
用户名:ää ææ-åç
密码:123
爱好:sing,dance,football,
```

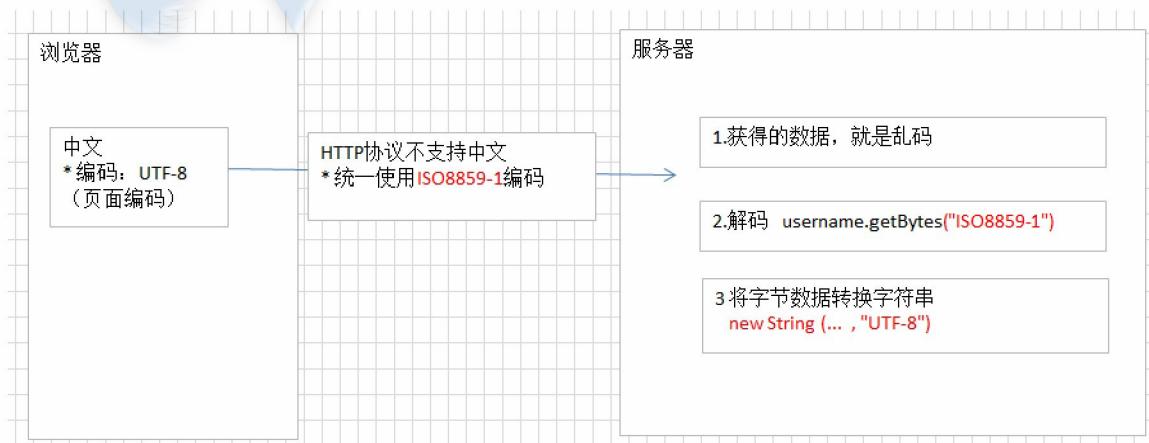
图3-7 运行结果

从图 4-23 中可以看出，使用 GET 方式提交表单，用户名出现了乱码，这就验证了 setCharacterEncoding() 方法只对 POST 提交方式有效的结论。为了解决 GET 方式提交表单时出现的中文乱码问题，接下来，对文件 4-15 进行修改，在第 10 行和第 11 行代码之间增加一行代码，如下所示：

```
name=new String(name.getBytes("iso8859-1"), "utf-8");
```

重启 Tomcat 服务器，再次访问 form.html 网页，输入中文用户名“传智播客”，这时，控制台的打印结果没有出现乱码，如图 4-24 所示。

```
Tomcat v7.0 Server at localhost [Apache Tomcat] D:\Java\jdk1.7.0_60\bin\javaw.exe (2016-4-12 下午04:22:48)
用户名:传智播客
密码:123
爱好:sing,dance,football,
```



第2章 案例：登录错误提示信息

2.1 需求

2.2 相关知识点

2.2.1 通过 Request 对象传递数据

Request 对象不仅可以获取一系列数据，还可以通过属性传递数据。在 `ServletRequest` 接口中，定义了一系列操作属性的方法，具体如下：

- **setAttribute()**方法

该方法用于将一个对象与一个名称关联后存储进 `ServletRequest` 对象中，其完整语法定义如下：

```
public void setAttribute(java.lang.String name,java.lang.Object o);
```

需要注意的是，如果 `ServletRequest` 对象中已经存在指定名称的属性，`setAttribute()`方法将会先删除原来的属性，然后再添加新的属性。如果传递给 `setAttribute()` 方法的属性值对象为 `null`，则删除指定名称的属性，这时的效果等同于 `removeAttribute()` 方法。

- **getAttribute()**方法

该方法用于从 `ServletRequest` 对象中返回指定名称的属性对象，其完整的语法定义如下：

```
public java.lang.String getAttribute (java.lang.String name);
```

- **removeAttribute()**方法

该方法用于从 `ServletRequest` 对象中删除指定名称的属性，其完整的语法定义如下：

```
public void removeAttribute (java.lang.String name);
```

- **getAttributeNames()**方法

该方法用于返回一个包含 `ServletRequest` 对象中的所有属性名的 `Enumeration` 对象，在此基础上，可以对 `ServletRequest` 对象中的所有属性进行遍历处理。`getAttributeNames()` 方法的完整语法定义如下：

```
public java.util.Enumeration getAttributeNames ();
```

需要注意的是，只有属于同一个请求中的数据才可以通过 `ServletRequest` 对象传递数据。关于 `ServletRequest` 对象操作属性的具体用法，将在后面的小节进行详细讲解。在此，大家只需了解即可。

2.2.2 RequestDispatcher 对象的应用

2.2.2.1 RequestDispatcher 接口

当一个 Web 资源收到客户端的请求后，如果希望服务器通知另外一个资源去处理请求，这时，除了使用 `sendRedirect()` 方法实现请求重定向外，还可以通过 `RequestDispatcher` 接口的实例对象来实现。在 `ServletRequest` 接口中定义了一个获取 `RequestDispatcher` 对象的方法，如表 4-5 所示。

表1-4 获取 RequestDispatcher 对象的方法

方法声明	功能描述
getRequestDispatcher(String path)	返回封装了某个路径所指定资源的 RequestDispatcher 对象。其中，参数 path 必须以 “/” 开头，用于表示当前 Web 应用的根目录。需要注意的是，WEB-INF 目录中的内容对 RequestDispatcher 对象也是可见的，因此，传递给 getRequestDispatcher(String path) 方法的资源可以是 WEB-INF 目录中的文件

获取到 RequestDispatcher 对象后，最重要的工作就是通知其它 Web 资源处理当前的 Servlet 请求，为此，在 RequestDispatcher 接口中，定义了两个相关方法，如表 4-6 所示。

表1-5 RequestDispatcher 接口的方法

方法声明	功能描述
forward(ServletRequest request, ServletResponse response)	该方法用于将请求从一个 Servlet 传递给另外的一个 Web 资源。在 Servlet 中，可以对请求做一个初步处理，然后通过调用这个方法，将请求传递给其它资源进行响应。需要注意的是，该方法必须在响应提交给客户端之前被调用，否则将抛出 IllegalStateException 异常
include(ServletRequest request, ServletResponse response)	该方法用于将其它的资源作为当前响应内容包含进来

表 4-6 列举的两个方法中，forward()方法可以实现请求转发，include()方法可以实现请求包含，关于请求转发和请求包含的相关知识，将在下面的小节中进行详细讲解。

2.2.2.2 请求转发

在 Servlet 中，如果当前 Web 资源不想处理请求时，可以通过 forward()方法将当前请求传递给其它的 Web 资源进行处理，这种方式称为请求转发。为了使读者更好地理解使用 forward()方法实现请求转发的工作原理，接下来通过一张图来描述，如图 4-25 所示。

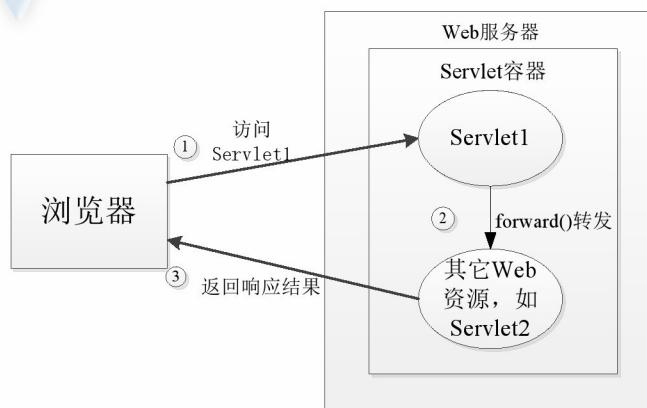


图3-8 forward()方法的工作原理

从图 4-25 中可以看出，当客户端访问 Servlet1 时，可以通过 forward()方法将请求转发给其它 Web 资源，其它 Web 资源处理完请求后，直接将响应结果返回到客户端。

了解了 forward()方法的工作原理后，接下来，通过一个案例来学习 forward()方法的使用。

在 chapter04 项目的 cn.itcast.chapter04.request 包中编写一个名为 RequestForwardServlet 的 Servlet 类，该类使用 forward()方法将请求转发到一个新的 Servlet 页面，如文件 4-16 所示。

文件4-6 RequestForwardServlet.java

```

1 package cn.itcast.chapter04.request;
2 import java.io.IOException;
3 import javax.servlet.*;
4 import javax.servlet.http.*;
5 public class RequestForwardServlet extends HttpServlet {
6     public void doGet(HttpServletRequest request,
7         HttpServletResponse response) throws ServletException, IOException {
8         response.setContentType("text/html;charset=utf-8");
9         // 将数据存储到 request 对象中
10        request.setAttribute("company", "北京传智播客教育有限公司");
11        RequestDispatcher dispatcher = request
12            .getRequestDispatcher("/ResultServlet");
13        dispatcher.forward(request, response);
14    }
15    public void doPost(HttpServletRequest request,
16        HttpServletResponse response) throws ServletException, IOException {
17        doGet(request, response);
18    }
19 }
```

在文件 4-16 中，通过使用 forward()方法，将当前 Servlet 的请求转发到 ResultServlet 页面，在 cn.itcast.chapter04.request 包中编写一个名为 ResultServlet 的 Servlet 类，该类用于获取 RequestForwardServlet 类中保存在 request 对象中的数据并输出，ResultServlet 类的代码实现如文件 4-17 所示。

文件4-7 ResultServlet.java

```

104package cn.itcast.chapter04.request;
105import java.io.*;
106import javax.servlet.*;
107import javax.servlet.http.*;
108public class ResultServlet extends HttpServlet {
109    public void doGet(HttpServletRequest request,
110        HttpServletResponse response) throws ServletException, IOException {
111        response.setContentType("text/html;charset=utf-8");
112        // 获取 PrintWriter 对象用于输出信息
113        PrintWriter out = response.getWriter();
114        // 获取 request 请求对象中保存的数据
115        String company = (String) request.getAttribute("company");
116        if (company != null) {
117            out.println("公司名称: " + company + "<br>");
118        }
119    }
```

```

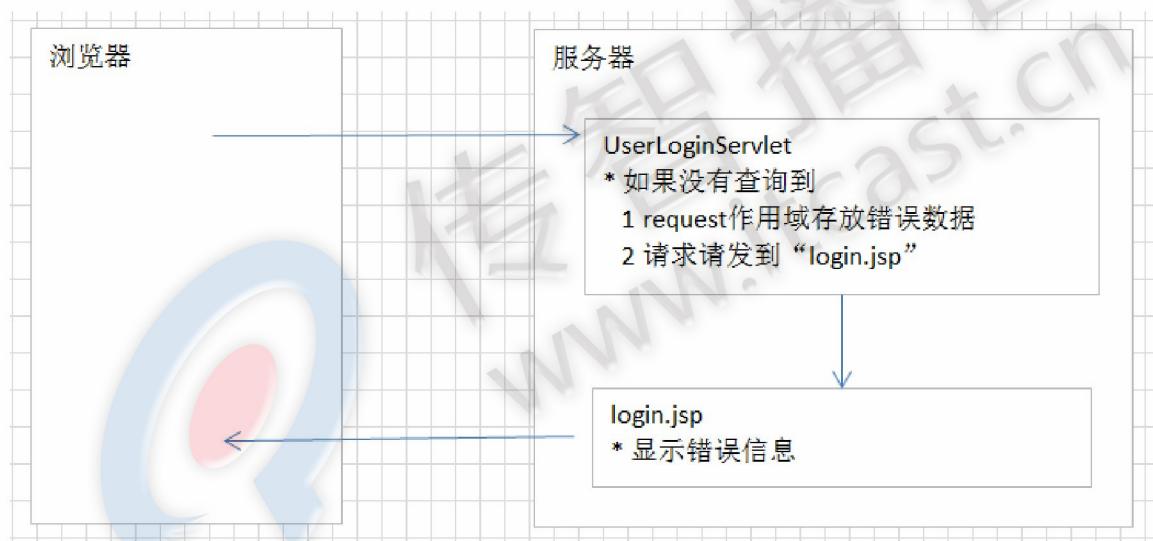
120     public void doPost(HttpServletRequest request,
121             HttpServletResponse response) throws ServletException, IOException {
122         doGet(request, response);
123     }
124 }
```

在 web.xml 中，添加完两个 Servlet 的映射信息后，启动 Tomcat 服务器，在浏览器中输入地址“<http://localhost:8080/chapter04/RequestForwardServlet>”访问 RequestForwardServlet，浏览器的显示结果如图 4-26 所示。



图3-9 运行结果

2.3 分析



2.4 实现

- 步骤 1：修改“UserLoginServlet”，登录错误时，使用 request 设置设置错误信息，并请求转发到“login.jsp”。
 - 1. 创建“login.jsp”
 - 2. 添加页面编码（固定写法）

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
```

 - 3. 复制“login.html”内容到“login.jsp”

```
File: UserLoginServlet.java
67         response.getWriter().println("</script>");
68     } else {
69         //登录不成功
70     //    response.getWriter().print("error");
71     //    * request作用设置错误信息
72     request.setAttribute("msg", "用户名或密码不匹配");
73     // * 请求转发到登录页面
74     request.getRequestDispatcher("login.jsp").forward(request, response);
75 }
76
// * request作用设置错误信息
request.setAttribute("msg", "用户名或密码不匹配");
// * 请求转发到登录页面
request.getRequestDispatcher("login.jsp").forward(request, response);
```

- 步骤 2: 在 “login.jsp” 页面显示错误信息 (固定写法)

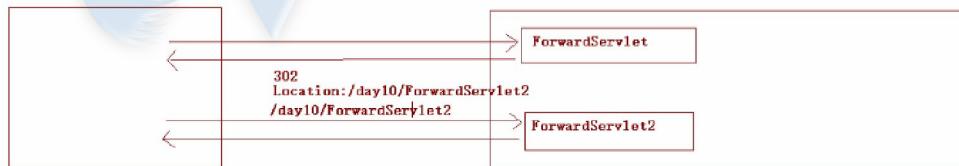
```
File: login.jsp
117 <font>会员登录</font>USER LOGIN
118
119 <div> <%=request.getAttribute("msg") %> </div>
120
<%=request.getAttribute("msg") %>
```

2.5 总结

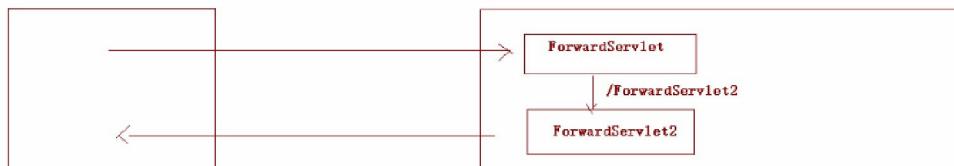
2.5.1 重定向和转发的区别:(redirect 和 forward 的区别)

重定向和转发的区别:

重定向:



转发:



- * 1.重定向的地址栏会发生变化,转发的地址栏不变.
- * 2.重定向两次请求两次响应,转发一次请求一次响应.
- * 3.重定向路径需要加工程名,转发的路径不需要加工程名.

* 4.重定向可以跳转到任意网站,转发只能在服务器内部进行转发.

