

Response 响应对象

今日内容介绍

- ◆ 案例一：文件下载
- ◆ 案例二：生成验证码

今日内容学习目标

- ◆ 使用 response 成功发送中文数据
- ◆ 使用 response 可以设置响应头

第1章 案例：完成文件的下载.

1.1 需求：

在登录成功后，页面跳转到文件下载的列表的页面，点击列表中的某些链接，下载文件。

文件下载的列表页面

超链接的下载

[hello.txt](#)
[cs10001.jpg](#)
[hello.zip](#)

手动编码方式下载

[hello.txt](#)
[cs10001.jpg](#)
[hello.zip](#)
[美女.jpg](#)

1.2 相关知识点：

1.2.1 HttpServletResponse 对象

在 Servlet API 中，定义了一个 `HttpServletResponse` 接口，它继承自 `ServletResponse` 接口，专门用来封装 HTTP 响应消息。由于 HTTP 响应消息分为状态行、响应消息头、消息体三部分，因此，在 `HttpServletResponse` 接口中定义了向客户端发送响应状态码、响应消息头、响应消息体的方法，接下来，本节将针对这些方法进行详细的讲解。

1.2.1.1 发送状态码相关的方法

当 Servlet 向客户端回送响应消息时，需要在响应消息中设置状态码。为此，在 `HttpServletResponse` 接口中，定义了两个发送状态码的方法，具体如下。

1. `setStatus(int status)` 方法

该方法用于设置 HTTP 响应消息的状态码，并生成响应状态行。由于响应状态行中的状态描述信息直接与状态码相关，而 HTTP 版本由服务器确定，因此，只要通过 `setStatus(int status)` 方法设置了状态码，即可实现状态行的发送。需要注意的是，正常情况下，Web 服务器会默认产生一个状态码为 200 的状态行。

2. `sendError(int sc)` 方法

该方法用于发送表示错误信息的状态码，例如，404 状态码表示找不到客户端请求的资源。在 `response` 对象中，提供了两个重载的 `sendError(int sc)` 方法，具体如下：

```
public void sendError(int code) throws java.io.IOException  
public void sendError(int code, String message) throws java.io.IOException
```

在上面重载的两个方法中，第一个方法只是发送错误信息的状态码，而第二个方法除了发送状态码外，还可以增加一条用于提示说明的文本信息，该文本信息将出现在发送给客户端的正文内容中。

1.2.1.2 发送响应消息头相关的方法

当 Servlet 向客户端发送响应消息时，由于 HTTP 协议的响应头字段有很多种，为此，在 HttpServletResponse 接口中，定义了一系列设置 HTTP 响应头字段的方法，如表 4-1 所示。

表1-1 设置响应消息头字段的方法

方法声明	功能描述
void addHeader(String name, String value)	这两个方法都是用来设置 HTTP 协议的响应头字段，其中，参数 name 用于指定响应头字段的名称，参数 value 用于指定响应头字段的值。不同的是，addHeader()方法可以增加同名的响应头字段，而 setHeader()方法则会覆盖同名的头字段
void addIntHeader(String name, int value)	这两个方法专门用于设置包含整数值的响应头。避免了使用 addHeader()与 setHeader()方法时，需要将 int 类型的设置值转换为 String 类型的麻烦
void setIntHeader(String name, int value)	
void setContentLength(int len)	该方法用于设置响应消息的实体内容的大小，单位为字节。对于 HTTP 协议来说，这个方法就是设置 Content-Length 响应头字段的值
void setContentType(String type)	该方法用于设置 Servlet 输出内容的 MIME 类型，对于 HTTP 协议来说，就是设置 Content-Type 响应头字段的值。例如，如果发送到客户端的内容是 jpeg 格式的图像数据，就需要将响应头字段的类型设置为 “image/jpeg”。需要注意的是，如果响应的内容为文本，setContentType()方法的还可以设置字符编码，如：text/html;charset=UTF-8
void setLocale(Locale loc)	该方法用于设置响应消息的本地化信息。对 HTTP 来说，就是设置 Content-Language 响应头字段和 Content-Type 头字段中的字符集编码部分。需要注意的是，如果 HTTP 消息没有设置 Content-Type 头字段，setLocale()方法设置的字符集编码不会出现在 HTTP 消息的响应头中，如果调用 setCharacterEncoding()或 setContentType()方法指定了响应内容的字符集编码，setLocale()方法将不再具有指定字符集编码的功能
void setCharacterEncoding(String charset)	该方法用于设置输出内容使用的字符编码，对 HTTP 协议来说，就是设置 Content-Type 头字段中的字符集编码部分。如果没有设置 Content-Type 头字段，setCharacterEncoding 方法设置的字符集编码不会出现在 HTTP 消息的响应头中。setCharacterEncoding()方法比 setContentType()和 setLocale()方法的优先权高，它的设置结果将覆盖 setContentType() 和 setLocale()方法所设置的字符码表

需要注意的是，在表 4-1 列举的一系列方法中，`addHeader()`、`setHeader()`、`addIntHeader()`、`setIntHeader()`方法都是用于设置各种头字段的，而`setContetType()`、`setLoacale()`和`setCharacterEncoding()`方法用于设置字符编码，这些设置字符编码的方法可以有效解决乱码问题。

1.2.1.3 发送响应消息体相关的方法

由于在 HTTP 响应消息中，大量的数据都是通过响应消息体传递的，因此，`ServletResponse` 遵循以 IO 流传递大量数据的设计理念。在发送响应消息体时，定义了两个与输出流相关的方法，具体如下。

1.`getOutputStream()`方法

该方法所获取的字节输出流对象为`ServletOutputStream`类型。由于`ServletOutputStream`是`OutputStream`的子类，它可以直接输出字节数组中的二进制数据。因此，要想输出二进制格式的响应正文，就需要使用`getOutputStream()`方法。

2.`getWriter()`方法

该方法所获取的字符输出流对象为`PrintWriter`类型。由于`PrintWriter`类型的对象可以直接输出字符文本内容，因此，要想输出内容全为字符文本的网页文档，需要使用`getWriter()`方法。

了解了`response`对象发送响应消息体的两个方法后，接下来，通过一个案例来学习这两个方法的使用。

在 Eclipse 中创建 Web 项目 chapter04，在项目的 src 目录下，新建一个名称为`cn.itcast.chapter04.response`的包，在包中编写一个名为`PrintServlet`的 Servlet 类，该类中使用了`response`对象的`getOutPutStream()`方法获取输出流对象，如文件 4-1 所示。

文件 4-1 PrintServlet.java

```

1 package cn.itcast.chapter04.response;
2 import java.io.*;
3 import javax.servlet.*;
4 import javax.servlet.http.*;
5 public class PrintServlet extends HttpServlet {
6     public void doGet(HttpServletRequest request,
7             HttpServletResponse response) throws ServletException, IOException {
8         String data = "itcast";
9         // 获取字节输出流对象
10        OutputStream out = response.getOutputStream();
11        out.write(data.getBytes());// 输出信息
12    }
13    public void doPost(HttpServletRequest request,
14            HttpServletResponse response) throws ServletException, IOException {
15        doGet(request, response);
16    }
17 }
```

在 web.xml 中配置完`PrintServlet`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/PrintServlet>”访问`PrintServlet`，浏览器的显示结果如图 4-2 所示。



图3-1 运行结果

从图 4-2 中可以看出，浏览器显示出了 response 对象响应的数据。由此可见，response 对象的 `getOutputStream()` 方法可以很方便的发送响应消息体。

接下来，对文件 4-1 进行修改，使用 `getWriter()` 方法发送消息体，修改后的代码如文件 4-2 所示。

文件4-2 PrintServlet.java

```

1 package cn.itcast.chapter04.response;
2 import java.io.*;
3 import javax.servlet.*;
4 import javax.servlet.http.*;
5 public class PrintServlet extends HttpServlet {
6     public void doGet(HttpServletRequest request,
7         HttpServletResponse response) throws ServletException, IOException {
8         String data = "itcast";
9         // 获取字符输出流对象
10        PrintWriter print = response.getWriter();
11        print.write(data); // 输出信息
12    }
13    public void doPost(HttpServletRequest request,
14        HttpServletResponse response) throws ServletException, IOException {
15        doGet(request, response);
16    }
17 }
```

重启 Tomcat 服务器，在浏览器的地址栏中输入地址 “<http://localhost:8080/chapter04/PrintServlet>” 再次访问 PrintServlet，浏览器的显示结果同样如图 4-2 所示。

注意：

虽然 response 对象的 `getOutputStream()` 和 `getWriter()` 方法都可以发送响应消息体，但是，它们之间互相排斥，不可同时使用，否则会发生 `IllegalStateException` 异常，如图 4-3 所示。

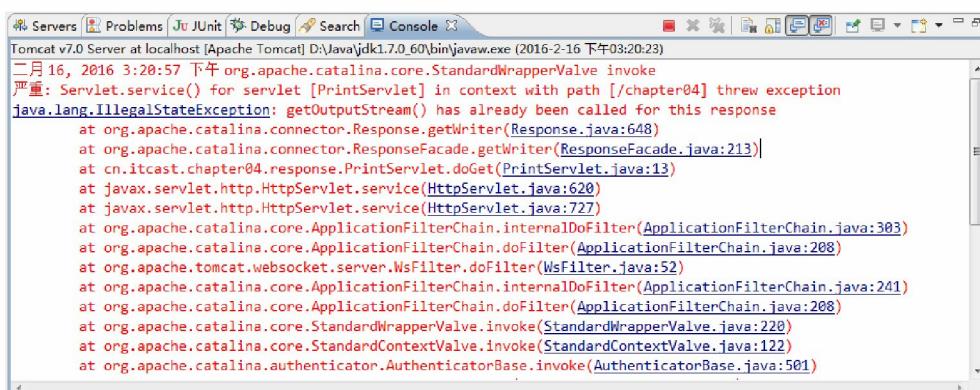


图3-2 运行结果

图 4-3 中发生异常的原因就是在 Servlet 中，调用 `response.getWriter()` 方法之前已经调用了

response.getOutputStream() 方法。

1.2.2 文件下载原理

对于文件下载，相信读者并不会陌生，因为通常在上网时所下的图片、文档和影片等都是文件下载的范畴。现在很多网站都提供了下载各类资源的功能，因此在学习 Web 开发过程中，有必要学习文件下载的实现方式。

实现文件下载功能比较简单，通常情况下，不需要使用第三方组件实现，而是直接使用 Servlet 类和输入/输出流实现。与访问服务器文件不同的是，要实现文件的下载，不仅需要指定文件的路径，还需要在 HTTP 协议中设置两个响应消息头，具体如下：

```
//设定接收程序处理数据的方式
Content-Disposition: attachment; filename =
//设定实体内容的MIME类型
Content-Type: application/x-msdownload
```

浏览器通常会直接处理响应的实体内容，需要在 HTTP 响应消息中设置两个响应消息头字段，用来指定接收程序处理数据内容的方式为下载方式。当单击“下载”超链接时，系统将请求提交到对应的 Servlet。在该 Servlet 中，首先获取下载文件的地址，并根据该地址创建文件字节输入流，然后通过该流读取下载文件内容，最后将读取的内容通过输出流写到目标文件中。

1.3 代码实现

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // 1.接收参数
        String filename = request.getParameter("filename");
        // 2.完成文件下载:
        // 2.1 设置Content-Type 头
        String type = this.getServletContext().getMimeType(filename);
        response.setHeader("Content-Type", type);
        // 2.2 设置Content-Disposition 头
        response.setHeader("Content-Disposition",
"attachment;filename="+filename);
        // 2.3 设置文件的InputStream.
        String realPath = this.getServletContext().getRealPath("/download/"+filename);
        InputStream is = new FileInputStream(realPath);
        // 获得response 的输出流:
        OutputStream os = response.getOutputStream();
        int len = 0;
        byte[] b = new byte[1024];
        while ((len = is.read(b)) != -1) {
            os.write(b, 0, len);
        }
    }
}
```

```
        while((len = is.read(b)) != -1) {
            os.write(b, 0, len);
        }
        is.close();
    }

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

}
```

1.4 总结：

1.4.1 中文文件的下载：

IE 浏览器下载中文文件的时候采用的 URL 的编码.

Firefox 浏览器下载中文文件的时候采用的是 Base64 的编码.

```
/**
 * 文件下载的 Servlet
 */
public class DownloadServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // 1.接收参数
        String filename = new String(request.getParameter("filename").getBytes("ISO-8859-1"), "UTF-8");
        System.out.println(filename);

        // 2.完成文件下载:
        // 2.1 设置 Content-Type 头
        String type = this.getServletContext().getMimeType(filename);
        response.setHeader("Content-Type", type);
        // 2.3 设置文件的 InputStream.
        String realPath = this.getServletContext().getRealPath("/download/" + filename);

        // 根据浏览器的类型处理中文文件的乱码问题:
        String agent = request.getHeader("User-Agent");
        System.out.println(agent);
        if(agent.contains("Firefox")) {
```

```

        filename = base64EncodeFileName(filename);
    }else{
        filename = URLEncoder.encode(filename,"UTF-8");
    }

    // 2.2 设置 Content-Disposition 头
    response.setHeader("Content-Disposition",
"attachment;filename='"+filename+"';

InputStream is = new FileInputStream(realPath);
// 获得 response 的输出流:
OutputStream os = response.getOutputStream();
int len = 0;
byte[] b = new byte[1024];
while((len = is.read(b)) != -1) {
    os.write(b, 0, len);
}
is.close();
}

public static String base64EncodeFileName(String fileName) {
    BASE64Encoder base64Encoder = new BASE64Encoder();
    try {
        return "=?UTF-8?B?"
            + new String(base64Encoder.encode(fileName
                .getBytes("UTF-8"))) + "?=";
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
}

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

}

```

1.4.2 response 解决中文输出乱码问题

由于计算机中的数据都是以二进制形式存储的，因此，当传输文本时，就会发生字符和字节之间的转换。字符与字节之间的转换是通过查码表完成的，将字符转换成字节的过程称为编码，将字节转换成字符的过程称为解码，如果编码和解码使用的码表不一致，就会导致乱码问题。通过本任

务的学习，读者能够解决中文输出乱码问题。

【实现步骤】

1. 创建 Servlet

在 day10 项目的 cn.itcast.chapter04.response 包中编写一个名为 ChineseServlet 的类，在该类中定义一个中文字字符串，然后使用字符输出流输出，如文件 4-3 所示。

文件4-3 ChineseServlet.java

```

18 package cn.itcast.chapter04.response;
19 import java.io.*;
20 import javax.servlet.*;
21 import javax.servlet.http.*;
22 public class ChineseServlet extends HttpServlet {
23     public void doGet(HttpServletRequest request,
24         HttpServletResponse response) throws ServletException, IOException {
25         String data = "中国";
26         PrintWriter out = response.getWriter();
27         out.println(data);
28     }
29     public void doPost(HttpServletRequest request,
30         HttpServletResponse response) throws ServletException, IOException {
31         doGet(request, response);
32     }
33 }
```

2. 配置映射信息，查看运行结果

在 web.xml 中配置完 ChineseServlet 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/day10/ChineseServlet>”访问 ChineseServlet，浏览器的显示结果如图 4-4 所示。



图3-3 运行结果

从图 4-4 中可以看出，浏览器显示的内容都是“??”，说明发生了乱码问题。实际上此处产生乱码的原因是 response 对象的字符输出流在编码时，采用的是 ISO-8859-1 的字符码表，该码表并不兼容中文，会将“中国”编码为“63 63”（在 ISO8859-1 的码表中查不到的字符就会显示 63）。当浏览器对接收到的数据进行解码时，会采用默认的码表 GB2312，将“63”解码为“？”，因此，浏览器将“中国”两个字符显示成了“??”，具体分析如图 4-5 所示。



图3-4 编码错误分析

为了解决上述编码错误，在 `HttpServletResponse` 接口中，提供了一个 `setCharacterEncoding()` 方法，该方法用于设置字符的编码方式，接下来对文件 4-3 进行修改，在第 7 行和第 8 行代码之间增加一行代码，设置字符编码使用的码表为 UTF-8，代码具体如下：

```
response.setCharacterEncoding("utf-8");
```

在浏览器的地址栏中输入 URL “<http://localhost:8080/chapter04/ChineseServlet>” 再次访问 `ChineseServlet`，浏览器的显示结果如图 4-6 所示。



图3-5 运行结果

从图 4-6 可以看出，浏览器中显示的乱码虽然不是“??”，但也不是需要输出的“中国”。通过分析发现，这是由于浏览器解码错误导致的，因为 `response` 对象的字符输出流设置的编码方式为 UTF-8，而浏览器使用的解码方式是 GB2312，具体分析过程如图 4-7 所示。



图3-6 解码错误分析

对于图 4-7 所示的解码错误，可以通过修改浏览器的解码方式解决。在浏览器中点击菜单栏中的【查看】→【编码】→【Unicode（UTF-8）】选项（或者鼠标右击浏览器内部，在弹出的窗口中选择【编码】→【其它】→【Unicode（UTF-8）】），将浏览器的编码方式设置为 UTF-8，浏览器的显示结果如图 4-8 所示。



图3-7 运行结果

从图 4-8 中可以看出，浏览器的显示内容没有出现乱码，由此说明，通过修改浏览器的编码方式可以解决乱码，但是，这样的做法仍然是不可取的，因为你不能让用户每次都去改动浏览器编码。为此，在 `HttpServletResponse` 对象中，提供了两种解决乱码的方案，具体如下：

第一种方式：

```
// 设置 HttpServletResponse 使用 utf-8 编码
response.setCharacterEncoding("utf-8");
// 通知浏览器使用 utf-8 解码
response.setHeader("Content-Type", "text/html; charset=utf-8");
```

第二种方式：

```
// 包含第一种方式的两个功能
response.setContentType("text/html; charset=utf-8");
```

通常情况下，为了使代码更加简洁，会采用第二种方式。接下来，对文件 4-3 进行修改，使用 `HttpServletResponse` 对象的第二种方式来解决乱码问题，修改后的代码如文件 4-4 所示。

文件4-4 ChineseServlet.java

```
34 package cn.itcast.chapter04.response;
35 import java.io.*;
36 import javax.servlet.*;
37 import javax.servlet.http.*;
38 public class ChineseServlet extends HttpServlet {
39     public void doGet(HttpServletRequest request,
40         HttpServletResponse response) throws ServletException, IOException {
41         //设置字符编码
42         response.setContentType("text/html;charset=utf-8");
43         String data="中国";
44         PrintWriter out = response.getWriter();
45         out.println(data);
46     }
47     public void doPost(HttpServletRequest request,
48         HttpServletResponse response) throws ServletException, IOException {
49         doGet(request,response);
50     }
51 }
```

启动 Tomcat 服务器，在浏览器的地址栏中输入地址
“<http://localhost:8080/chapter04/ChineseServlet>”访问 `ChineseServlet`，浏览器显示出了正确的中文字符，如图 4-9 所示。



第2章 案例：验证码

2.1 需求：

在访问登录页面时，需要生产验证码。从而防止用户使用程序恶意登录。

会员登录 USER LOGIN

用户名

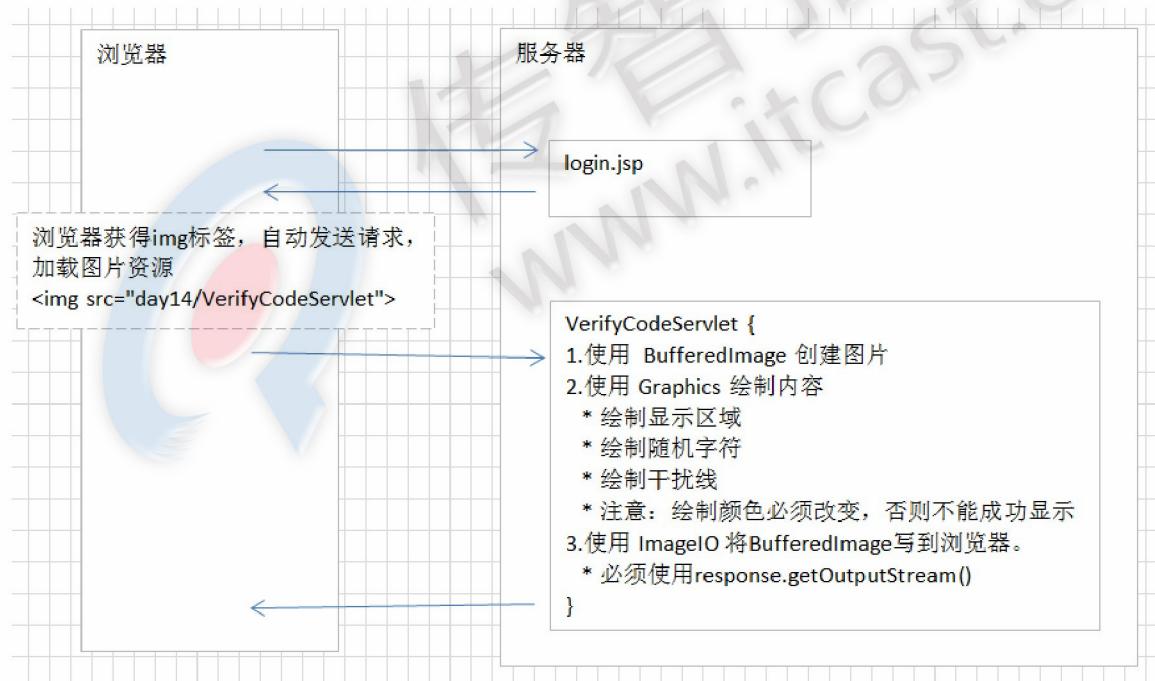
密码

验证码

自动登录 记住用户名

登录

2.2 分析



2.3 代码实现

- 步骤 1：修改 login.jsp 页面，确定验证码图片显示的位置

```

login.jsp
80      <input type="text" class="form-control" id="inputPassword3" placeholder="请输入密码" style="width: 150px; margin-bottom: 10px;">
81      </div>
82      <div class="col-sm-3">
83          
84      </div>
85

```

● 步骤 2：编写 VerifyCodeServlet 实现类

```

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

    //生成图片
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        //1 高和宽
        int height = 30;
        int width = 60;
        String data = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz";
        Random random = new Random();

        //2 创建一个图片
        BufferedImage image = new BufferedImage(width, height,
        BufferedImage.TYPE_INT_RGB);

        //3 获得画板
        Graphics g = image.getGraphics();
        //4 填充一个矩形
        // * 设置颜色
        g.setColor(Color.BLACK);
        g.fillRect(0, 0, width, height);

        g.setColor(Color.WHITE);
        g.fillRect(1, 1, width-2, height-2);
        // * 设置字体
        g.setFont(new Font("宋体", Font.BOLD|Font.ITALIC, 25));

        //5 写随机字
        for(int i = 0 ; i < 4 ; i ++){
            // 设置颜色--随机数
            g.setColor(new Color(random.nextInt(255), random.nextInt(255),
            random.nextInt(255)));

```

```
// 获得随机字
int index = random.nextInt(data.length());
String str = data.substring(index, index + 1);

// 写入
g.drawString(str, width/6 * (i + 1), 20);

}

//6 干扰线
for(int i = 0 ; i < 3 ; i ++){
    // 设置颜色--随机数
    g.setColor(new Color(random.nextInt(255), random.nextInt(255),
random.nextInt(255)));
    // 随机绘制先
    g.drawLine(random.nextInt(width), random.nextInt(height),
random.nextInt(width), random.nextInt(height));
    // 随机点
    g.drawOval(random.nextInt(width), random.nextInt(height), 2, 2);
}

//end 将图片响应给浏览器
ImageIO.write(image, "jpg", response.getOutputStream());
}

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

}
```

● 步骤 3：编写 web.xml 文件

```
<servlet>
    <servlet-name>VerifyCodeServlet</servlet-name>
    <servlet-class>com.itheima.web.servlet.VerifyCodeServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>VerifyCodeServlet</servlet-name>
    <url-pattern>/VerifyCodeServlet</url-pattern>
</servlet-mapping>
```

2.4 扩展

- 添加点击图片或文字，更换验证码



- 代码实现，点击图片或文件，都可以更换验证码图片。

```
login.jsp
82  <div class="col-sm-5">
83      &nbsp;&nbsp;
85      <a href="javascript:void(0)" onclick="changeImg()" >换一张</a>
86      <script type="text/javascript">
87          function changeImg(){
88              $("#imgId").attr("src", "${pageContext.request.contextPath}/VerifyCodeServlet?t="+new Date());
89          }
90      </script>
91  </div>

<div class="col-sm-5">
    &nbsp;&nbsp;
    <a href="javascript:void(0)" onclick="changeImg()" >换一张</a>
    <script type="text/javascript">
        function changeImg() {

            $("#imgId").attr("src", "${pageContext.request.contextPath}/VerifyCodeServlet?t="
+new Date());
        }
    </script>
</div>
```