Servlet 客户端 HTTP 请求 Servlet HTTP 状态码

Servlet 服务器 HTTP 响应

正如前面的章节中讨论的那样，当一个 Web 服务器响应一个 HTTP 请求时，响应通常包括一个状态行、一些响应报头、一个空行和文档。一个典型的响应如下所示：

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
  (Blank Line)
<!doctype ...>
<html>
<head>...</head>
<body>
...
</body>
</html>
```

状态行包括 HTTP 版本（在本例中为 HTTP/1.1）、一个状态码（在本例中为 200）和一个对应于状态码的短消息（在本例中为 OK）。

下表总结了从 Web 服务器端返回到浏览器的最有用的 HTTP 1.1 响应报头，您会在 Web 编程中频繁地使用它们：

头信息	描述
Allow	这个头信息指定服务器支持的请求方法（GET、POST 等）。
Cache-Control	这个头信息指定响应文档在何种情况下可以安全地缓存。可能的值有：**public、private** 或 **no-cache** 等。Public 意味着文档是可缓存，Private 意味着文档是单个用户私用文档，且只能存储在私有（非共享）缓存中，no-cache 意味着文档不应被缓存。
Connection	这个头信息指示浏览器是否使用持久 HTTP 连接。值 **close** 指示浏览器不使用持久 HTTP 连接，值 **keep-alive** 意味着使用持久连接。
Content-Disposition	这个头信息可以让您请求浏览器要求用户以给定名称的文件把响应保存到磁盘。
Content-Encoding	在传输过程中，这个头信息指定页面的编码方式。
Content-Language	这个头信息表示文档编写所使用的语言。例如，en、en-us、ru 等。
Content-Length	这个头信息指示响应中的字节数。只有当浏览器使用持久（keep-alive）HTTP 连接时才需要这些信息。

Content-Type	这个头信息提供了响应文档的 MIME（Multipurpose Internet Mail Extension）类型。
Expires	这个头信息指定内容过期的时间，在这之后内容不再被缓存。
Last-Modified	这个头信息指示文档的最后修改时间。然后，客户端可以缓存文件，并在以后的请求中通过 **If-Modified-Since** 请求头信息提供一个日期。
Location	这个头信息应被包含在所有的带有状态码的响应中。在 300s 内，这会通知浏览器文档的地址。浏览器会自动重新连接到这个位置，并获取新的文档。
Refresh	这个头信息指定浏览器应该如何尽快请求更新的页面。您可以指定页面刷新的秒数。
Retry-After	这个头信息可以与 503（Service Unavailable 服务不可用）响应配合使用，这会告诉客户端多久就可以重复它的请求。
Set-Cookie	这个头信息指定一个与页面关联的 cookie。

设置 HTTP 响应报头的方法

下面的方法可用于在 Servlet 程序中设置 HTTP 响应报头。这些方法通过 *HttpServletResponse* 对象可用。

序号	方法 & 描述
1	**String encodeRedirectURL(String url)** 为 sendRedirect 方法中使用的指定的 URL 进行编码，或者如果编码不是必需的，则返回 URL 未改变。
2	**String encodeURL(String url)** 对包含 session 会话 ID 的指定 URL 进行编码，或者如果编码不是必需的，则返回 URL 未改变。
3	**boolean containsHeader(String name)** 返回一个布尔值，指示是否已经设置已命名的响应报头。
4	**boolean isCommitted()** 返回一个布尔值，指示响应是否已经提交。
5	**void addCookie(Cookie cookie)** 把指定的 cookie 添加到响应。
6	**void addDateHeader(String name, long date)** 添加一个带有给定的名称和日期值的响应报头。
7	**void addHeader(String name, String value)** 添加一个带有给定的名称和值的响应报头。
8	**void addIntHeader(String name, int value)**

	添加一个带有给定的名称和整数值的响应报头。
9	**void flushBuffer()** 强制任何在缓冲区中的内容被写入到客户端。
10	**void reset()** 清除缓冲区中存在的任何数据，包括状态码和头。
11	**void resetBuffer()** 清除响应中基础缓冲区的内容，不清除状态码和头。
12	**void sendError(int sc)** 使用指定的状态码发送错误响应到客户端，并清除缓冲区。
13	**void sendError(int sc, String msg)** 使用指定的状态发送错误响应到客户端。
14	**void sendRedirect(String location)** 使用指定的重定向位置 URL 发送临时重定向响应到客户端。
15	**void setBufferSize(int size)** 为响应主体设置首选的缓冲区大小。
16	**void setCharacterEncoding(String charset)** 设置被发送到客户端的响应的字符编码（MIME 字符集）例如，UTF-8。
17	**void setContentLength(int len)** 设置在 HTTP Servlet 响应中的内容主体的长度，该方法设置 HTTP Content-Length 头。
18	**void setContentType(String type)** 如果响应还未被提交，设置被发送到客户端的响应的内容类型。
19	**void setDateHeader(String name, long date)** 设置一个带有给定的名称和日期值的响应报头。
20	**void setHeader(String name, String value)** 设置一个带有给定的名称和值的响应报头。
21	**void setIntHeader(String name, int value)** 设置一个带有给定的名称和整数值的响应报头。
22	**void setLocale(Locale loc)** 如果响应还未被提交，设置响应的区域。

23	**void setStatus(int sc)** 为该响应设置状态码。

HTTP Header 响应实例

您已经在前面的实例中看到 setContentType() 方法，下面的实例也使用了同样的方法，此外，我们会用 **setIntHeader()** 方法来设置 **Refresh** 头。

```
//导入必需的 java 库
import java.io.IOException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

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

@WebServlet("/Refresh")

//扩展 HttpServlet 类
public class Refresh extends HttpServlet {

    // 处理 GET 方法请求的方法
      public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                throws ServletException, IOException
      {
          // 设置刷新自动加载时间为 5 秒
          response.setIntHeader("Refresh", 5);
          // 设置响应内容类型
          response.setContentType("text/html;charset=UTF-8");

          //使用默认时区和语言环境获得一个日历
          Calendar cale = Calendar.getInstance();
          //将Calendar类型转换成Date类型
          Date tasktime=cale.getTime();
          //设置日期输出的格式
          SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
          //格式化输出
          String nowTime = df.format(tasktime);
          PrintWriter out = response.getWriter();
          String title = "自动刷新 Header 设置 - 菜鸟教程实例";
          String docType =
```

```
            "<!DOCTYPE html>\n";
            out.println(docType +
              "<html>\n" +
              "<head><title>" + title + "</title></head>\n"+
              "<body bgcolor=\"#f0f0f0\">\n" +
              "<h1 align=\"center\">" + title + "</h1>\n" +
              "<p>当前时间是: " + nowTime + "</p>\n");
        }
        // 处理 POST 方法请求的方法
        public void doPost(HttpServletRequest request,
                           HttpServletResponse response)
            throws ServletException, IOException {
           doGet(request, response);
        }
}
```

以上测试实例是位于 TomcatTest 项目下，对应的 web.xml 配置为：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <servlet>
     <!-- 类名 -->
    <servlet-name>Refresh</servlet-name>
    <!-- 所在的包 -->
    <servlet-class>com.runoob.test.Refresh</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Refresh</servlet-name>
    <!-- 访问的网址 -->
    <url-pattern>/TomcatTest/Refresh</url-pattern>
    </servlet-mapping>
</web-app>
```

现在，调用上面的 Servlet，每隔 5 秒会显示当前系统时间。只要运行 Servlet 并稍等片刻，即可看到如下的结果：

← Servlet 客户端 HTTP 请求 Servlet HTTP 状态码 →

点我分享笔记