**day11-Request和Response**

1、掌握HTTPServletResponse的使用

2、熟悉HTTPServletRequest的使用

**第七章 Response讲解**

7.1 Response简介

​ 定义辅助 servlet 将响应发送到客户端的对象。servlet 容器创建 ServletResponse 对象，并将它作为参数传递给 servlet 的 service 方法。 要发送 MIME 正文响应中的二进制数据，请使用 #getOutputStream 返回的 ServletOutputStream。要发送字符数据，请使用 #getWriter 返回的 PrintWriter 对象。可使用 #setCharacterEncoding 和 #setContentType 显式指定 MIME 正文响应的 charset，或使用 #setLocale 方法隐式指定它。显式指定优先于隐式指定。如果未指定 charset，则将使用 ISO-8859-1。setCharacterEncoding、setContentType 或 setLocale 方法必须在调用 getWriter 之前，并且必须在提交采用要使用的字符编码的响应之前调用。

7.2 HttpServletResponse介绍

扩展 ServletResponse 接口以提供特定于 HTTP 的发送响应功能。例如，该接口拥有访问 HTTP 头和 cookie 的方法。 客户端向服务器发起的都是HTTP协议操作,所以我们大部分使用HttpServletResponse对象作为直接操作对象!

# 7.3 HttpServletResponse 常用API介绍

|  |  |
| --- | --- |
| **方法名称** | **作用** |
| **setStatus(int code)** | 设置响应状态码 |
| **setHeader(name,value)** | 设置响应信息头 |
| **setCharacterEncoding(String);** | 设置编码格式 |
| **setContentType(String)** | 设置返回数据mimetype |
| **getWriter()** | 获取字符输出流 |
| **getOutputStream()** | 获取字节输出流 |

7.4 设置返回字符编码格式

\* 方案1

|  |
| --- |
| response.setHeader("Content-type","text/html;charset=UTF-8")  方案按相对简单,通过设置响应头告知浏览器解析字符串的编码格式! |

\* 方案2

|  |
| --- |
| response.setContentType("text/html;charset=UTF-8")  利用setContentTyoe这种综合性的写法解决问题!此方法也是开发中常用的方法!方便! |

7.5 Response案例

1. 用户下载服务器图片

|  |
| --- |
| String path = getServletContext().getRealPath("/图片.png");  System.***out***.println(path);  //文件输入流  FileInputStream stream = **new** FileInputStream(path);  //获取文件的名字 /  String filename = path.substring(path.lastIndexOf(“//”)+1);  System.***out***.println(filename);  //设置响应头  //content-disposition", "attachment;filename="+filename  //下载用的 attachment下载： filename下载文件的名字  response.setHeader("content-disposition", "attachment;filename="+filename);  //根据文件名字的后缀名获取类型  String mimetype = getServletContext().getMimeType(filename);  System.***out***.println(mimetype);  response.setContentType(mimetype); //下载文件的类型  ServletOutputStream outputStream = response.getOutputStream();  **byte** [] buffer = **new** **byte** [8\*1024];  **int** len = 0;  **while**((len=stream.read(buffer)) != -1){  outputStream.write(buffer, 0, len);  }  stream.close(); |

2. 页面中添加验证码

1. 项目中添加第三方jar包validatecode.jar

2. 创建返回验证码的servlet!

|  |
| --- |
| //1.设置生成规则  /\*\*  \* width：验证码的宽度 px 像素  \* height: px  \* codeCount:生成验证码有几个数  \* lineCount:有几根线  \*/  ValidateCode code = **new** ValidateCode(200, 50, 6, 20);  //2.获取生成的验证码的字符串值  System.***out***.println(code.getCode()); //获取正确值  HttpSession session = request.getSession();  session.setAttribute("yzm",code.getCode());  //3.响应写回验证图片  code.write(response.getOutputStream()); |

**第八章 Request讲解**

8.1 ServletRequst介绍

定义将客户端请求信息提供给某个 servlet 的对象。servlet 容器创建 `ServletRequest` 对象，并将该对象作为参数传递给该 servlet 的 `service` 方法。

8.2 HttpServletRequest介绍

HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，HTTP请求头中的所有信息都封装在这个对象中，开发人员通过这个对象的方法，可以获得客户这些信息。

\*小结:\* 同响应相同,客户端请求协议都是基于HTTP所以我们选用HttpServletRequest来操作用户发送过来的请求的数据!

8.3 HttpServletRequest常用API

|  |  |
| --- | --- |
| 获取请求路径相关参数 | |
| getRequestURL | 返回客户端发出请求时的完整URL |
| getRequestURI | 返回请求行中的资源名部分 |
| getQueryString | 返回请求行中的参数部分 |
| getRemoteAddr | 返回发出请求的客户机的IP地址 |
| getRemotePort | 返回客户机所使用的网络端口号 |
| getLocalAddr | 返回WEB服务器的IP地址 |
| getMethod | 返回客户机请求方式 |

|  |  |
| --- | --- |
| 获取请求头信息 | |
| getHead(name) | 返回请求头的内容 |
| 获取请求正文参数 | |
| getParameter(name) | 返回指定参数请求内容 |
| getParameterValues（String name） | 返回指定参数请求内容 |
| getParameterNames() | 返回所有参数名 |
| getParameterMap() | 返回所有参数集合 |
|  |  |

8.4 请求案例

|  |
| --- |
| // 1.测试获取请求行数据的方法和请求头的方法  // 1.获取请求行的方法  // 获取请求方式  String method = request.getMethod();  // 获取请求的url  String url = request.getRequestURL() + "";  // 获取uri  String uri = request.getRequestURI();  // 获取请求的参数 get  String query = request.getQueryString();  // 获取请求人的id  String ip = request.getRemoteAddr();  // 获取请求的主机名  String host = request.getRemoteHost();  System.***out***.println(method + " url:" + url + " uri:" + uri + " query:" + query + " ip:" + ip + " host:" + host);  // 获取请求信息  // 全部输出  String header = request.getHeader("user-agent");  **if** (header.contains("firefox")) {  }  // 获取所有请求的names  Enumeration<String> headerNames = request.getHeaderNames();  **while** (headerNames.hasMoreElements()) {  String name = (String) headerNames.nextElement();  // 根据name获取keys  Enumeration<String> headers = request.getHeaders(name);  **while** (headers.hasMoreElements()) {  String key = (String) headers.nextElement();  System.***out***.println(name + "--->" + key);  }  } |

8.5 Servlet如何处理HTTP协议

Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的HttpServletRequest对象、和代表响应的HttpServletResponse对象。

HttpServletRequest和HttpServletResponse对象即然代表请求和响应，那我们要获取客户端提交过来的数据，只需要找HttpServletRequest对象就行了。要向客户端输出数据，只需要找HttpServletResponse对象就行了

#8.6.1 HttpServletRequest

HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，HTTP请求头中的所有信息都封装在这个对象中，通过这个对象提供的方法，可以获得客户端请求的所有信息

|  |
| --- |
| //编码格式  //request.setCharacterEncoding("UTF-8");  //获取参数方法  System.***out***.println(request.getParameter("name"));  System.***out***.println(Arrays.*toString*(request.getParameterValues("name")));  System.***out***.println(request.getParameterMap());  System.***out***.println(request.getParameterNames());  //获取请求方法  System.***out***.println("getMethod--->"+request.getMethod());  //获取远程ip地址  System.***out***.println(request.getRemoteAddr());  //获取远程端口号  System.***out***.println(request.getRemotePort());  System.***out***.println(request.getRemoteHost());  System.***out***.println(request.getRemoteUser());  System.***out***.println("getRequestURL:"+request.getRequestURL());  System.***out***.println("getContextPath:"+request.getContextPath());  System.***out***.println("getServletPath--->"+request.getServletPath());  System.***out***.println("getServerName--->"+request.getServerName());  Enumeration<String> ets=request.getHeaderNames();  **while** (ets.hasMoreElements()) {  String key=ets.nextElement();  System.***out***.println("请求消息头：key="+key+"---值："+request.getHeader(key));  } |

8.6.2 HttpServletResponse

HttpServletResponse对象代表服务器的响应。这个对象中封装了向客户端发送数据、发送响应头，发送响应状态码的方法。

|  |  |
| --- | --- |
| getOutputStream | 获取输出字节流向客户端写出内容 |
| getWriter | 获取输出字符流向客户端写出内容 |
| sendRedirect | 重定向 |
| addCookie | 添加cookie到客户端 |
| setHeader | 设置带有给定的名字和数值的响应消息头 |
| setStatus | 设置Http协议的状态响应码 |