1. 响应消息
2. 响应消息的组成
3. 请求行
4. 请求头
5. 请求空行
6. 请求体
7. 响应行
8. 组成

协议/版本 响应状态码 状态码描述

1. 响应状态码

服务器告诉客户端浏览器本次请求和响应的状态。状态码一般都是3位数字，分为5大类：

1. 1xx

一百多的数字代表着服务器接收到了客户端的请求，但是没有接收完成。

1. 2xx

二百多的数字代表着请求成功。

1. 3xx

三百多的数字代表着重定向，客户端向服务器发送请求，服务器将该请求重定向到其他位置，如缓存中（304）

1. 4xx

四百多的数字代表着客户端的请求错误，如404代表着客户端的请求路径中没有对应的资源，如405代表着请求方式没有对应的doxxx方法

1. 5xx

五百多的数字代表着服务器端的错误，如500代表着服务器内部异常。

1. 响应头
2. 格式

头名称：值

1. 常见响应头
2. Content-Type

服务器告诉客户端本次响应体数据格式以及编码格式

Content-Type: text/html;charset=UTF-8

1. Content-disposition

服务器告诉客户端以什么方式打开响应体数据

默认值为 in-line，代表着在当前页面打开

值为attachment;filename=xx代表着以附件形式打开响应体，一般用于文件下载中。

1. Response对象
2. 设置Response对象
3. 设置响应行
4. 格式

HTTP/1.1 200 ok

1. 设置状态码

setStatus(int src)

1. 设置响应头

setHeader(String name, String value)

1. 设置响应体
2. 获取输出流

字符输出流： printWriter getWriter()

字节输出流： ServletOutputStream getOutputStream()

1. 使用输出流，将数据输出到客户端浏览器中
2. 重定向实例
3. 思路

设置状态码为302，返回另一个资源的路径。

1. 代码实现

//访问demo1，会跳转到demo2  
System.*out*.println("demo1");  
//1.设置状态码为302  
resp.setStatus(302);  
//2. 设置响应头location  
resp.setHeader("location", "/Login/Response/responseDemo2");

//简单的重定向方法

resp.sendRedirect(“/Login/Response/responseDemo2”)

1. 重定向的特点
2. 地址栏发生变化
3. 重定向可以访问其他服务器的资源
4. 重定向是两次请求，因此不能像转发一样使用request域来共享数据
5. 路径的写法
6. 相对路径

以./开头的路径，要找到当前资源路径和目标资源路径之间的相对位置关系。

如：

./index.html

1. 绝对路径

通过绝对路径可以确定唯一的资源，可以省略协议和域名，直接以/开头写绝对路径，如：

/login/response

1. 路径的使用

对于客户端和服务器，我们提供的路径是不一样的。

若路径是提供给客户端浏览器的，则需要添加虚拟目录（项目的访问路径），这个虚拟目录可以通过动态获取，如：

request.getContextPath()

若路径是提供给服务器的，则不需要添加虚拟路径，直接写路径即可。

1. 输出字符数据实例
2. 中文字符的乱码问题

当我们想要输出中文字符到浏览器中时，会出现乱码问题，这是因为系统使用的编码表和Tomcat返回的字符输出流的编码表不一致。Tomcat服务器返回的字符输出流的编码表为ISO-8859-1，不支持中文，因此我们需要设置字符输出流的编码为支持中文的编码表，如GBK，”utf-8”，然后通过设置response的响应头的content-type属性，将使用的编码表告诉浏览器，建议浏览器使用该编码表解码。这样，乱码问题就解决了。

1. 思路
2. 设置输出流的编码方式
3. 获取响应的字符输出流
4. 输出数据到浏览器中
5. 实现代码

//设置流的默认编码  
resp.setCharacterEncoding("GBK");  
//告诉浏览器，服务器发送的消息体数据的编码格式  
resp.setHeader("content-type", "text/html;charset=GBK");  
//简单的设置编码方式  
resp.setContentType("text/html;charset=GBK");  
  
//获取字符输出流  
PrintWriter pw = resp.getWriter();  
//输出数据  
pw.write(" 你的 world");

1. 输出字节数据到浏览器
2. 思路

获取字节输出流

输出数据到浏览器

1. 代码实现

resp.setContentType("text/html;chareset=utf-8");  
  
//获取字节输出流  
ServletOutputStream sos = resp.getOutputStream();  
//输出数据  
sos.write("你好".getBytes());

1. ServletContext对象
2. 概述

ServletContext对象代表着整个web应用，可以和程序的容器（服务器）进行通信。

1. 获取ServletContext对象
2. 通过request对象获取

request.getServletContext();

1. 通过HttpServlet获取

this.getServletContext()；

1. ServletContext对象的功能
2. 获取MIME类型

MIME类型：在互联网通信过程中定义的一种文件数据类型

格式：大类型/小类型

如：text/html ; image/jpeg

//获取ServletContext对象  
ServletContext context = this.getServletContext();  
//定义文件名称  
String filename = "a.jpg";  
//获取MIME类型  
String mimeType = context.getMimeType(filename);  
System.*out*.println(mimeType);

1. 域对象共享数据

通过ServletContext对象来设置参数，共享数据。用这个方式时要谨慎，因为ServletContext对象的范围是面向所有用户的所有请求数据，并不安全。

setAttribute(String name, Object value)

getAttribute(String name)

removeAttribute(String name)

1. 通过ServletContext对象获取文件服务器路径

通过ServletContext对象可以获取项目中文件的路径。

1. 文件在web目录下

当文件在web目录下时，直接在前面加上反斜杠获取路径:

String a = context.getRealPath(“/a.txt”);

1. 文件在web目录下的文件夹内

若文件在web目录下的WEB-INF文件夹中，则在反斜杠后再添加WEB-INF文件夹的名称即可：

String b = context.getRealPath(“/WEB-INF/b.txt”);

1. 文件在src目录下

若文件在src目录下，则需要到web目录下的WEB-INF文件夹中的classes文件夹中查找：

String c = context.getRealPath(“/WEB-INF/classes/c.txt”);

1. 文件下载实例
2. 需求分析
3. 在页面中显示超链接

超链接指向的资源如果能够被浏览器解析，则在浏览器中展示，如果不能解析，则弹出下载提示框。

1. 点击超链接后弹出下载提示框

通过设置响应头的资源打开方式为attachment即可：

content-disposition:attachment;filename=xxx

1. 点击下载
2. 实现步骤
3. 定义页面，编辑超链接href属性，将href指向一个Servlet对象，不论下载什么资源都使用这个Servlet对象，并传递资源的名称filename。

<a href=”/Response/downloadServlet?filename=1.jpg”>图片</a>

1. 实现Servlet
2. 获取href中资源文件的名称

String filename = req.getParameter("filename");

1. 获取资源文件的真实路径

ServletContext context = this.getServletContext();  
String realPath = context.getRealPath("/img/"+filename);

1. 使用字节输入流将资源文件加载到内存中

//用字节流关联，将文件字节读入到内存汇总  
FileInputStream fis = new FileInputStream(realPath);

1. 指定response对象的响应头

//设置response的响应头  
//设置响应头的文件类型 content-type  
String mimeType = context.getMimeType(filename);  
resp.setHeader("content-type", mimeType);

//设置响应头的打开方式 content-disposition  
resp.setHeader("content-disposition", "attachment;filename=" + filename);

1. 将数据写出到response输出流中

//将输入流的数据写出到输出流中  
//为response对象关联一个输出流  
ServletOutputStream sos = resp.getOutputStream();  
byte[] buff = new byte[1024 \* 8];  
int len = 0;  
while((len = fis.read(buff)) != 0){  
 sos.write(buff, 0, len);  
}

1. 解决中文编码问题
2. 获取user-agent请求头、

获取客户度使用的浏览器版本信息：

String agent= request.getHeader(“user-agent”);

1. 使用DownLoadUtils工具类方法编码文件名

根据不同的版本信息，设置filename的不同编码方式：

filename = DownLoadUtils.getFileName(agent, filename);