day28-ServletContext&Response

学习目标

1. 能够说出servlet生命周期方法执行流程
2. 能够使用servletcontext域对象
3. 能够使用Response对象操作HTTP响应内容
4. 能够完成文件下载案例
5. 能够处理响应乱码

Servlet进阶

HTTP（HyperTextTransferProtocol）即超文本传输协议，目前网页传输的通用协议。HTTP协议采用了请求/响应模型，浏览器或其他客户端发出请求，服务器给与响应。就整个网络资源传输而言，包括message-header和message-body两部分。首先传递message- header，即**http header**消息 。http header 消息通常被分为4个部分：general header, request header, response header, entity header。但是这种分法就理解而言，感觉界限不太明确。根据维基百科对http header内容的组织形式，大体分为Request和Response两部分。

Requests部分

Header	解释	示例
Accept	指定客户端能够接收的内容类型	Accept: text/plain, text/html
Accept-Charset	浏览器可以接受的字符编码集。	Accept-Charset: iso-8859-5
Accept-Encoding	指定浏览器可以支持的web服务器返回内容压缩编码类型。	Accept-Encoding: compress, gzip
Accept-Language	浏览器可接受的语言	Accept-Language: en,zh
Accept-Ranges	可以请求网页实体的一个或者多个子范围字段	Accept-Ranges: bytes
Authorization	HTTP授权的授权证书	Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Cache-Control	指定请求和响应遵循的缓存机制	Cache-Control: no-cache
Connection	表示是否需要持久连接。（HTTP 1.1默认进行持久连接）	Connection: close
Cookie	HTTP请求发送时，会把保存在该请求域名下的所有cookie值一起发送给web服务器。	Cookie: $Version=1; Skin=new;
Content-Length	请求的内容长度	Content-Length: 348
Content-Type	请求的与实体对应的MIME信息	Content-Type: application/x-www-form-urlencoded
Date	请求发送的日期和时间	Date: Tue, 15 Nov 2010 08:12:31 GMT
Expect	请求的特定的服务器行为	Expect: 100-continue
From	发出请求的用户的Email	From: user@email.com
Host	指定请求的服务器的域名和端口号	Host: www.zcmhi.com
If-Match	只有请求内容与实体相匹配才有效	If-Match: “737060cd8c284d8af7ad3082f209582d”
If-Modified-Since	如果请求的部分在指定时间之后被修改则请求成功，未被修改则返回304代码	If-Modified-Since: Sat, 29 Oct 2010 19:43:31 GMT
If-None-Match	如果内容未改变返回304代码，参数为服务器先前发送的Etag，与服务器回应的Etag比较判断是否改变	If-None-Match: “737060cd8c284d8af7ad3082f209582d”
If-Range	如果实体未改变，服务器发送客户端丢失的部分，否则发送整个实体。参数也为Etag	If-Range: “737060cd8c284d8af7ad3082f209582d”
If-Unmodified-Since	只在实体在指定时间之后未被修改才请求成功	If-Unmodified-Since: Sat, 29 Oct 2010 19:43:31 GMT
Max-Forwards	限制信息通过代理和网关传送的时间	Max-Forwards: 10
Pragma	用来包含实现特定的指令	Pragma: no-cache
Proxy-Authorization	连接到代理的授权证书	Proxy-Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Range	只请求实体的一部分，指定范围	Range: bytes=500-999
Referer	先前网页的地址，当前请求网页紧随其后,即来路	Referer: http://www.zcmhi.com/archives/71.html
TE	客户端愿意接受的传输编码，并通知服务器接受接受尾加头信息	TE: trailers,deflate;q=0.5
Upgrade	向服务器指定某种传输协议以便服务器进行转换（如果支持）	Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11
User-Agent	User-Agent的内容包含发出请求的用户信息	User-Agent: Mozilla/5.0 (Linux; X11)
Via	通知中间网关或代理服务器地址，通信协议	Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)
Warning	关于消息实体的警告信息	Warn: 199 Miscellaneous warning

Responses 部分

Header	解释	示例
Accept-Ranges	表明服务器是否支持指定范围请求及哪种类型的分段请求	Accept-Ranges: bytes
Age	从原始服务器到代理缓存形成的估算时间（以秒计，非负）	Age: 12
Allow	对某网络资源的有效的请求行为，不允许则返回405	Allow: GET, HEAD
Cache-Control	告诉所有的缓存机制是否可以缓存及哪种类型	Cache-Control: no-cache
Content-Encoding	web服务器支持的返回内容压缩编码类型。	Content-Encoding: gzip
Content-Language	响应体的语言	Content-Language: en,zh
Content-Length	响应体的长度	Content-Length: 348
Content-Location	请求资源可替代的备用的另一地址	Content-Location: /index.htm
Content-MD5	返回资源的MD5校验值	Content-MD5: Q2hlY2sgSW50ZWdyaXR5IQ==
Content-Range	在整个返回体中本部分的字节位置	Content-Range: bytes 21010-47021/47022
Content-Type	返回内容的MIME类型	Content-Type: text/html; charset=utf-8
Date	原始服务器消息发出的时间	Date: Tue, 15 Nov 2010 08:12:31 GMT
ETag	请求变量的实体标签的当前值	ETag: “737060cd8c284d8af7ad3082f209582d”
Expires	响应过期的日期和时间	Expires: Thu, 01 Dec 2010 16:00:00 GMT
Last-Modified	请求资源的最后修改时间	Last-Modified: Tue, 15 Nov 2010 12:45:26 GMT
Location	用来重定向接收方到非请求URL的位置来完成请求或标识新的资源	Location: http://www.zcmhi.com/archives/94.html
Pragma	包括实现特定的指令，它可应用到响应链上的任何接收方	Pragma: no-cache
Proxy-Authenticate	它指出认证方案和可应用到代理的该URL上的参数	Proxy-Authenticate: Basic
refresh	应用于重定向或一个新的资源被创造，在5秒之后重定向（由网景提出，被大部分浏览器支持）	Refresh: 5; url= http://www.zcmhi.com/archives/94.html

Retry-After	如果实体暂时不可取，通知客户端在指定时间之后再次尝试	Retry-After: 120
Server	web服务器软件名称	Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Set-Cookie	设置Http Cookie	Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1
Trailer	指出头域在分块传输编码的尾部存在	Trailer: Max-Forwards
Transfer-Encoding	文件传输编码	Transfer-Encoding:chunked
Vary	告诉下游代理是使用缓存响应还是从原始服务器请求	Vary: *
Via	告知代理客户端响应是通过哪里发送的	Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)
Warning	警告实体可能存在的问题	Warning: 199 Miscellaneous warning
WWW-Authenticate	表明客户端请求实体应该使用的授权方案	WWW-Authenticate: Basic
Content-Disposition	指示客户端下载文件	attachment;filename="filename.xls"

一,Servlet的生命周期【重点】 Servlet类

1 ,生命周期概述

一个对象从创建到销毁的过程

2 ,Servlet生命周期方法

servlet从创建到销毁的过程

出生：`init(ServletConfig config)`（初始化）用户第一次访问时执行。

活着：`service(ServletRequest req, ServletResponse res)`（服务）应用活着。每次访问都会执行。

死亡：`destroy()`（销毁）应用卸载。

3,Servlet生命周期描述

当客户端第一次请求的时候，会执行init方法,把Servlet创建出来

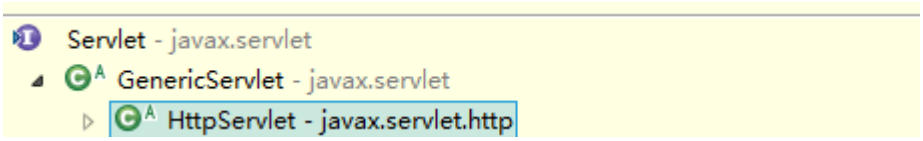
客户端任何一次请求会执行Service方法，

当servlet从服务器移除或者服务器正常关闭会执行destroy方法,销毁

servlet是单例(只有一个对象,init()调用一次), 多线程(服务器会针对每次请求获得一个线程处理这个请求)的, 在Servlet里面不要用全局变量(可能会导致线程不安全)

二,Servlet体系结构

idea中ctrl+h查看已知类的子类，ctrl+F12查看当前类的方法



在servlet中，真正执行程序逻辑的是service，对于servlet的初始化和销毁，由服务器调用执行，开发者本身不需要关心。因此，有没有一种更加简洁的方式来开发servlet程序呢？

在servlet接口规范下，官方推荐使用继承的方式，继承GenericServlet 或者HttpServlet来实现接口，那么我们接下来再去查看一下这两个类的API：

- **GenericServlet**

  GenericServlet类是抽象类,实现Servlet接口。HttpServlet也是抽象类,继承GenericServlet。

我们来使用GenericServlet 创建servlet：

```
1. 创建一个类
2. 继承GenericServlet
3. 重写service方法
```

- 实例代码xml和注解二选一:

  xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
           version="2.5">
<servlet>
    <servlet-name>Servlet</servlet-name>
    <servlet-class>javaweb.ServletGeneric</servlet-class>
</servlet>

    <servlet-mapping>
        <servlet-name>Servlet</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>
</web-app>
```

java

```
@WebServlet(name = "re",urlPatterns = "/ser")
public class ServletGeneric extends GenericServlet {
    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException {
        System.out.println("服务器运行");
    }
}
```

虽然，GenericServlet已经简化了servlet开发，但是我们平时开发程序需要按照一种互联网传输数据的协议来开发程序——http协议，因此，sun公司又专门提供了HttpServlet，来适配这种协议下的开发。

- **HttpServlet**

HttpServlet更专业更细致。

HttpServlet重写了service方法中并在方法中强转了Servlet的请求和响应为Http类型，

然后调用自己重载的service方法，req.getMethod()获得请求方式，判断如果get请求,调用doGet(),如果是post请求，调用doPost()

我们需要重写doGet、doPost等方法中一个即可，根据Http不同的请求，我们需要实现相应的方法。

我们来使用HttpServlet创建servlet：

```
1. 创建一个类
2. 继承HttpServlet
3. 重写doGet方法
```

xml

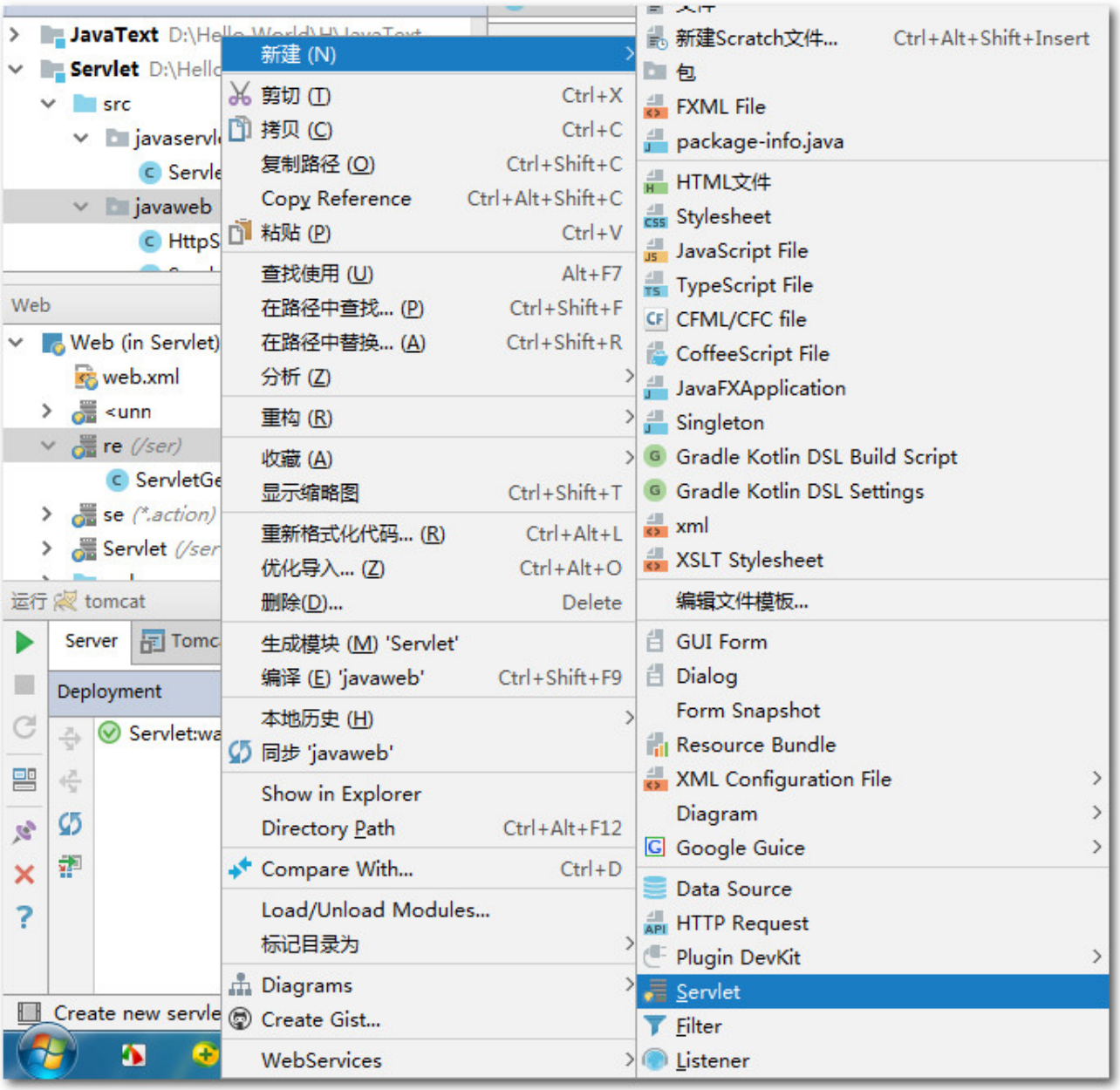
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
           version="2.5">
<servlet>
    <servlet-name>Servlet</servlet-name>
    <servlet-class>javaweb.HttpSer</servlet-class>
</servlet>

    <servlet-mapping>
        <servlet-name>Servlet</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>
</web-app>
```

java

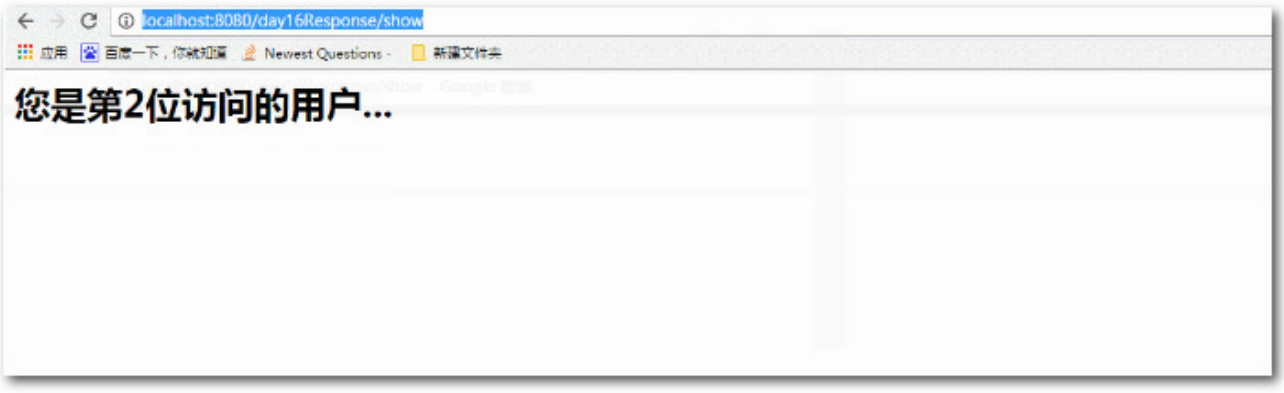
```
@WebServlet("/let2")
public class HttpSer extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println("Http演示");
    }
}
```

- 直接new Servlet



案例一：统计一下网站被访问的总次数

一，需求分析



- 在页面中显示您是第x位访问的用户.

二，技术分析

1.ServletContext概述 通过GenericServlet类获得

服务器为每一个应用都创建了一个ServletContext。ServletContext属于整个应用的，不局限于某个Servlet。 项目当做咱班43期, Servlet当做学生, ServletContext当做班主任.

2.作用

作为域对象存取数据 (当前案例使用这个功能)

获得文件mini类型（文件上传和下载）

获得全局初始化参数

获取web资源路径

3.获取ServletContext

HttpServlet的父类**GenericServlet**有个方法

`getServletContext()` 返回ServletContext

由于创建的类继承了HttpServlet,直接继承了GenericServlet,所以可以直接调用父类getServletContext()

4.作为域对象存取数据

范围: 在当前应用，使多个Servlet共享数据

- getAttribute(String name) ;向ServletContext对象的map取数据
- setAttribute(String name, Object object) ;从ServletContext对象的map中添加数据
- removeAttribute(String name) 根据name去移除数据

三，思路分析

- 有两个Servlet一个Servlet用于计数，另一个Servlet用于展示计数的结果
- 为什么不用一个Servlet来实现案例呢？因为count这个数不论是成员变量还是局部变量都会随类的消失而消失，无法保存，而ServletContext能持续保存
- 展示Servlet如何拿到计数Servlet的数据？通过ServletContext得到数据共享
- 计数的时候将数据存到ServletContext里面去，然后展示的时候可以从ServletContext取出
- 计数Servlet如何实现？在初始化的时候存0到ServletContext，然后每访问一次doGet()则count+1存到ServletContext里面去
- 展示Servlet,在doGet()里面读取count。

四，代码实现

xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
```

```
          http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
           version="2.5">
    <servlet>
        <servlet-name>ServletCount</servlet-name>
        <servlet-class>javaweb.ServletCon</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ServletShow</servlet-name>
        <servlet-class>javaweb.ServletSho</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ServletCount</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ServletShow</servlet-name>
        <url-pattern>/serv</url-pattern>
    </servlet-mapping>
</web-app>
```

计数Servlet

```
@WebServlet("/con")
public class ServletCon extends HttpServlet {
    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        ServletContext sc=getServletContext();
        sc.setAttribute("count",0);
    }

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        int count= (int) getServletContext().getAttribute("count");
        count++;
        getServletContext().setAttribute("count",count);
        response.getWriter().print("Wellcome...");
    }
}
```

展示Servlet

```
@WebServlet("/sho")
public class ServletSho extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
      //处理响应乱码
response.setContentType("text/html;charset=utf-8");
        int count= (int) getServletContext().getAttribute("count");
      response.getWriter().print(count);
    }
}
```

五，总结

1.servletContext

1.1作为域对象存取值【重点】

- getAttribute(String name) ;向ServletContext对象的map取数据
- setAttribute(String name, Object object) ;从ServletContext对象的map中添加数据
- removeAttribute(String name) ;根据name去移除数据

1.2获得文件mime类型（文件下载）

- getMimeType(String file)

```
@WebServlet("/mim")
public class mimeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
   doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        String name=getServletContext().getMimeType("a.mp3");
        System.out.println(name);
    }
}
```

1.3获得全局初始化参数【重点】 后面Spring会用到

- String getInitParameter(String name) ; //根据配置文件中的key得到value

  xml

```
    <context-param>
        <param-name>key</param-name>
        <param-value>myvalue</param-value>
    </context-param>
```

java

```
@WebServlet("/qu")
public class Para extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            String s=getServletContext().getInitParameter("key");
        System.out.println(s);
    }
}
```

1.4获取web资源路径【重点】

- String getRealPath(String path);根据资源名称的相对web路径得到资源的绝对路径.
- getResourceAsStream()获取流

下例中,已存在图片目录web/img/1.jpg

```
@WebServlet("/im")
public class imgServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    String path=getServletContext().getRealPath("/img/1.jpg");
    response.getWriter().print(path);
    }
}
```

案例二：完成文件下载

一需求分析

- 创建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

二，技术分析

1. HttpServletResponse概述

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响应消息分为响应行、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、响应头、响应体的方法。

2.操作响应三部分

2.1操作响应行

协议版本 ==状态码== 状态描述

void	**setStatus**(int sc) Sets the status code for this response.

常用的状态码：

200：成功

302：重定向

304：访问缓存

404：客户端错误

500：服务器错误

2.2操作响应头

一个key对应一个value

void	**setDateHeader**(String name, long date) Sets a response header with the given name and date-value.
void	**setHeader**(String name, String value) Sets a response header with the given name and value.
void	**setIntHeader**(String name, int value) Sets a response header with the given name and integer value.

一个key对应多个value

void	**addDateHeader**(String name, long date) Adds a response header with the given name and date-value.
void	**addHeader**(String name, String value) Adds a response header with the given name and value.
void	**addIntHeader**(String name, int value) Adds a response header with the given name and integer value.

常用的响应头 response.setHeader(String key, String value);

Refresh:定时跳转

Location:重定向

Content-Disposition:设置文件下载时候的头,告诉浏览器去下载内容

Content-Type：设置响应内容的MIME类型, 告诉浏览器需要内容的类型

2.3响应体

以下两个方法来自于父类**ServletResponse**

ServletOutputStream	**getOutputStream**() Returns a ServletOutputStream suitable for writing binary data in the response.
PrintWriter	**getWriter**() Returns a PrintWriter object that can send character text to the client.

3.文件下载

3.1什么是文件下载

将服务器上已经存在的文件,输出到客户端浏览器.

说白了就是把服务器端的文件拷贝一份到客户端, 文件的拷贝---> 流(输入流和输出流)的拷贝

3.2文件下载的方式

- 第一种:超链接方式（不推荐）

  链接的方式：直接将服务器上的文件的路径写到href属性中.如果浏览器不支持该格式文件,那么就会提示进行下载, 如果 浏览器支持这个格式(eg: png, jpg....)的文件,那么直接打开,不再下载了

```
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>你好啊，傻逼idea</div>
<a href="img/1.jpg">下载一</a>
<a href="img/WEB01.zip">下载二</a>
<img src="img/1.jpg"/>
</body>
</html>
```

- 第二种:手动编码方式（推荐）

  手动编写代码实现下载.无论浏览器是否识别该格式的文件,都会下载.

3.3手动编码方式要求

设置两个头和一个流

设置的两个头:

Content-Dispostion:浏览器识别该格式文件,提示浏览器下载.

Content-Type:文件类型.(MIME的类型)

设置一个流：

获得要下载的文件的输入流.

三，思路分析

1.超链接方式

- 创建资源下载页面。
- 设置超链接，href值设置资源的路径

2.编码方式

- 创建资源下载页面
- 设置超链接，把文件名提交到downloadServlet中，设置两头一流，进行下载

四，代码实现

html

```
<html lang="en">
<head>
    <meta charset="UTF-8">

    <title>Title</title>
```

```
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>你好啊，傻逼idea</div>
<a href="http://localhost/ds?fileName=1.jpg">下载一</a>
<br/>
<a href="http://localhost/ds?fileName=中国.zip">下载二</a>
<br/>
<img src="img/1.jpg"/>
</body>
</html>
```

java

```
@WebServlet("/ds")
public class Ds extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //获取请求参数
        String fileName=request.getParameter("fileName");
        System.out.println(fileName);
        InputStream is = getServletContext().getResourceAsStream("/img/" + fileName);
        //响应头，告诉浏览器下载
        response.setHeader("Content-Disposition","attachment;filename="+fileName);
        String file=getServletContext().getMimeType(fileName);
        response.setHeader("Content-Type",file);
        OutputStream osw=response.getOutputStream();
        IOUtils.copy(is,osw);
        osw.close();
        is.close();
    }
}
```

五，总结

1.下载中文的文件

中文文件在不同的浏览器中编码方式不同：火狐和谷歌是Base64编码,其它浏览器是URL编码

`setContentType()`和`设置服务器编码setCharacterEncoding`均来自于父类**ServletResponse**

html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>你好啊，傻逼idea</div>
<a href="http://localhost/down?fileName=1.jpg">下载一</a>
<br/>
<a href="http://localhost/down?fileName=中国.zip">下载二</a>
<br/>
<img src="img/1.jpg"/>
</body>
</html>
```

java

```
@WebServlet("/down")
public class DownServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //获取请求参数
        String fileName=request.getParameter("fileName");
        System.out.println(fileName);
        InputStream is = getServletContext().getResourceAsStream("/img/" + fileName);
        String agent=request.getHeader("User-Agent");
        System.out.println(agent);
        if(agent.contains("Chrome")||agent.contains("Firefox")){
            fileName=base64EncodeFileName(fileName);
        }else{
            URLEncoder.encode(fileName,"utf-8");
        }
        //响应头，告诉浏览器下载
        response.setHeader("Content-Disposition","attachment;filename="+fileName);
        String file=getServletContext().getMimeType(fileName);
        response.setHeader("Content-Type",file);
        OutputStream osw=response.getOutputStream();
        IOUtils.copy(is,osw);
        osw.close();
        is.close();
    }
    public static String base64EncodeFileName(String fileName) {
```

```
        Base64.Encoder encoder = Base64.getEncoder();
        try {
            return "=?UTF-8?B?"
                    + new String(encoder.encode(fileName
                    .getBytes("UTF-8"))) + "?=";
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    }
}
```

2 Response其他操作

2.1定时刷新

```
response.setHeader("refresh","秒数;url=跳转的路径"); //几秒之后跳转到指定的路径上
```

代码实现

html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>response定时刷新</div>
<a href="http://localhost/refl">5秒去博学谷</a><br/>
<img src="img/1.jpg"width="300px"/>
</body>
</html>
```

java

```
@WebServlet("/refl")
public class reflush extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setHeader("Refresh","5; url=http://ntlias-stu.boxuegu.com/#/login");
    }
}
```

2.2重定向【重点】

```
void sendRedirect(String location)
        Sends a temporary redirect response to the client using the specified redirect location URL.
```

简化

```
    response.setStatus(302);
    response.setHeader("Location","http://localhost/recho");
```

简化为一步实现跳转功能 `response.sendRedirect("http://localhost/recho");`

代码实现

html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>response重定向</div>
<a href="http://localhost/cho">去借钱</a><br/>
<img src="img/1.jpg"width="300px"/>
</body>
</html>
```

servlet借钱

```
@WebServlet("/cho")
```

```
public class cho extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
            doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println("我是张三，没钱你找王五吧");
        /*response.setStatus(302);
        response.setHeader("Location","http://localhost/recho");
       */
        response.sendRedirect("http://localhost/recho");
    }
}
```

servlet借到钱

```
@WebServlet("/recho")
public class recho extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        System.out.println("我借你钱");
        response.getWriter().print("Money are ready");
    }
}
```

2.3 向页面输出内容:

以下两个方法来自于父类**ServletResponse**

ServletOutputStream	**getOutputStream**() Returns a ServletOutputStream suitable for writing binary data in the response.
PrintWriter	**getWriter**() Returns a PrintWriter object that can send character text to the client.

页面输出只能使用其中的一个流实现,两个流是互斥的.

简化

```
        //设置服务器编码
        response.setCharacterEncoding("utf-8");
        //通过响应头告诉浏览器用utf-8解码
        response.setHeader("Content-Type"," text/html; charset=utf-8");
```

简化为一步 response.setContentType("text/html; charset=utf-8");

html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div{
            color:red;
        }
    </style>
</head>
<body>
<div>response乱码</div>
<a href="http://localhost/ma">中文乱码</a><br/>
<img src="img/1.jpg"width="300px"/>
</body>
</html>
```

java

```
@WebServlet("/ma")
public class Luanma extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        doGet(request,response);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

       /* //设置服务器编码
        response.setCharacterEncoding("utf-8");
        //通过响应头告诉浏览器用utf-8解码
        response.setHeader("Content-Type"," text/html; charset=utf-8");
*/
       response.setContentType("text/html; charset=utf-8");
        response.getWriter().print("中文");
    }
}
```