Response和ServletContext学习

学习目标

1.能够使用Response对象操作HTTP响应内容(掌握)

2.能够处理响应乱码 (掌握)

3.能够完成文件下载案例 (掌握，难点)

4.能够使用servletcontext域对象(掌握，掌握存值和取值和移出值，重点)

5.能够使用ServletContext获取资源文件的真实路径(最好是要掌握)

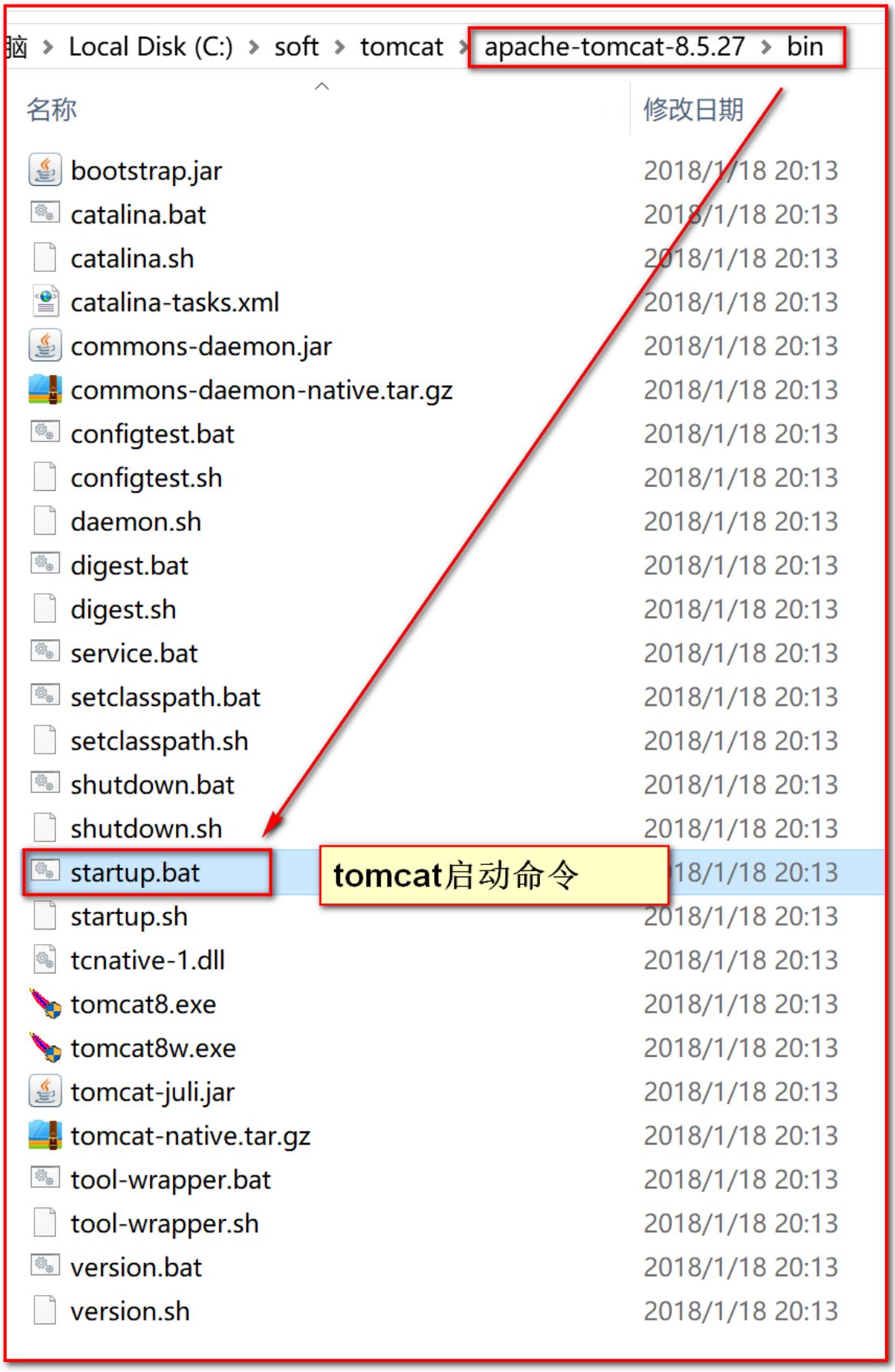
6.能够使用ServletContext将资源文件转换成字节输入流(需要掌握)

第1章 response对象操作

1.1 response对象的概念

1.1.1 什么是response对象

我们要学习使用response对象，先要了解它，我们先来看下官方文档截图：




```
总结：HttpServletResponse对象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。开发人员必须学会使用HttpServletResponse，才能给浏览器发送数据。
```

响应的组成部分:

1. 响应行(协议版本、响应状态码)

2. 响应头(键值对)
3. 响应体(显示在页面上的内容、下载的内容)

1.1.2 response常用API

1.1.2.1 API介绍

```
void setStatus(int sc) 设置此响应的状态代码(了解即可)
```

```
void setHeader(String name, String value)  用给定名称和值设置响应头(掌握)
```

1.1.2.3 使用步骤

1. 创建ResponseServlet
2. 调用setHeader，设置消息头（"Refresh"," 3;url=http://www.jd.com"）
3. 调用setStatus，设置响应状态码

1.1.2.4 演示代码

```
package cn.itcast.web;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ResponseServlet",urlPatterns = "/response")
public class ResponseServlet 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 {
    //自动刷新操作，在3秒后跳转京东主页
    response.setHeader("Refresh"," 3;url=http://www.jd.com");
    //设置响应状态码
    response.setStatus(200);
  }
}
```

学会了简单消息头设置，是远远不够的，响应消息头的内容非常的多，接下来向大家介绍几个，需要掌握的常见的响应头。

1.2 常见的响应头介绍

1. **location(比较重要)：**

   重定向操作：通常告知浏览器马上向该地址发送请求，通常和响应码302 一起使用

2. **content-encoding：(了解)**

设置当前数据的压缩格式，告知浏览器以何种压缩格式解压数据

3. **content-disposition：**

   告诉浏览器弹出一个下载的提示框

   通知浏览器以何种方式获取数据（直接解析数据（网页，图片文本），或者以附件方式（下载文件））

4. **content-type：**

   告诉客户端，服务器响应的数据的类型是什么。

   实体头部用于指示资源的MIME类型（MIME类型：用于提示当前文件的媒体类型，例如图片——（image/png）、音频——（audio/ogg）。它的作用与传统上Windows上的文件扩展名相同。该名称源于 最初用于电子邮件的MIME标准。）

```
注意：我们content-type常用的设置一般都是--“text/html;charset=utf-8”，其中“text/html;”--设置浏览器以文件格式解析数据；“charset=utf-8”--响应数据的编码表。
```

响应头虽然学习了，但是简单的介绍肯定记不住，因此，我们准了几个案例，让大家来练习。

1.3 案例1：使用location响应头实现跳转

1.3.1 使用location响应头实现重定向跳转页面

1.3.1.1 案例需求

使用location响应头实现重定向跳转淘宝主页www.taobao.com

1.3.1.2 案例效果

浏览器访问RedirectServlet之后跳转淘宝主页

1.3.1.3案例分析

1. 创建servlet
2. 使用response对象，发送location消息头和302响应码给浏览器

1.3.1.4 代码实现

```
package cn.itcast.web;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "RedirectServlet",urlPatterns = "/redirect")
public class RedirectServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

```

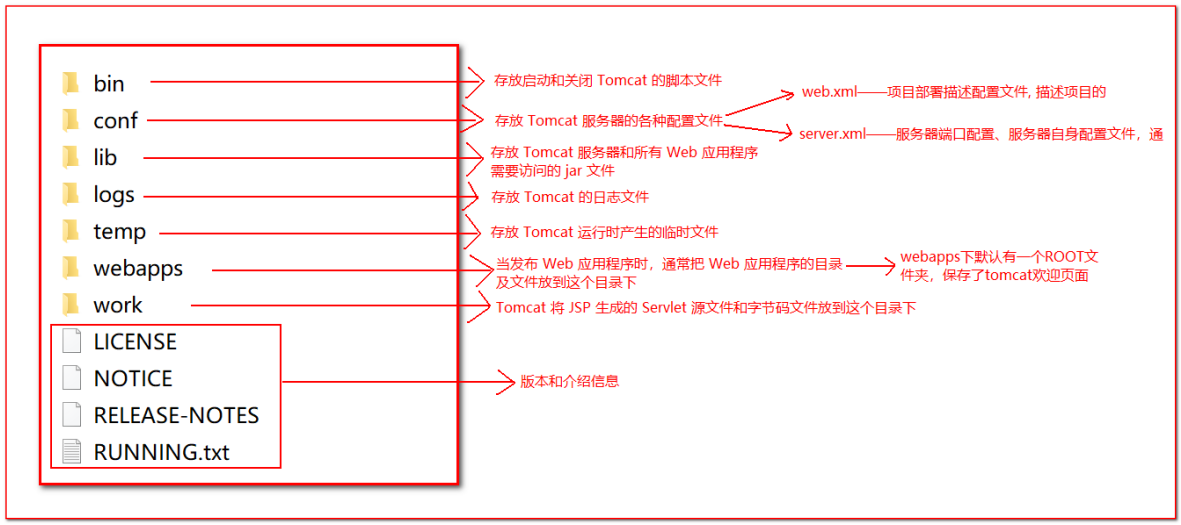
```
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //设置重定向响应头
       // response.setHeader("location","http://www.taobao.com");
        //设置状态码
        //response.setStatus(302);
        //============上面代码的简化方式=======================
        response.sendRedirect("http://www.taobao.com");
    }
}
```

3.4 案例2：使用Content-Type响应头向浏览器输出中文数据无乱码方案

3.4.1 案例需求

向页面输出中文数据没有乱码

3.4.2 案例效果



3.4.3 案例分析

1. 创建servlet
2. 使用response对象，调用setContentType方法传入参数：“text/html;charset=utf-8”)
3. 使用response对象，向页面输出“你好中文世界”

3.4.4 代码实现

```
package cn.itcast.web;

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

@WebServlet(name = "EncodingServlet",urlPatterns = "/encoding")
public class EncodingServlet 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.setContentType("text/html;charset=utf-8");
        response.getWriter().write("你好中文世界");
    }
}

```

3.5 案例3：以附件形式下载文件

3.5.1 案例需求

完成文件下载功能。

3.5.2 案例效果

用户点击页面的链接，浏览器开始下载文件。

3.5.3 案例分析

1. 创建一个页面，展示所有要被下载文件的链接
2. 链接将要下载的文件名称，发送给服务器的servlet，让servlet进行处理
3. 服务器加载文件资源
4. 提示浏览器，以下载的方式，获取服务器资源
5. 使用IO的方式，将文件数据输出到浏览器（response.getOutputStream();）

3.5.4 代码实现

1. html页面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文件下载案例</title>
</head>
<body>
    <h1>非常简单地实现文件的下载</h1>
    <a href="dwl/1.jpg">1.jpg</a><br>
    <a href="dwl/2.jpg">2.jpg</a><br>
    <a href="dwl/3.txt">3.txt</a><br>
    <a href="dwl/heidisql.rar">heidisql.rar</a><br>
    <a href="dwl/传智播客.txt">传智播客</a><br>
    <a href="demo01.html">传智首页</a>

```

```
    <h1>自己使用流向客户端输出内容</h1>
    <a href="download?fileName=1.jpg">1.jpg</a><br>
    <a href="download?fileName=2.jpg">2.jpg</a><br>
    <a href="download?fileName=3.txt">3.txt</a><br>
    <a href="download?fileName=heidisql.rar">heidisql.rar</a><br>
    <a href="download?fileName=传智播客.txt">传智播客</a><br>
</body>
</html>
```

2. servlet演示代码

```
@WebServlet(name = "DownloadServlet",urlPatterns = "/download")
public class DownloadServlet 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 {
        //在这里就要写文件下载的代码
        System.out.println("收到了一个请求...");

        //刚刚在客户端携带过来的文件名就是请求参数，请求参数就封装request对象中
        //服务器就要获取客户端携带过来的请求参数,,请求参数获取都是String类型
        String fileName = request.getParameter("fileName");
        System.out.println(fileName);

        //在文件下载之前，要先告诉客户端，请弹出一个文件下载的提示框
        //通过设置一个名为“Content-Disposition”的响应头
        //响应头和请求头中都不能出现中
        //将中文进行编码,使用URLEncoder进行编码
        //获取到这个文件名之后，要将这个文件名拼接成文件路径
        String path =
"C:/JavaEE_Relation/JavaEE39/itheima39_web/out/artifacts/day28_ResponseDemo_01_war_exploded/dwl/"+fileName;

        //URLEncoder编码的内容之后URLDecoder能解码，除了火狐浏览器之外的其它浏览器都是使用的URLDecoder进行解码的
        //火狐是使用的Base64进行解码的，所以针对火狐浏览器要特殊处理，使用Base64进行编码

        //怎么判断是不是火狐，根据请求头"User-agent",获取一个名为"User-agent"的请求头
        String agent = request.getHeader("User-Agent");

        if(!agent.contains("Firefox")){
            //%E4%BC%A0%E6%99%BA%E6%92%AD%E5%AE%A2.txt
            fileName = URLEncoder.encode(fileName,"UTF-8");
            System.out.println("编码后的fileName=" + fileName);
        }else {
            //使用Base64进行编码
            fileName = base64Encode(fileName);
        }
        //放在header中的应该是编码过后的字符串
```

```
38          response.setHeader("Content-
Disposition","attachment;filename="+fileName);
39
40
41
42          //将文件转换成字节输入流(InputStream)
43          InputStream in = new FileInputStream(path);
44          //已经将文件转换成字节输入流
45
46          /*//接下来就要使用字节输出流将这个文件输出到客户端
47          //定义一个字节数组buffer
48          byte[] buffer = new byte[1024];
49
50          //获取一个字节输出流，这个输出流应该能将字节输出到客户端
51          ServletOutputStream out = response.getOutputStream();
52
53          //定义一个标示，用来判断是否读写完了
54          int len = 0;
55
56          while((len = in.read(buffer)) != -1){
57              //上述操作已经将in中的字节读取到buffer中
58              //接下来就要将buffer中的使用out来写出去
59              out.write(buffer,0,len);
60          }
61          out.close();
62          in.close();*/
63
64          //使用jar包方法完成读写
65          ServletOutputStream out = response.getOutputStream();
66          IOUtils.copy(in,out);
67
68          out.close();
69          in.close();
70      }
71
72      //使用Base64对中文字符串进行编码
73      private String base64Encode(String fileName) throws
UnsupportedEncodingException {
74          Base64.Encoder encoder = Base64.getEncoder();
75          fileName = encoder.encodeToString(fileName.getBytes("UTF-8"));
76          fileName = "=?utf-8?B?" + fileName + "?=";
77          return fileName;
78      }
79  }
```

Response的小结

1. response的概念:服务器发送给客户端的响应内容，我们通过response对象的方法，**设置响应内容**，然后由服务器将其发送给客户端
2. setStatus(状态码)方法，设置响应状态码(重要)
3. setHeader(name,value),设置响应头
4. 常见的响应头:Refresh,location,Content-Type,Content-Disposation
5. 发起重定向:response.SendRedirect("路径")(重要)
6. 解决响应的中文乱码问题:response.setContentType("text/html;charset=UTF-8")(重要)
7. 获取响应的字符输出流:response.getWriter()(重要)
8. 获取响应的字节输出流:response.getOutputStream()(下载的时候使用)

第2章 servletContext

2.1 ServletContext的概述

ServletContext是一个容器（域对象）可以存储键值对数据（String key,Object value），保存在ServletContext中的数据不仅可以提供给所有的servlet使用，而且可以在整个项目范围内使用（后期的过滤器、监听器也可以使用ServletContext）。

服务器会在启动的时候，就为它所部署的每一个项目都创建一个ServletContext实例对象，并且每个项目中有且只有一个ServletContext实例对象。

2.2 ServletContext作为域对象

域对象:有一定的作用域，在该作用域中可以进行数据共享的对象。

ServletContext中定义了很多方法，在javaweb阶段我最常用的就是将ServletContext作为容器（域对象）使用，因此，接下来我们要学习这个容器（域对象的）API。

2.2.1 API介绍

GenericServlet：

```
1.  ServletContext getServletContext()
    获取ServletContext对象
```

ServletContext：

```
1.  void setAttribute(String name, Object object)
    往servletcontext容器中存入数据，name为数据名称，object为数据的值
2.  Object getAttribute(String name)
    从ServletContext中获取数据，根据指定的数据名称
3.  void removeAttribute(String name)
    从ServletContext中移除数据，根据指定的数据名称
```

2.2.2 使用步骤

1. 创建ServletContextServlet1和ServletContextServlet2
2. ServletContextServlet1调用存方法
3. ServletContextServlet2调用取方法
4. ServletContextServlet2调用删方法
5. ServletContextServlet2调用取方法

2.2.4 演示代码

ServletContextServlet1：

```
package cn.itcast.web;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```java
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ServletContextServlet1" ,urlPatterns = "/context1")
public class ServletContextServlet1 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 {
        //获取容器
        ServletContext context = getServletContext();
        //存入数据
        context.setAttribute("addr","广州");
    }
}

```

ServletContextServlet2：

```java
package cn.itcast.web;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ServletContextServlet2" ,urlPatterns = "/context2")
public class ServletContextServlet2 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 {
        //获取容器
        ServletContext context = getServletContext();
        //获取数据
        String addr = (String) context.getAttribute("addr");
        System.out.println("存入之后，获取数据："+addr);
        //移除数据
        context.removeAttribute("addr");
        //重新获取数据
        String addr2 = (String) context.getAttribute("addr");
        System.out.println("移除之后，获取数据："+addr2);

    }
}

```

代码准备好之后，分别访问context1和context2，我们会得到一下结果：

```
存入之后，获取数据：广州
移除之后，获取数据：null
```

通过这个结果，我们发现两次请求虽然访问了不同的servlet，但是依然能通过ServletContext共享数据，而且即使是由一个同学先访问，然后由另一个同学访问也是同样的结果。

```
总结：保存在ServletContext中的数据是项目全局共享的数据。
```

2.3ServletContext获取资源文件的真实路径

如果某个文件直接在web文件夹中，那么它会直接显示在发布后的项目路径下。如果要将这类文件转换成流，使用ServletContext对象的getResourceAsStream()方法

如果某个文件在resource文件夹中，那么它会显示在发布后的项目路径下的"WEB-INF/classes"中，如果要将这类文件转换成流使用"类加载器"的getResourceAsStream()方法

servletContext对象调用getRealPath("路径")方法，可以获得文件的真实路径 getRealPath()获取的默认路径是tomcat安装路径下的webApps目录下的项目路径，然后找到该项目的WEB-INF目录，里面有个classes文件夹，src下的文件变异后全部放在该文件夹中。

代码演示

```
public class ServletContextDemo05 extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @SuppressWarnings("resource")
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //读取到info.properties配置文件中的数据
        //1.创建对象
        Properties properties = new Properties();
        //2.将资源文件转换成流
        //使用ServletContext对象获取资源文件的真实路径,默认能获取项目路径
        ServletContext servletContext = getServletContext();
        String realPath = servletContext.getRealPath("/WEB-INF/classes/info.properties");
        System.out.println(realPath);

        InputStream inStream = new FileInputStream(realPath);
        //3.使用properties对象加载流
        properties.load(inStream);
        //4.调用properties对象的getProperty()方法
        String username = properties.getProperty("username");
        System.out.println(username);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

2.4ServletContext将资源文件转换成字节输出流

getResourceAsStream("路径") 该方法类似classLoader的getResourceAsStream("路径")方法，但是该文件路径注意， 必须写"/WEB-INF/classes/",因为它默认只能找到项目的根路径

```
public class ServletContextDemo06 extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //将资源文件转换成流:1.FileInputStream、2.类加载器的getResourceAsStream、3.ServletContext的getResourceAsStream方法
        //1.获取ServletContext对象
        ServletContext servletContext = getServletContext();
        //2.调用getResourceAsStream()方法
        InputStream inputStream = servletContext.getResourceAsStream("WEB-INF/classes/info.properties");

        Properties properties = new Properties();
        //使用包装流，将inputStream转换成Reader，将字节流转换成字符流，可以指定字符集
        InputStreamReader reader = new InputStreamReader(inputStream,"UTF-8");

        properties.load(reader);

        //tomcat中使用的字符集是ISO-8859-1
        String username = properties.getProperty("username");
        System.out.println(username);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

}
```

2.3 ServletContext作用总结

概念:首先是一个容器，以键值对的形式进行数据存储。它会在服务器启动的时候，创建对象，并且每一个项目只有一个ServletContext实例对象。

作用域:整个项目

ServletContext对象一共三个作用:

1. **可以读取资源在当前项目中的文件位置，获取真实路径(web下面的文件的真实路径),getRealPath()**
2. **可以作为域对象在项目全局范围内提供共享数据**
    1. setAttribute(name,value)，存数据
    2. removeAttribute(name),移除数据
    3. getAttribute(name),获取数据
3. **将资源文件转换成流(web下面的文件),getResourceAsStream()**