# 第五十讲Servlet

## 一、ServletContext

### （一）概念

ServletContext是一个接口，表示Servlet上下文对象，对象的创建和销毁由服务器管理。

**服务器启动时创建，服务器关闭时销毁。服务器创建的ServletContext对象是单例的。**

### 获取方式

1. request.getServletContext();

2. this.getServletContext();

HttpServletRequest:

package com.offcn.servlet;

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("/Demo01Servlet")

public class Demo01Servlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//request作用域中添加内容

req.setAttribute("classAddress","6-106");

//转发

req.getRequestDispatcher("/Demo02Servlet").forward(req,resp);

}

}

package com.offcn.servlet;

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;

import java.io.PrintWriter;

@WebServlet("/Demo02Servlet")

public class Demo02Servlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//request作用域取内容

String classAddress = (String)req.getAttribute("classAddress");

//展示到页面上

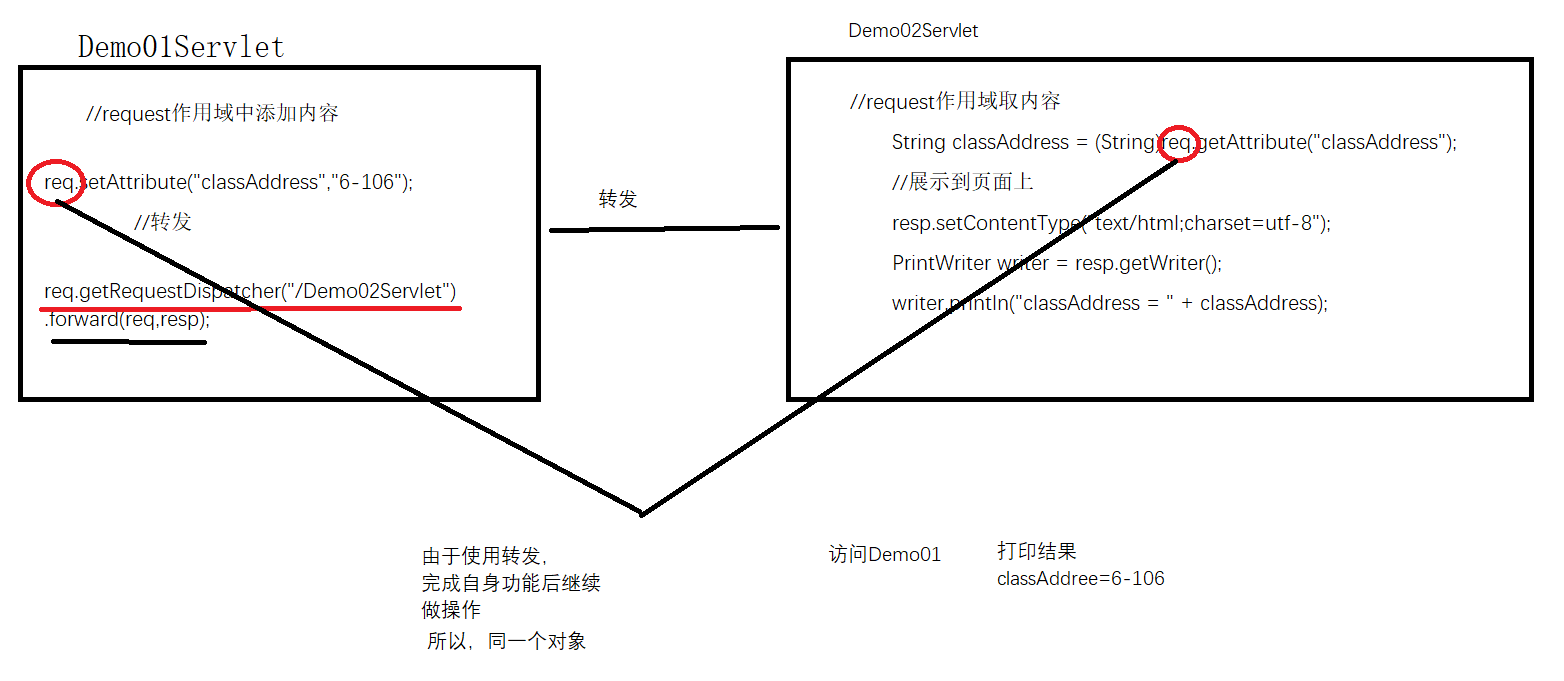
resp.setContentType("text/html;charset=utf-8");

PrintWriter writer = resp.getWriter();

writer.println("classAddress = " + classAddress);

}

}



HttpServletRequest: ServletContext:

package com.offcn.servlet;

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("/Demo01Servlet")

public class Demo01Servlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//request作用域中添加内容

req.setAttribute("classAddress","6-106");

// ServletContext创建对象添加

ServletContext sc = this.getServletContext();

sc.setAttribute("className","1122java");

}

}

package com.offcn.servlet;

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;

import java.io.PrintWriter;

@WebServlet("/Demo02Servlet")

public class Demo02Servlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

//request作用域取内容

String classAddress = (String)req.getAttribute("classAddress");

//sc对象

ServletContext sc = this.getServletContext();

String className = (String)sc.getAttribute("className");

//展示到页面上

resp.setContentType("text/html;charset=utf-8");

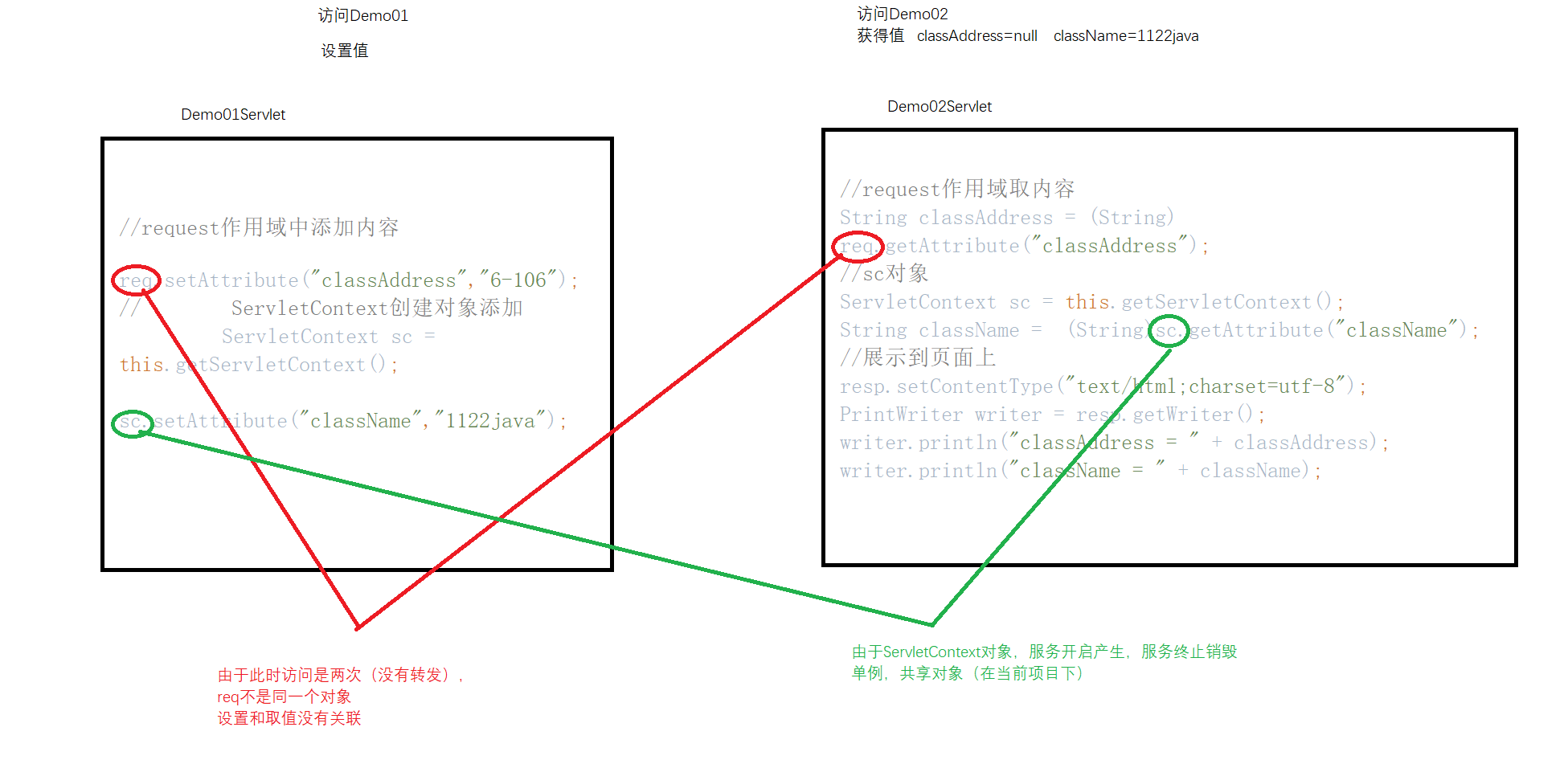
PrintWriter writer = resp.getWriter();

writer.println("classAddress = " + classAddress); //null

writer.println("className = " + className); //1122java

}

}



### （三）作用

#### 1、作为全局容器

ServletContext与request一样，可以作为作用域使用。而且，由于在同一个工程中的所有Servlet都共享同一个ServletContext对象，因为ServletContext作为作用域的取值范围更广，可以作为全局容器。

#### 2、获得文件的真实路径

获取文件的真实路径方法:

getRealPath("web 下的资源路径"); getRealPath(“source/face5.jpg”)

获取资源路径并转化成流：

InputStream is = 对象.getResourceAsStream("source/a.jpg");

//获得真实路径

String realPath = sc.getRealPath("/Demo01Servlet");

System.out.println(realPath);



#### 3、获取服务器信息

getServerInfo();

//获得服务器信息

String serverInfo = sc.getServerInfo();

System.out.println(serverInfo);



## 二、Servlet响应response

### （一）概述

请求和响应是生死相依的一对恋人，永远也不分离。Servlet的service方法（也可能是doGet或doPost方法）中request是请求对象，而response是响应对象。

http协议的响应分为 响应行、响应头和响应体三个部分。

响应行描述的是：协议版本 响应状态码和状态码描述等信息。

响应头可以控制给客户端产生的响应的特点和状态等。

响应体通常是响应给客户端浏览器的内容。

### 响应头

我们可以通过设置响应头来设置给客户端产生的响应的特征和状态。

设置响应头方法描述

|  |  |
| --- | --- |
| **方法** | **描述** |
| setHeader(String key,String value) | 设置响应头信息 |
| addHeader(String key,String value) | 设置响应头信息-设置重复值（一个key包含多个value的情况） |
| **setContentType(String str)** | **设置响应类型**  **例如：**  **response.setContentType(“text/html;charset=utf-8”)** |

// 设置3秒后自动跳转到百度

resp.setHeader("refresh","3;url=http://www.baidu.com");

// 设置响应的类型和编码方式

// 设置响应头类型

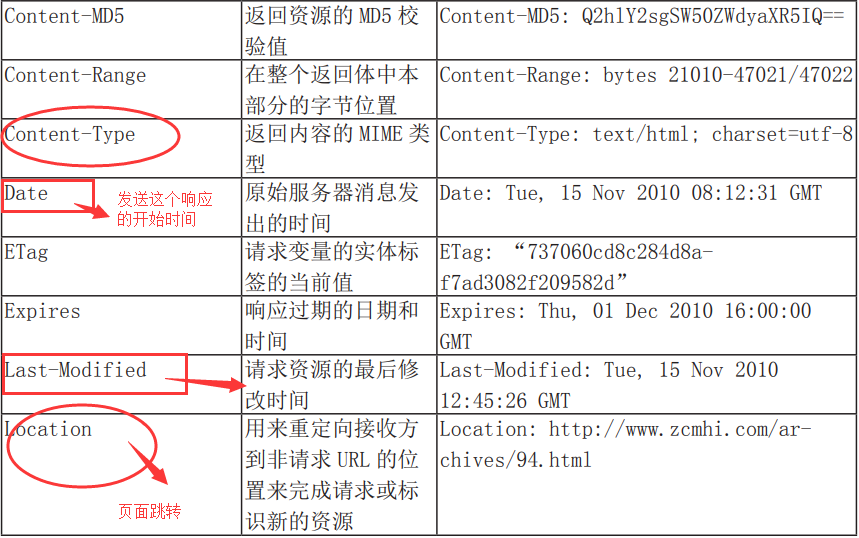
resp.setContentType("text/html;charset=utf-8");

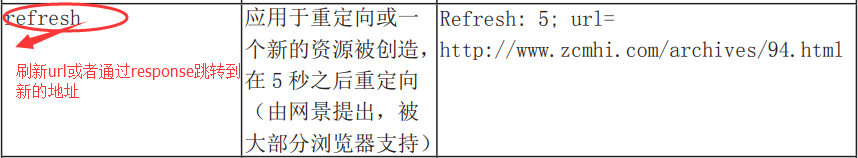
// 设置响应头

resp.setHeader("content-encoding","gzip");

**响应头大全（作为了解即可）**







### 响应体

可以通过response对象获取输出流来指定响应体内容(不可同时使用)

PrintWriter out = response.getWriter(); // 获取字符流

或

OutputStream os = response.getOutputStream();//获取字节流

用字节流或字符流输出的内容将作为响应体响应给客户端浏览器

PrintWriter writer = resp.getWriter();

// writer.println("hello");

ServletOutputStream outputStream = resp.getOutputStream();

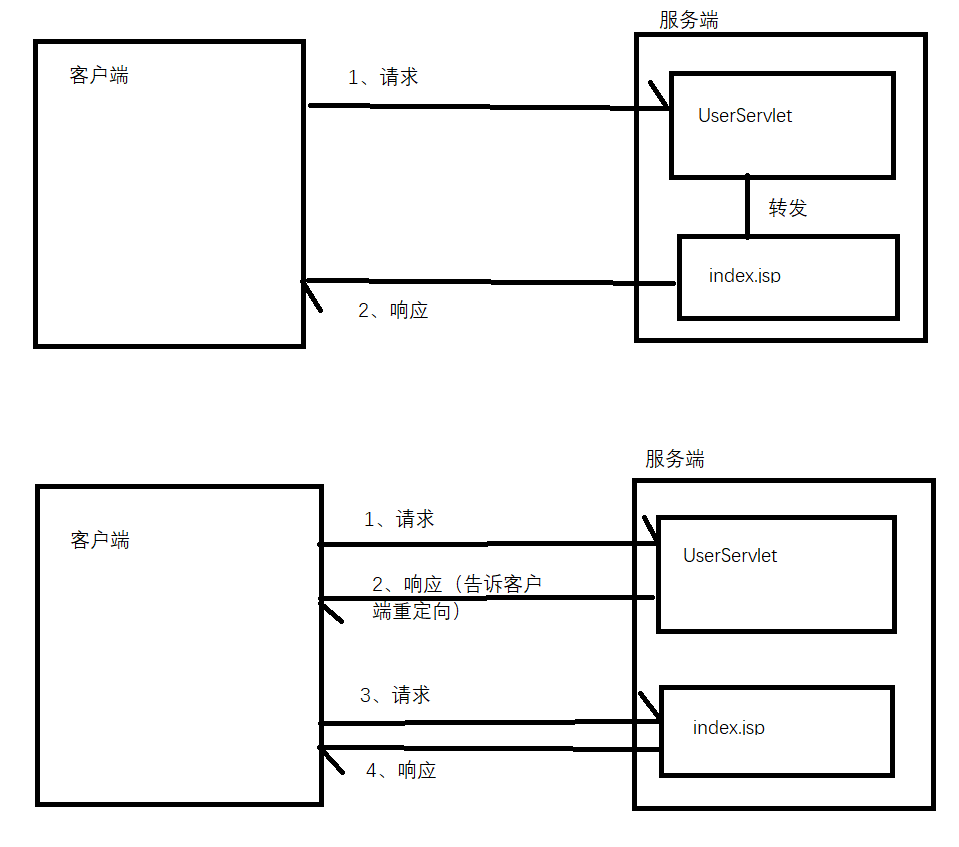
outputStream.println("world");

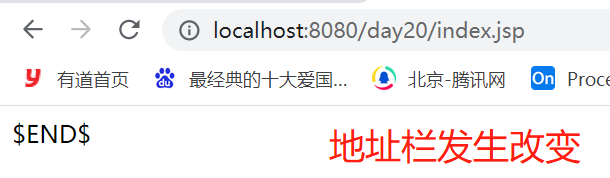
### 重定向

重定向(Redirect)就是通过各种方法将各种网络请求重新定个方向转到其它位置。

修改RegisterServlet

response.sendRedirect("index.jsp");





### 转发和重定向的区别

1、发生位置不同，转发是在服务器行为，重定向是客户端行为

2、转发浏览器地址栏不发生改变，重定向地址栏变

3、转发是一次请求一次响应，重定向至少是两次请求两次响应

4、转发时 request 作用域的数据可共享，而重定向时 request 作用域的数据不能共享

5、跳转语句不同：

（1）转发：request.getRequestDispatcher("url").forward(request, response);

（2）重定向：response.sendRedirect("url");

6、转发速度要高于重定向,因为是在服务器内部完成

7、转发的 url 一定是本 web 应用范围内，而重定向可以是任意 url。

### （六）案例：文件下载

#### 1、超链接直接指向文件的下载方式

可以在页面上使用超链接直接指向要下载的文件完成文件下载。但这种方式有缺陷：当浏览器能直接打开这个文件时，将不会出现文件下载的效果。

<body>

<h1>超链接直接指向文件的下载方式</h1>

<p><a href="download/a.txt">a.txt</a></p>

<p><a href="download/day49-Servlet请求-request.docx">day49-Servlet请求-request.docx</a></p>

<p><a href="download/mysql-connector-java-8.0.26.jar">mysql-connector-java-8.0.26.jar</a></p>

<p><a href="download/xyy.webp">xyy.webp</a></p>

<p><a href="download/笔记.rar">笔记.rar</a></p>

</body>

#### 通过Servlet响应的下载方式

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<h1>超链接直接指向文件的下载方式</h1>

<p><a href="download/a.txt">a.txt</a></p>

<p><a href="download/day49-Servlet请求-request.docx">day49-Servlet请求-request.docx</a></p>

<p><a href="download/mysql-connector-java-8.0.26.jar">mysql-connector-java-8.0.26.jar</a></p>

<p><a href="download/xyy.webp">xyy.webp</a></p>

<p><a href="download/笔记.rar">笔记.rar</a></p>

<h1>Servlet进行文件的下载</h1>

<p><a href="downloadServlet?fileName=a.txt">a.txt</a></p>

<p><a href="downloadServlet?fileName=day49-Servlet请求-request.docx">day49-Servlet请求-request.docx</a></p>

<p><a href="downloadServlet?fileName=mysql-connector-java-8.0.26.jar">mysql-connector-java-8.0.26.jar</a></p>

<p><a href="downloadServlet?fileName=xyy.webp">xyy.webp</a></p>

<p><a href="downloadServlet?fileName=笔记.rar">笔记.rar</a></p>

</body>

</html>

package com.offcn.servlet;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

@WebServlet("/downloadServlet")

public class downloadServlet extends HttpServlet {

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

}

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

//1、获得参数 文件名称

String fileName = request.getParameter("fileName");

//2、设置响应头

response.setHeader("Content-Disposition","attachment;filename=" + fileName);

//3、创建文件输入流

InputStream is = this.getServletContext().getResourceAsStream("download/" + fileName);

//4、创建文件输出流

OutputStream os = response.getOutputStream();

//5、文件处理 读写

byte[] data = new byte[1024\*8];

int len = -1; //字节个数

while((len = is.read(data)) != -1){

os.write(data,0,len);

}

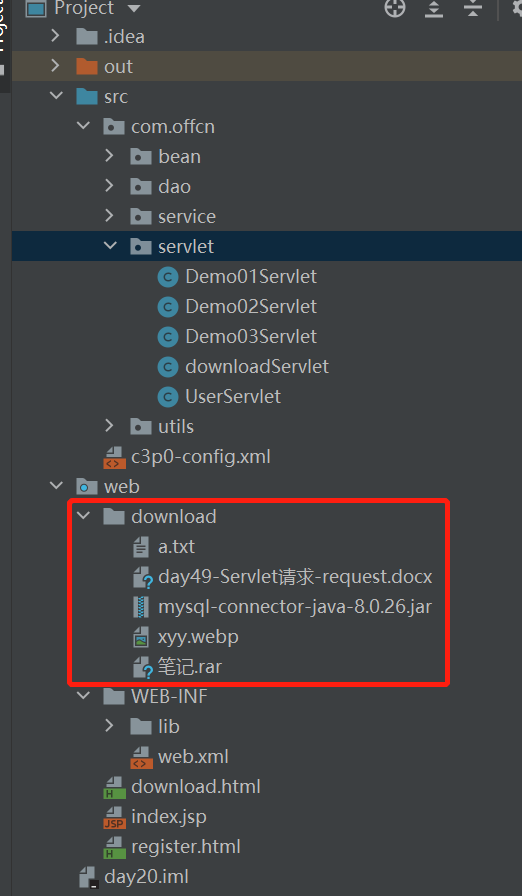
//6、关流

is.close();

os.close();

}

}



## 一、JSP

### JSP简介

JSP（全称JavaServer Pages）是由[Sun](https://baike.baidu.com/item/Sun%20Microsystems) 公司主导创建的一种动态网页技术标准。JSP部署于服务器上，可以响应客户端发送的请求，并根据请求内容动态地生成[HTML](https://baike.baidu.com/item/HTML)、[XML](https://baike.baidu.com/item/XML)或其他格式文档的[Web](https://baike.baidu.com/item/Web)网页，然后返回给请求者。JSP技术以[Java](https://baike.baidu.com/item/Java)语言作为[脚本语言](https://baike.baidu.com/item/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80)，为用户的[HTTP](https://baike.baidu.com/item/HTTP)请求提供服务，并能与服务器上的其它Java程序共同处理复杂的业务需求。

静态网页：可以直接被浏览器渲染。但是不能做数据交互。

动态网页：不可以被浏览器直接渲染，需要第三方辅助（服务器），可以与Java进行数据交互操作。

#### 1、JSP特点

JSP≈HTML+Java代码。

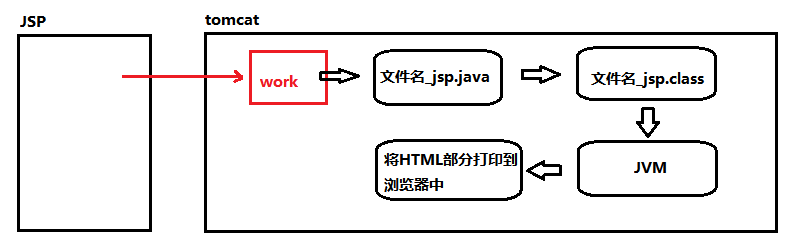
JSP是动态资源，存放的位置、编写和访问上很像静态资源。

#### 2、JSP的运行原理

JSP本质上就是Servlet。

JSP如何运行？

JSP在第一次访问时，会被服务器中的jsp引擎（容器中编译jsp的软件模块）编译成一个Servlet ：jsp名字\_jsp.java、jsp名字\_jsp.class，并且放在 Tomcat的work目录下。



**使用IntelliJ Idea，JSP编译后并没有放在Tomcat的work下（即使把项目发布到Tomcat的webapps中），而是存放在：**

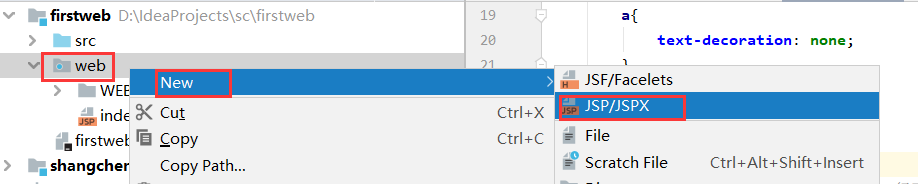
**C:\Users\自己的计算机名\**

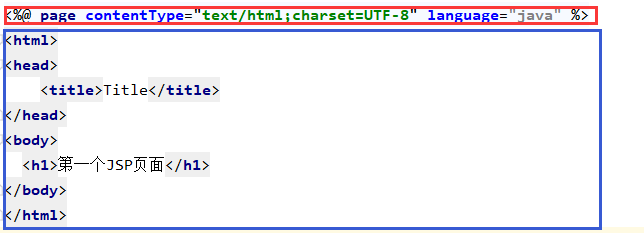
**AppData\Local\JetBrains\IntelliJIdea2020.1\tomcat\\_Default\_(Template)\_Project\work\Catalina\localhost\myweb\org\apache\jsp**

### （二）JSP编写

#### 1、JSP文件创建

和静态资源 HTML、CSS、JS文件等相同，放在工程的web目录下。





#### 2、JSP页面访问

和HTML页面访问方式相同，在浏览器地址栏直接输入访问即可。

#### 3、JSP语法

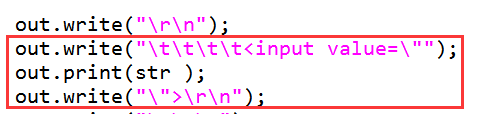
1. 代码片段(内联代码)：

<% Java代码 %>

1. 脚本表达式：

<%= 值 %>

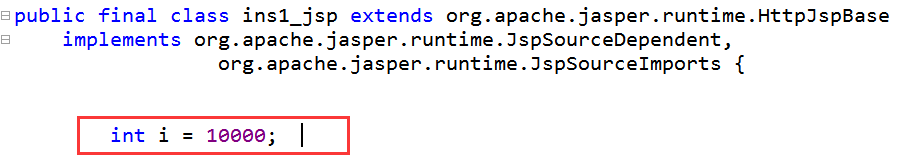
意思就是将Java的内容输出到页面上，等同 PrintWriter。



1. JSP的声明（一般不用）

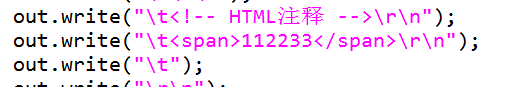
语法：<%！java声明 %>

解释：对应Java就是全局变量。



1. JSp注释

语法：<%-- 注释代码 --%>



在JSp中HTML的注释和JSP的注释都可以使用。

区别是：

HTML注释最终会原样输出到HTML代码上，不丢失。

**JSP注释在Tomcat转化Java类的时候直接给去掉，为了能够让字节码文件更简洁，运行更速度。在客户端中是不可见的。**

两种方式自行选择。

<%--

Created by IntelliJ IDEA.

User: 13578

Date: 2022/4/14

Time: 15:55

To change this template use File | Settings | File Templates.

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ page import="java.util.List" %>

<html>

<head>

<title>jsp</title>

</head>

<body>

<h1>这是第一个jsp</h1>

<%-- jsp注释 --%>

<%-- java书写 设置 --%>

<%

int x = 1;

%>

<%-- 打印输出内容到页面 --%>

<%=x%>

<%--打印10遍某句话 标签 <h1>明天考试</h1> --%>

<%

for (int i = 0; i < 10; i++) {

%>

<h1>这是第一个jsp</h1>

<%

}

%>

</body>

</html>

作业：

1、文件下载