JavaWEB

# JavaWEB基础知识

## 基础概念

* WEB，在英语中web即表示网页的意思，它用于表示Internet主机上供外界访问的资源。
* Internet上供外界访问的Web资源分为：
  + 静态web资源（如html 页面）：指web页面中供人们浏览的数据始终是不变。
  + 动态web资源：指web页面中供人们浏览的数据是由程序产生的，不同时间点访问web页面看到的内容各不相同。
* 静态web资源开发技术
  + Html
* 常用动态web资源开发技术：
  + JSP/Servlet、ASP、PHP等
  + 在Java中，动态web资源开发技术统称为Javaweb，我们课程的重点也是教大家如何使用Java技术开发动态的web资源，即动态web页面。

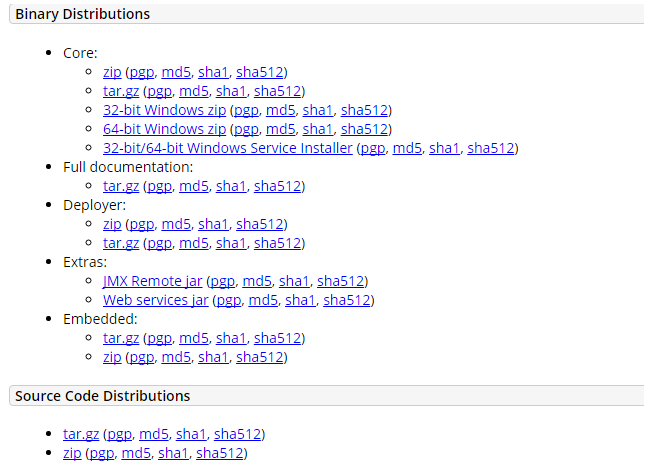
## 常见的web服务器

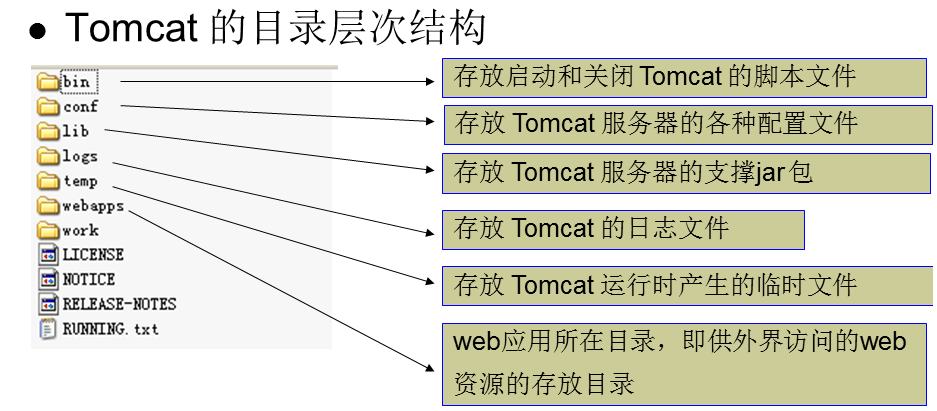
* WebLogic是BEA公司的产品，是目前应用最广泛的Web服务器，支持J2EE规范，而且不断的完善以适应新的开发要求，启动界面如图
* 另一个常用的Web服务器是IBM公司的WebSphere，支持J2EE规范，启动界面如图
* **在小型的应用系统或者有特殊需要的系统中，可以使用一个 免费的Web服务器：Tomcat，该服务器支持全部JSP以及Servlet规范，启动界面如图**

## 下载和配置Tomcat

**Tomcat官方站点**：https://tomcat.apache.org/download-80.cgi

* **获取Tomcat安装程序包**
  + tar.gz文件是Linux操作系统下的安装版本
  + exe文件是Windows系统下的安装版本
  + zip文件是Windows系统下的压缩版本
* **安装Tomcat**



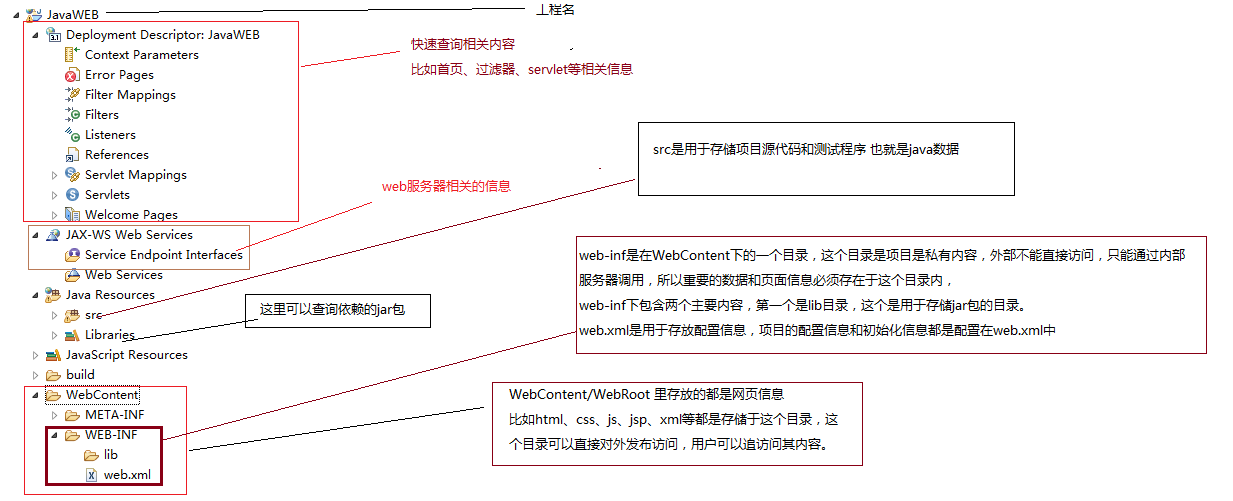


Eclipseee中配置tomcat

# Servlet开发

* Servlet是sun公司提供的一门用于开发动态web资源的技术。
* Sun公司在其API中提供了一个servlet接口，用户若想用发一个动态web资源(即开发一个Java程序向浏览器输出数据)，需要完成以下2个步骤：
  + 编写一个Java类，实现servlet接口。
  + 把开发好的Java类部署到web服务器中。
* 快速入门，用servlet向浏览器输出“hello servlet”。
  + 阅读Servlet API，解决两个问题：
  + 输出hello servlet的java代码应该写在servlet的哪个方法内？
  + 如何向IE浏览器输出数据？

## Web的目录结构



## Servlet入门基础

**方式一： 使用配置文件的方式配置servlet**

**package** com.guigu.base;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**public** **class** HelloServlet **extends** HttpServlet {

@Override

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

doGet(req, resp);

}

@Override

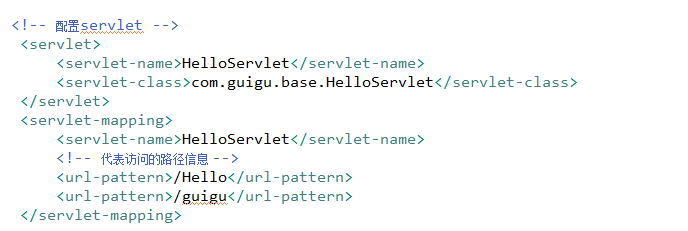
**protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

resp.getOutputStream().write("hello servlet".getBytes());

}

}

在web.xml中进行相关的配置



**方式二： 使用注解的方式配置servlet**

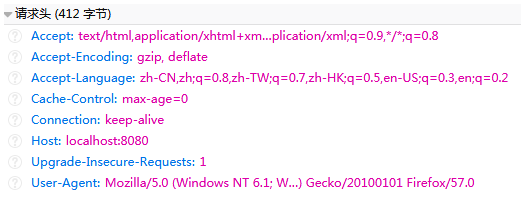


## Servlet的运行流程

Servlet

WEB服务器

http://localhost:8080/JavaWEB/helloguigu



2.发送http请求

1.连接web服务器

IE浏览器

3.解析客户端需要访问的主机

http://localhost:8080

4.解析需要访问的web应用

http://localhost:8080/JavaWEB

http://localhost:8080/JavaWEB/helloguigu

5.接续客户端需要访问的web资源

6.如果是第一次访问servlet 服务器就是加载servlet对象，并且创建servlet对象

11.web服务解析response对象然后根据解析的结果

返回给浏览器

10.把response对象返回给web服务器

9.service方法执行完毕会把数据写入到response对象中

调用service方法

response.getWriter().append("Served at: guigu\_hello").append(request.getContextPath());

8.调用service方法，service方法会根据请求

分别调用doGet和doPost方法

调用init方法

7.如果是第一次访问调用servlet的init方法进行初始化

## Servlet的生命周期

笔面试题 请描述servlet的生命周期

**Servlet的生命周期和工作原理**

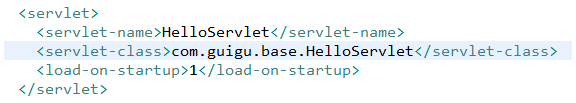
Servlet的的声明周期分为三个阶段，1.初始化阶段。2响应客户阶段。3.终止阶段

这三个阶段分别对应init方法 service方法 和destory方法 这三个方法本别在不同的阶段进行调用。

**1）Servlet的初始化阶段**

在以下时刻servlet进行初始化

1.servlet容器启动后，自动加载某些配置信息。需要配置<load-on-startup></load-on-startup>



2.在servlet容器启动后，客户端第一次向servlet发起请求

3.servlet的文件被更改后 重新加载servlet

**注意： init方法只会在第一次初始化servlet的时候执行。**

**2）servlet响应客户端阶段**

Servlet当接受到客户端请求后，servlet会根据这个请求创建两个对象HttpRequest对象和HttpResponse对象，这个对象分别代表接受请求和响应请求，然后servlet调用service方法。Service方法会根据HttpRequest对象从请求中获取相关的数据，然后处理请求，最后通过HttpResponse对象把数据写出到客户端。

**3）servlet终止阶段**

当web应用被终止，或者servlet容器终止运行，或servlet容器被重新加载

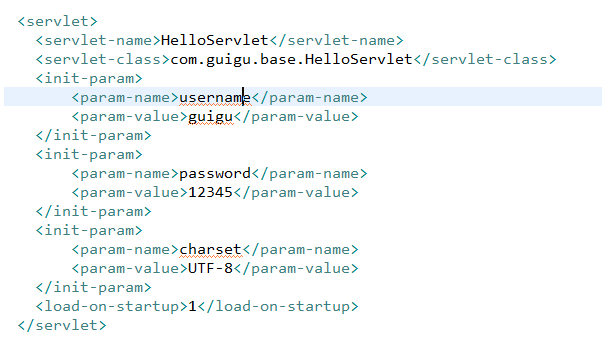
当前终止的时候回调用destroy方法进行终止。

**连接源码**

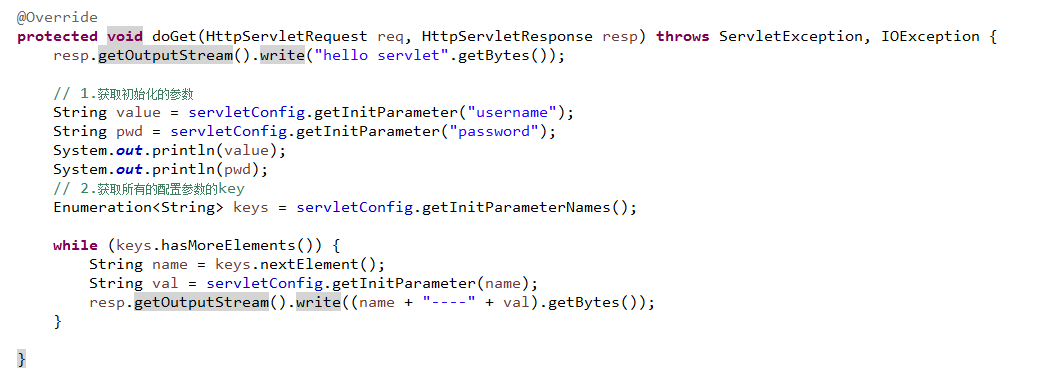
## Servlet相关的对象

### ServletConfig对象

* 在Servlet的配置文件中，可以使用一个或多个<init-param>标签为servlet配置一些初始化参数。
* 当servlet配置了初始化参数后，web容器在创建servlet实例对象时，会自动将这些初始化参数封装到ServletConfig对象中，并在调用servlet的init方法时，将ServletConfig对象传递给servlet。进而，程序员通过ServletConfig对象就可以得到当前servlet的初始化参数信息。
* 阅读ServletConfig API，并举例说明该对象的作用：
  + 获得字符集编码
  + 获得数据库连接信息
  + 获得配置文件，查看struts案例的web.xml文件



**HelloServlet**



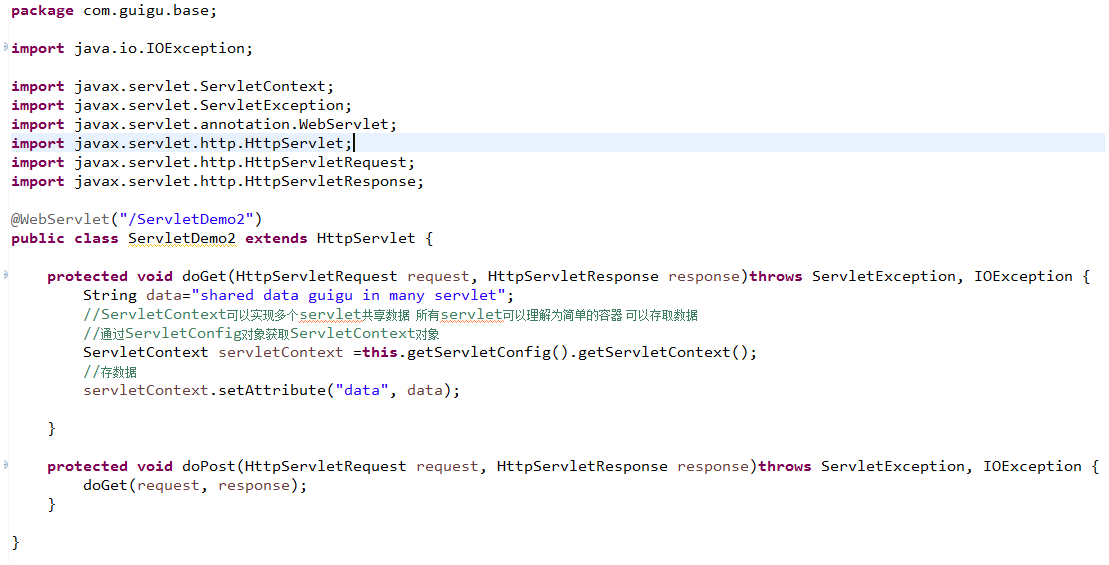
方式二在注解开发中使用配置参数



### ServletContext

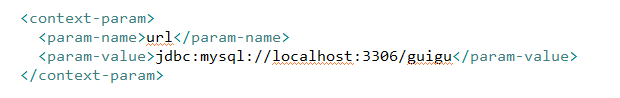
* WEB容器在启动时，它会为每个WEB应用程序都创建一个对应的ServletContext对象，它代表当前web应用。
* ServletConfig对象中维护了ServletContext对象的引用，开发人员在编写servlet时，可以通过ServletConfig.getServletContext方法获得ServletContext对象。
* 由于一个WEB应用中的所有Servlet共享同一个ServletContext对象，因此Servlet对象之间可以通过ServletContext对象来实现通讯。ServletContext对象通常也被称之为**context域对象**。
* 查看ServletContext API文档，了解ServletContext对象的功能。

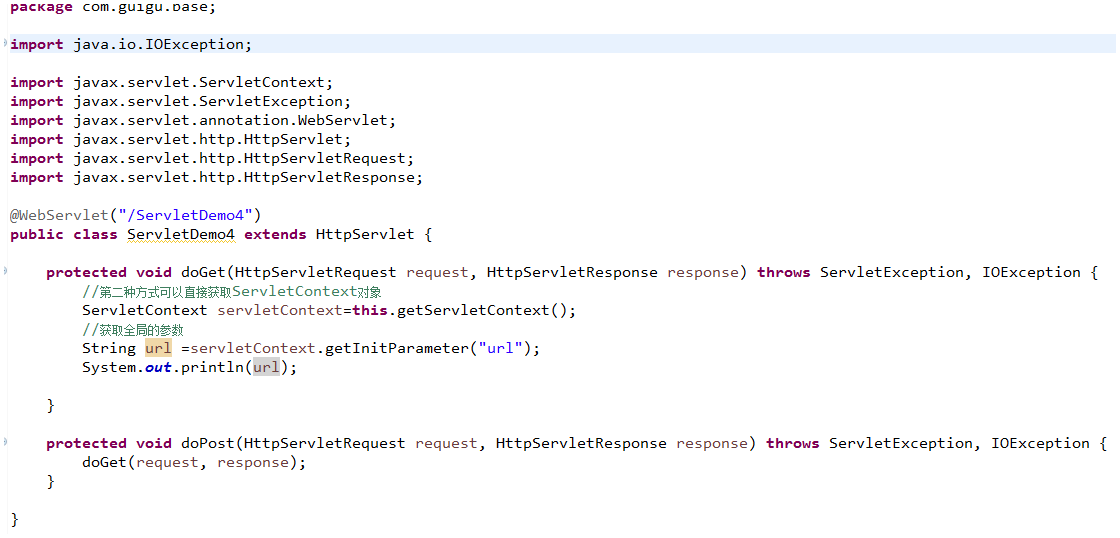
**第一个案例 多个servlet共享数据**



**第二个案例 获取全局的web初始化参数**

在web.xml中配置的信息





**第三个案例 实现servlet之间的转发**



**第四个案例 读取资源文件**



## Request对象和Response对象

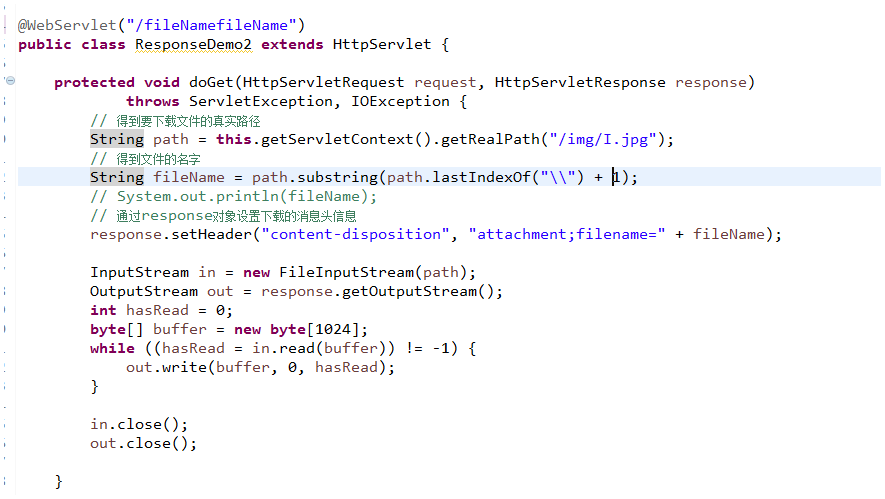
* Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的request对象、和代表响应的response对象。
* request和response对象即然代表请求和响应，那我们要获取客户机提交过来的数据，只需要找request对象就行了。要向客户机输出数据，只需要找response对象就行了。

### HttpServletResponse对象

**第一个案例 通过response对象输出中文到客户端**



**第二个案例 通过response对象输完成文件的下载**



**第三个案例 通过response对象生成随机图片(验证码)**

**package** com.guigu.response;

**import** java.awt.Color;

**import** java.awt.Font;

**import** java.awt.Graphics;

**import** java.awt.image.BufferedImage;

**import** java.io.IOException;

**import** java.util.Random;

**import** javax.imageio.ImageIO;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/ResponseDemo3")

**public** **class** ResponseDemo3 **extends** HttpServlet {

// 1.定义随机字典

**public** **static** **final** **char**[] ***CHARS*** = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F',

'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'};

//定义随机数

**public** **static** Random *random* =**new** Random();

//2.获取六位的随机数

**public** **static** String getRandomString() {

StringBuffer sb =**new** StringBuffer();

**for**(**int** i=0;i<6;i++) {

sb.append(***CHARS***[*random*.nextInt(***CHARS***.length)]);

}

**return** sb.toString();

}

//3.获取图片背景 随机颜色

**public** **static** Color getRandomColor() {

**return** **new** Color(*random*.nextInt(255), *random*.nextInt(255), *random*.nextInt(255));

}

//4.获取反色

**public** **static** Color getReverseColor(Color c) {

**return** **new** Color(255-c.getRed(), 255-c.getGreen(), 255-c.getBlue());

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

//设置输出的数据类型 imgage

response.setContentType("image/jpeg");

//得到随机的字符串

String randomString =*getRandomString*();

//得到随机颜色

Color color =*getRandomColor*();

//得到反色

Color reverse =*getReverseColor*(color);

//创建图片信息输出到客户端

BufferedImage bi =**new** BufferedImage(100, 30, BufferedImage.***TYPE\_INT\_RGB***);

//绘制图像 得到画笔

Graphics g =bi.getGraphics();

//设置颜色

g.setColor(color);

//绘制背景

g.fillRect(0, 0, 100, 30);

//设置相反的颜色

g.setColor(reverse);

//设置字体

g.setFont(**new** Font(Font.***SANS\_SERIF***, Font.***BOLD***, 16));

g.drawString(randomString, 18, 20);

//绘制噪点

**for**(**int** i=0;i<*random*.nextInt(500);i++) {

g.drawRect(*random*.nextInt(100), *random*.nextInt(30), 1, 1);

}

//把数据写出即可

ImageIO.*write*(bi,"jpg", response.getOutputStream());

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

doGet(request, response);

}

}

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<script type=*"text/javascript"*>

**function** reloadImage(){

document.getElementById('btn').disabled=**true**;

document.getElementById('identity').src='ResponseDemo3?ts='+**new** Date().getTime();

}

</script>

<body>

<img alt=*"图片不存在"* src=*"ResponseDemo3"* id=*"identity"* onload="btn.disabled=**false**">

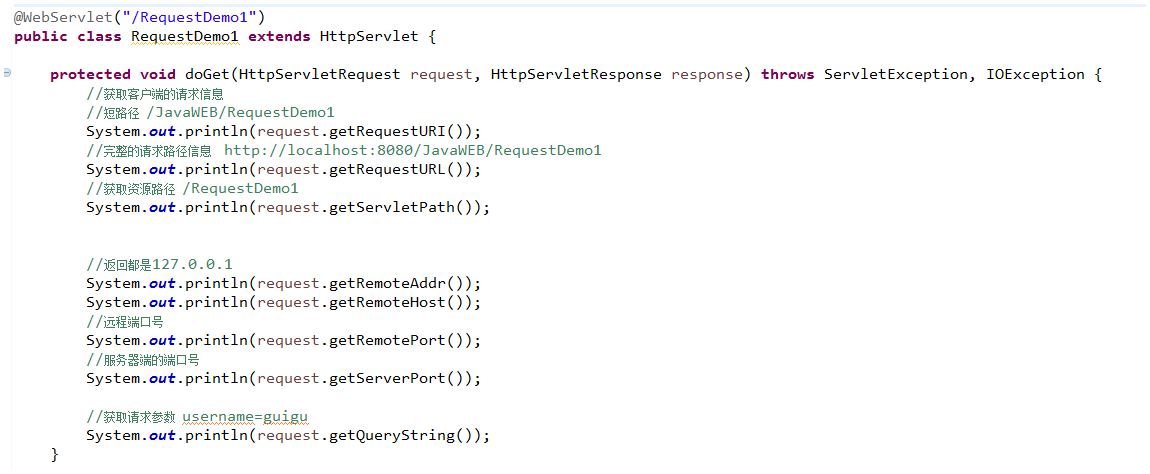
<input type=*"button"* value=*"换个图片"* onclick="reloadImage()" id=*"btn"*>

</body>

</html>

### HttpServletRequest对象

**第一个案例 通过request对象常用的方法**



**第二个案例获取表单的各项输入的数据**

表单

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title></title>

</head>

<script src=*"../js/user.js"* type=*"text/javascript"* charset=*"utf-8"*></script>

<script type=*"text/javascript"*>

**var** city=[

["杭州","宁波","温州","嘉兴"],

["济南","青岛","威海","济宁"],

["南京","苏州","徐州","南通"],

["厦门","福州","福安","龙岩"],

["兰州","敦煌","白银","定西"]

];

**function** getCity(){

//1.获取 用户选择了哪个省份

**var** sltProvince=document.forms[0].province;

**var** sltcity=document.forms[0].city;

//2得到省份对应的城市列表

//alert(sltProvince.selectedIndex)//得到用户选中的下标

**var** provinceCity=city[sltProvince.selectedIndex-1];

//3.将城市下拉列表置空

sltcity.length=1;

//4.填充数据

**for**( **var** i=0;i<provinceCity.length;i++){

//5.创建Option选项

sltcity[i+1]=**new** Option(provinceCity[i],provinceCity[i]);

}

}

//完成三级联动 ，作业

</script>

<body onload="pageInit()">

<form action=*"RequestDemo4"* method=*"post"* name=*"form1"*>

<table border=*"1"*>

<tr>

<td>用户名：</td>

<td><input type=*"text"* name=*"username"* /></td>

</tr>

<tr>

<td>密码</td>

<td><input type=*"password"* name=*"password"* ></td>

</tr>

<tr>

<td>确认密码</td>

<td><input type=*"password"* name=*"repassword"* /></td>

</tr>

<tr>

<td>性别:</td>

<td>

<input type=*"radio"* checked=*"checked"* name=*"sex"* value=*"male"* />男

<input type=*"radio"* name=*"sex"* value=*"female"* />女

</td>

</tr>

<tr>

<td>生日:</td>

<td>

<select id=*"year"* name=*"year"*>

<option name=*"year"* value=*"1900"*>1900</option>

</select>年

<select id=*"month"* name=*"month"*>

<option name=*"month"* value=*"01"*>01</option>

</select>月

<select id=*"day"* name=*"day"*>

<option name=*"day"* value=*"01"*>01</option>

</select>日

</td>

</tr>

<tr>

<td>兴趣爱好:</td>

<td>

<input type=*"checkbox"* name=*"hobby"* value=*"game"* />打游戏

<input type=*"checkbox"* name=*"hobby"* value=*"eat"* />吃饭

<input type=*"checkbox"* name=*"hobby"* value=*"sleep"* />睡觉

<input type=*"checkbox"* name=*"hobby"* value=*"qdm"* />敲代码

</td>

</tr>

<tr>

<td>学历：</td>

<td>

<select name=*"edu"*>

<option value=*"boshi"*>博士</option>

<option value=*"shuoshi"*>硕士</option>

<option value=*"benke"*>本科</option>

<option value=*"zhuanke"*>专科</option>

<option value=*"gaozhong"*>高中</option>

</select>

</td>

</tr>

<tr>

<td>地址：</td>

<td>

<select name=*"province"* onchange=*"getCity()"*>

<option value=*"0"*>请选择所在省份</option>

<option value=*"浙江省"*>浙江省</option>

<option value=*"山东省"*>山东省</option>

<option value=*"江苏省"*>江苏省</option>

<option value=*"福建省"*>福建省</option>

<option value=*"甘肃省"*>甘肃省</option>

</select>

<select name=*"city"* onchange=*""*>

<option value=*"0"*>请选择所在的城市</option>

</select>

</td>

</tr>

<tr>

<td>

<input type=*"reset"* value=*"重置"* />

</td>

<td>

<input type=*"submit"* value=*"提交"* />

</td>

</tr>

</table>

</form>

</body>

</html>

获取代码

**package** com.guigu.response;

**import** java.io.IOException;

**import** java.util.Arrays;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/html/RequestDemo4")

**public** **class** RequestDemo4 **extends** HttpServlet {

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

//设置编码集

request.setCharacterEncoding("UTF-8");

//这里获取数据是根据表单中的name属性进行获取 vaulue值

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

String pwd =request.getParameter("password");

String repwd=request.getParameter("repassword");

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

String birthday=request.getParameter("year")+"-"+request.getParameter("month")+"-"+request.getParameter("day");

String [] likes =request.getParameterValues("hobby");

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

String addr=request.getParameter("province")+"\_"+request.getParameter("city");

//把数据封装到Model对象中 然后对Model对象进行相关的操作 即可

System.***out***.println(username+"---"+pwd+"---"+repwd+"---"+sex+"--"+birthday+"---"+edu+"---"+addr+"--"+Arrays.*toString*(likes));

}

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

doGet(request, response);

}

}

**第三个案例 通过request请求参数中文乱码问题**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"RequestDemo2"* method=*"post"*>

用户名:<input type=*"text"* name=*"username"*>

<input type=*"submit"* value=*"提交"*>

</form>

<br/>

<form action=*"RequestDemo2"* method=*"get"*>

用户名:<input type=*"text"* name=*"username"*>

<input type=*"submit"* value=*"提交"*>

</form>

</body>

</html>

**package** com.guigu.request;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/html/RequestDemo2")

**public** **class** RequestDemo2 **extends** HttpServlet {

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

request.setCharacterEncoding("UTF-8");

//需要再次编码

// String uname=request.getParameter("username");

// uname=new String(uname.getBytes("ISO-8859-1"),"UTF-8");

// System.out.println(uname);

System.***out***.println(request.getParameter("username"));

}

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

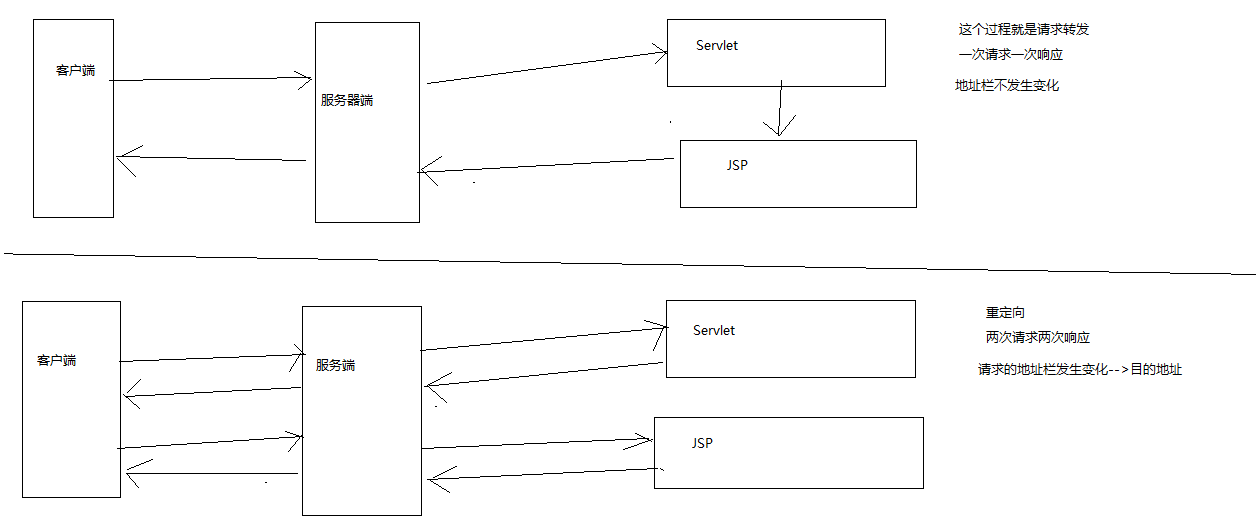
doGet(request, response);

}

}

### 请求转发和重定向

* 一个web资源接收到客户端请求后，通知浏览器去调用另外一个web资源进行处理，称之为请求转发
* 一个web资源接收到客户端请求后，通知浏览器去访问另外一个web资源，称之请求重定向



**package** com.guigu.request;

**import** java.io.IOException;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/RequestDemo3")

**public** **class** RequestDemo3 **extends** HttpServlet {

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

//重定向

// response.sendRedirect("/JavaWEB/index.html");

//请求转发

// RequestDispatcher rd=this.getServletContext().getRequestDispatcher("");

RequestDispatcher rd =request.getRequestDispatcher("/index.html");

rd.forward(request, response);

}

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

doGet(request, response);

}

}

**面试题：请求转发和重定向有什么区别？**

* RequestDispatcher.forward方法只能将请求转发给同一个WEB应用中的组件；而HttpServletResponse.sendRedirect 方法还可以重定向到同一个站点上的其他应用程序中的资源，甚至是使用绝对URL重定向到其他站点的资源。
* 如果传递给HttpServletResponse.sendRedirect 方法的相对URL以“/”开头，它是相对于整个WEB站点的根目录；如果创建RequestDispatcher对象时指定的相对URL以“/”开头，它是相对于当前WEB应用程序的根目录。
* 调用HttpServletResponse.sendRedirect方法重定向的访问过程结束后，浏览器地址栏中显示的URL会发生改变，由初始的URL地址变成重定向的目标URL；调用RequestDispatcher.forward 方法的请求转发过程结束后，浏览器地址栏保持初始的URL地址不变。
* HttpServletResponse.sendRedirect方法对浏览器的请求直接作出响应，响应的结果就是告诉浏览器去重新发出对另外一个URL的访问请求；RequestDispatcher.forward方法在服务器端内部将请求转发给另外一个资源，浏览器只知道发出了请求并得到了响应结果，并不知道在服务器程序内部发生了转发行为。
* RequestDispatcher.forward方法的调用者与被调用者之间共享相同的request对象和response对象，它们属于同一个访问请求和响应过程；而HttpServletResponse.sendRedirect方法调用者与被调用者使用各自的request对象和response对象，它们属于两个独立的访问请求和响应过程。

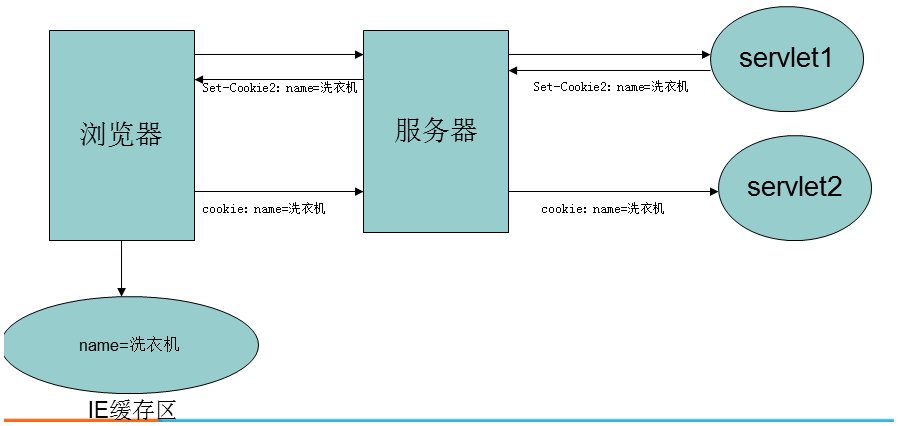
# 会话管理

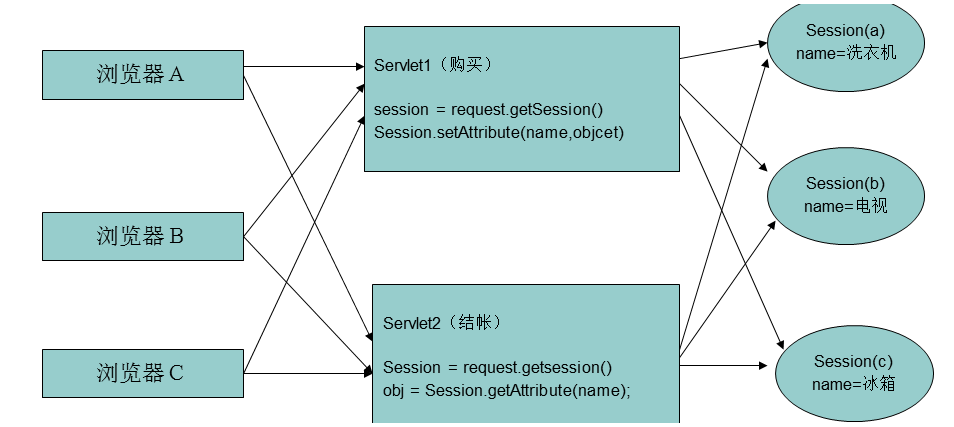
* 什么是会话？
  + 会话可简单理解为：用户开一个浏览器，点击多个超链接，访问服务器多个web资源，然后关闭浏览器，整个过程称之为一个会话。
* 会话过程中要解决的一些问题？
  + 每个用户在使用浏览器与服务器进行会话的过程中，不可避免各自会产生一些数据，程序要想办法为每个用户保存这些数据。
  + 例如：用户点击超链接通过一个servlet购买了一个商品，程序应该想办法保存用户购买的商品，以便于用户点结帐servlet时，结帐servlet可以得到用户购买的商品为用户结帐。
  + 思考：用户购买的商品保存在request或servletContext中行不行？

***保存*会话数据的两种技术：**

* Cookie
  + Cookie是客户端技术，程序把每个用户的数据以cookie的形式写给用户各自的浏览器。当用户使用浏览器再去访问服务器中的web资源时，就会带着各自的数据去。这样，web资源处理的就是用户各自的数据了。
* Session
  + Session是服务器端技术，利用这个技术，服务器在运行时可以为每一个用户的浏览器创建一个其独享的session对象，由于session为用户浏览器独享，所以用户在访问服务器的web资源时，可以把各自的数据放在各自的session中，当用户再去访问服务器中的其它web资源时，其它web资源再从用户各自的session中取出数据为用户服务。

## Cookie





**第一个案例 通过cookie进行保存和获取数据**

**package** com.guigu.cookie;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/CookieDemo1")

**public** **class** CookieDemo1 **extends** HttpServlet {

// 显示用户上次访问时间

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter pw = response.getWriter();

pw.write("这是网站首页!!<br/><br/>");

pw.write("您上次访问的时间是:");

// 通过cookie中数据查询记录

Cookie[] cookies = request.getCookies();

**for** (**int** i = 0;cookies!=**null** && i < cookies.length; i++) {

// 取出每一个Cookie

Cookie cok = cookies[i];

**if**(cok.getName().equals("lastAccessTime")) {

//得到对于的value值 然后显示到客户端

Long time =Long.*parseLong*(cok.getValue());

//把Long类型的数据转换为对应的日期类型

Date date =**new** Date(time);

SimpleDateFormat sdf =**new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String lastTime=sdf.format(date);

pw.write(lastTime);

}

}

// 把数据以Cookie的形式写出到客户端浏览器

Cookie cookie = **new** Cookie("lastAccessTime", System.*currentTimeMillis*() + "");

// 把cookie数据设置有效期和保存路径 然后保存到客户端

cookie.setMaxAge(10000);

cookie.setPath("/JavaWEB");

// 通过response对象把cookie添加到客户端

response.addCookie(cookie);

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

doGet(request, response);

}

}

火狐浏览器中通过aboutL:cache查看详细的cookie信息

**第二个案例 通过删除cookie中的数据**

**package** com.guigu.cookie;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/CookieDemo2")

**public** **class** CookieDemo2 **extends** HttpServlet {

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

//得到Cookie 然后进行相关的删除

//不通过已有的cookie进行获取 而是通过新建一个新的Cookie覆盖的之前的值

Cookie cookie = **new** Cookie("lastAccessTime", System.*currentTimeMillis*() + "");

//设置有效期

cookie.setMaxAge(0);

cookie.setPath("/JavaWEB");

//把数据添加到客户端

response.addCookie(cookie);

}

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

doGet(request, response);

}

}

**第三个案例 实现用户上次浏览的商品信息**

**注意的细节**

* 一个Cookie只能标识一种信息，它至少含有一个标识该信息的名称（NAME）和设置值（VALUE）。
* 一个WEB站点可以给一个WEB浏览器发送多个Cookie，一个WEB浏览器也可以存储多个WEB站点提供的Cookie。
* 浏览器一般只允许存放300个Cookie，每个站点最多存放20个Cookie，每个Cookie的大小限制为4KB。
* 如果创建了一个cookie，并将他发送到浏览器，默认情况下它是一个会话级别的cookie（即存储在浏览器的内存中），用户退出浏览器之后即被删除。若希望浏览器将该cookie存储在磁盘上，则需要使用maxAge，并给出一个以秒为单位的时间。将最大时效设为0则是命令浏览器删除该cookie。
* 注意，删除cookie时，path必须一致，否则不会删除

## Session

* 在WEB开发中，服务器可以为每个用户浏览器创建一个会话对象（session对象），注意：**一个浏览器独占一个session对象**(默认情况下)。因此，在需要保存用户数据时，服务器程序可以把用户数据写到用户浏览器独占的session中，当用户使用浏览器访问其它程序时，其它程序可以从用户的session中取出该用户的数据，为用户服务。
* Session和Cookie的主要区别在于：
  + Cookie是把用户的数据写给用户的浏览器。
  + Session技术把用户的数据写到用户独占的session中。
* Session对象由服务器创建，开发人员可以调用request对象的getSession方法得到session对象。

**第一个案例 Session 的基础应用**

**package** com.guigu.session;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

@WebServlet("/SessionDemo1")

**public** **class** SessionDemo1 **extends** HttpServlet {

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

//1.获取Session对象

HttpSession session =request.getSession();

//2.利用Session对象的方法进行存取数据

session.setAttribute("data", "hello2018,愿你十八所爱，是你八十所伴");

//3.通过Cookie把Session保存下来

// System.out.println(session.getId());

Cookie cookie =**new** Cookie("JSESSIONID", session.getId());

cookie.setMaxAge(1\*24\*3600);

cookie.setPath("/JavaWEB");

response.addCookie(cookie);

}

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

doGet(request, response);

}

}

**package** com.guigu.session;

**import** java.io.IOException;

**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** javax.servlet.http.HttpSession;

@WebServlet("/SessionDemo2")

**public** **class** SessionDemo2 **extends** HttpServlet {

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

HttpSession session =request.getSession();

//是从session中取出数据

System.***out***.println(session.getAttribute("data"));

}

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

doGet(request, response);

}

}

**第二个案例 Session实现购物功能**

**package** com.guigu.session;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Map;

**import** java.util.Set;

**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** com.guigu.cookie.Book;

**import** com.guigu.cookie.DB;

@WebServlet("/ShoppingIndex")

**public** **class** ShoppingIndex **extends** HttpServlet {

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

response.setContentType("text/html;charset=UTF-8");

PrintWriter pw =response.getWriter();

pw.write("本网站有如下书籍信息");

pw.write("<br/>");

pw.write("<br/>");

Set<Map.Entry<String, Book>> set =DB.*getAll*().entrySet();

**for**(Map.Entry<String, Book> mm:set) {

Book book =mm.getValue();

pw.write("<a href='/JavaWEB/BuyServlet?id="+book.getId()+"' target='\_blank'>"+book.getName()+"</a>");

pw.write("<br/>");

}

}

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

doGet(request, response);

}

}

**package** com.guigu.session;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.Cookie;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** javax.servlet.http.HttpSession;

**import** com.guigu.cookie.Book;

**import** com.guigu.cookie.DB;

@WebServlet("/BuyServlet")

**public** **class** BuyServlet **extends** HttpServlet {

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

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

//查询用户要购买的书籍信息

Book book =DB.*getAll*().get(id);

//通过Session对象保存用户的购物记录

HttpSession session =request.getSession();

Cookie cookie =**new** Cookie("JSESSIONID", session.getId());

cookie.setMaxAge(1\*24\*3600);

cookie.setPath("/JavaWEB");

response.addCookie(cookie);

//得到现有的购物信息

List shoppinglist =(List) session.getAttribute("list");

**if**(shoppinglist==**null**) {

shoppinglist=**new** ArrayList();

session.setAttribute("list", shoppinglist);

}

//把要购买的书籍放置到list集合中

shoppinglist.add(book);

//展示用户的购物车的内容

request.getRequestDispatcher("/ListCartServlet").forward(request, response);

}

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

// **TODO** Auto-generated method stub

doGet(request, response);

}

}

**package** com.guigu.session;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.List;

**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** javax.servlet.http.HttpSession;

**import** com.guigu.cookie.Book;

@WebServlet("/ListCartServlet")

**public** **class** ListCartServlet **extends** HttpServlet {

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

response.setContentType("text/html;charset=UTF-8");

PrintWriter pw =response.getWriter();

//通过Session对象获取相关的数据

HttpSession session =request.getSession();

List<Book> list =(List<Book>) session.getAttribute("list");

**if**(list==**null** || list.size()==0) {

pw.write("对不起,您还没有购买任何商品信息");

}

//显示用户的购物的商品信息

pw.write("您购买的商品信息如下:");

**for** (Book book : list) {

pw.write(book.getName()+"<br/>");

}

}

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

doGet(request, response);

}

}

**第三个案例 Session 实现登录操作**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"/JavaWEB/LoginServlet"* method=*"post"*>

用户名: <input type=*"text"* name=*"username"*><br/>

密码: <input type=*"password"* name=*"password"*><br/>

<input type=*"submit"* value=*"登录"*>

</form>

</body>

</html>

**package** com.guigu.session2;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")

**public** **class** LoginServlet **extends** HttpServlet {

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

response.setContentType("text/html;charset=UTF-8");

PrintWriter pw =response.getWriter();

//得到用户输入的用户名或者密码

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

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

User findUser =DBUser.*find*(username, password);

**if**(findUser==**null**) {

pw.write("用户名或者密码错误");

**return** ;

}

//把查询的到的用户的信息保存到session中

request.getSession().setAttribute("user", findUser);

//用户登录成功 跳转到项目的首页

response.sendRedirect("/JavaWEB/home.jsp");

}

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

// **TODO** Auto-generated method stub

doGet(request, response);

}

}

**package** com.guigu.session2;

/\*\*

\* <p>project\_name:JavaWEB</p>

\* <p>package\_name:com.guigu.session2.DBUser</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月2日上午9:24:40 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** DBUser {

**private** **static** List<User> *list* = **new** ArrayList<>();

**static** {

*list*.add(**new** User("aaa", "123a", 14));

*list*.add(**new** User("bbb", "123b", 15));

*list*.add(**new** User("ccc", "123c", 16));

*list*.add(**new** User("ddd", "123d", 17));

}

**public** **static** User find(String username, String password) {

**for** (User user : *list*) {

**if** (user.getUsername().equals(username) && user.getPassword().equals(password)) {

**return** user;

}

}

**return** **null**;

}

}

<%@page import=*"com.guigu.session2.User"*%>

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

欢迎您:

<%

User user =(User)session.getAttribute("user");

**if**(user!=**null**){

out.write(user.getUsername());

}

%>

<br/>

<a href=*"/JavaWEB/login.html"*>登录</a>

<a href=*"/JavaWEB/LogoutServlet"*>退出</a>

</body>

</html>

<%@page import=*"com.guigu.session2.User"*%>

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

欢迎您:

<%

User user =(User)session.getAttribute("user");

**if**(user!=**null**){

out.write(user.getUsername());

}

%>

<br/>

<a href=*"/JavaWEB/login.html"*>登录</a>

<a href=*"/JavaWEB/LogoutServlet"*>退出</a>

</body>

</html>

# JSP

## JSP的概述

* JSP全称是Java Server Pages，它和servle技术一样，都是SUN公司定义的一种用于开发动态web资源的技术。
* JSP这门技术的最大的特点在于，写jsp就像在写html，但它相比html而言，html只能为用户提供静态数据，而Jsp技术允许在页面中嵌套java代码，为用户提供动态数据。
* Jsp快速入门：在jsp页面中输出当前时间。

<%@page import=*"java.util.Date"*%>

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

Date date =**new** Date();

out.write(date.toLocaleString());

%>

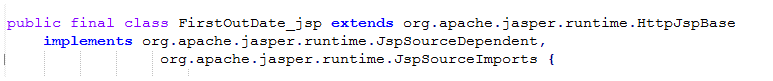
</body>

</html>

**JSP原理**

* Web服务器是如何调用并执行一个jsp页面的？

Tomcat把jsp页面翻译为.java文件 这个.java 文件实际上就是一个servlet





Jsp的执行过程 就是servlet的调用过程。

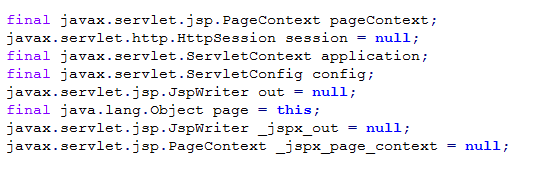
* Jsp页面中的html排版标签是如何被发送到客户端的？

调用servlet的service方法 通过jspwriter对象的writer方法直接写出到页面中

* Jsp页面中的java代码服务器是如何执行的？

Java代码没有人还不 直接到servlet中 通java文件执行

* Web服务器在调用jsp时，会给jsp提供一些什么java对象？



面试题：JSP和Servlet的区别？

## JSP的语法

* JSP模版元素
* JSP表达式
* JSP脚本片断
* JSP注释
* JSP指令
* JSP标签
* JSP内置对象
* 如何查找JSP页面中的错误

### JSP模版元素

* JSP页面中的HTML内容称之为JSP模版元素。
* JSP模版元素定义了网页的基本骨架，即定义了页面的结构和外观。

JSP源代码分为两个部分 模板数据和元素

模板数据：就是JSP中html代码，它的内容是固定的无论程序怎样运行 模板数据输出到客户端浏览器是不会发生变化的，模板数据不会控制程序的流程 仅仅是展示页面的结构和外观。

元素： 就是jsp中的java代码 包括脚本 以及指令还有标签 。元素决定程序的流程。

### JSP表达式 &脚本片断

* JSP脚本表达式（expression）用于将程序数据输出到客户端

语法：<%= 变量或表达式 %>

**举例：**当前时间:<%= new java.util.Date() %>

* JSP引擎在翻译脚本表达式时，会将程序数据转成字符串，然后在相应位置用out.print(…) 将数据输给客户端。
* JSP脚本表达式中的变量或表达式后面**不能有分号（;）**。

**注意的细节**

* JSP脚本片断（scriptlet）用于在JSP页面中编写多行Java代码。语法：

<%

多行java代码

%>

* 注意：JSP脚本片断中只能出现java代码，不能出现其它模板元素， JSP引擎在翻译JSP页面中，会将JSP脚本片断中的Java代码将被原封不动地放到Servlet的\_jspService方法中。
* JSP脚本片断中的Java代码必须严格遵循Java语法，例如，每执行语句后面必须用分号（;）结束。
* 单个脚本片断中的Java语句可以是不完整的，但是，多个脚本片断组合后的结果必须是完整的Java语句，例如：

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

System.out.println("hello");

**for**(**int** i=0; i<4;i++){

%>

<%= i %>

<%

}

%>

</body>

</html>

### JSP注释

<!-- 我是注释 我可以在页面上通过查看源码查看到 -->

<%-- 我是jsp的注释 在页面查看代码查看不到我 我是隐式注释--%>

<%

System.out.println("hello");

**for**(**int** i=0; i<4;i++){

%>

<%= i %>

<%

//---可以使用java 的注释

}

%>

### JSP指令

* JSP指令（directive）是为JSP引擎而设计的，它们并不直接产生任何可见输出，而只是告诉引擎如何处理JSP页面中的其余部分。在JSP 2.0规范中共定义了三个指令：
  + page指令
  + Include指令
  + taglib指令
* JSP指令的基本语法格式：

<%@ 指令 属性名="值" %>

举例：<%@ page contentType="text/html;charset=gb2312"%>

* 如果一个指令有多个属性，这多个属性可以写在一个指令中，也可以分开写。

例如：

<%@ page contentType="text/html;charset=gb2312"%>

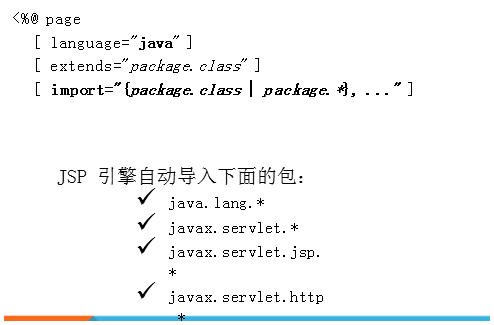
<%@ page import="java.util.Date"%>

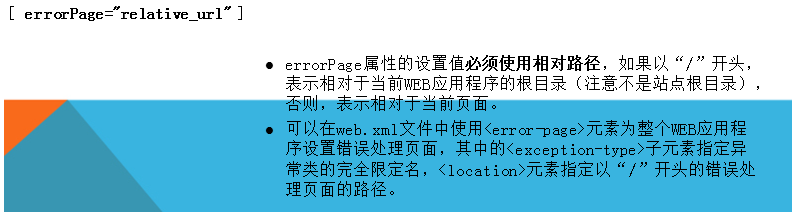
也可以写作：

<%@ page contentType="text/html;charset=gb2312" import="java.util.Date"%>

#### page指令

* page指令用于定义JSP页面的各种属性，无论page指令出现在JSP页面中的什么地方，它作用的都是整个JSP页面，为了保持程序的可读性和遵循良好的编程习惯，page指令最好是放在整个JSP页面的起始位置。





<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"* errorPage=*"../error.jsp"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

**int** i=10/0;

%>

</body>

</html>

Web.xml的配置

<error-page>

<exception-type>java.lang.ArithmeticException</exception-type>

<location>/error.jsp</location>

</error-page>

<error-page>

<error-code>404</error-code>

<location>/404.jsp</location>

</error-page>

#### Include指令

* include指令用于引入其它JSP页面，如果使用include指令引入了其它JSP页面，那么JSP引擎将把这**两个JSP翻译成一个servlet**。所以include指令引入通常也称之为静态引入。
* 语法：

<%@ include file="relativeURL"%>

其中的file属性用于指定被引入文件的路径。路径以“/”开头，表示代表当前web应用。

* 细节：
  + 被引入的文件必须遵循JSP语法。
  + 被引入的文件可以使用任意的扩展名，即使其扩展名是html，JSP引擎也会按照处理jsp页面的方式处理它里面的内容，为了见明知意，JSP规范建议使用.jspf（JSP fragments）作为静态引入文件的扩展名。
  + 由于使用include指令将会涉及到2个JSP页面，并会把2个JSP翻译成一个servlet，所以这2个JSP页面的指令不能冲突（除了pageEncoding和导包除外）。

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%@ include file=*"foot.jspf"* %>

<h1>我是main中的内容</h1>

<%@ include file=*"head.jspf"* %>

</body>

</html>

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

<h1>我是foot的部分内容</h1>

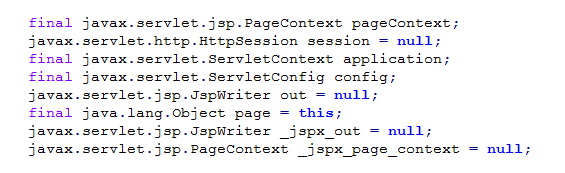
<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>

<h1>我是head的部分内容</h1>

### JSP内置对象

* 每个JSP 页面在第一次被访问时，WEB容器都会把请求交给JSP引擎（即一个Java程序）去处理。JSP引擎先将JSP翻译成一个\_jspServlet(实质上也是一个servlet) ，然后按照servlet的调用方式进行调用。
* 由于JSP第一次访问时会翻译成servlet，所以第一次访问通常会比较慢，但第二次访问，JSP引擎如果发现JSP没有变化，就不再翻译，而是直接调用，所以程序的执行效率不会受到影响。
* JSP引擎在调用JSP对应的\_jspServlet时，会传递或创建9个与web开发相关的对象供\_jspServlet使用。JSP技术的设计者为便于开发人员在编写JSP页面时获得这些web对象的引用，特意定义了9个相应的变量，开发人员在JSP页面中通过这些变量就可以快速获得这9大对象的引用。
* 这9个对象分别是哪些，以及作用也是笔试经常考察的知识点。

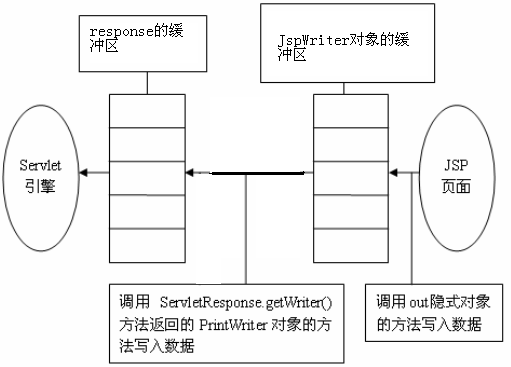
JSP有九大内置对象



1. 输出和输入对象 ：Request对象，response对象 Out对象
2. 通信控制对象：pageContext对象，session对象，application对象
3. Servlet对象： page对象 ，config对象
4. 错误处理对象:Exception对象

#### Out对象(JspWriter对象)

* out隐式对象用于向客户端发送文本数据。
* out对象是通过调用pageContext对象的getOut方法返回的，其作用和用法与ServletResponse.getWriter方法返回的PrintWriter对象非常相似。
* JSP页面中的out隐式对象的类型为JspWriter，JspWriter相当于一种带缓存功能的PrintWriter，设置JSP页面的page指令的buffer属性可以调整它的缓存大小，甚至关闭它的缓存。
* 只有向out对象中写入了内容，且满足如下任何一个条件时，out对象才去调用ServletResponse.getWriter方法，并通过该方法返回的PrintWriter对象将out对象的缓冲区中的内容真正写入到Servlet引擎提供的缓冲区中：
  + 设置page指令的buffer属性关闭了out对象的缓存功能
  + out对象的缓冲区已满
  + 整个JSP页面结束



<%

//out是jsp的内置对象 JspWriter

out.write("aaaaa");

//response.getWriter得到的是PrintWriter

response.getWriter().write("bbbb");

out.write("cccc");

response.getWriter().write("dddd");

%>

#### 域对象

域对象 是能存取数据 并且有自己的存取范围 四大域对象

**pageContext 当前页有效**

**request 一次请求有效**

**session 一个ip地址从头到尾有效 一次会话**

**application 所有的都有效**

* pageContext对象是JSP技术中最重要的一个对象，它代表JSP页面的运行环境，**这个对象不仅封装了对其它8大隐式对象的引用，它自身还是一个域对象**，可以用来保存数据。并且，这个对象还封装了web开发中经常涉及到的一些常用操作，例如引入和跳转其它资源、检索其它域对象中的属性等。
* **getException方法返回exception隐式对象**
* **getPage方法返回page隐式对象**
* **getRequest方法返回request隐式对象**
* **getResponse方法返回response隐式对象**
* **getServletConfig方法返回config隐式对象**
* **getServletContext方法返回application隐式对象**
* **getSession方法返回session隐式对象**
* **getOut方法返回out隐式对象**
* **pageContext封装其它8大内置对象的意义，思考：如果在编程过程中，把pageContext对象传递给一个普通java对象，那么这个java对象将具有什么功能？**

<%

//1.pageContext作为域对象存储数据

pageContext.setAttribute("data", "guigu");

//2.pageContext作为域对象取出数据

String data =(String) pageContext.getAttribute("data");

//3. pageContext对象还有其他八大隐式对象的引用 并且能把数据设置到其他的域对象中

// HttpSession session2 =pageContext.getSession();

pageContext.setAttribute("data2", "hangzhouguigu",pageContext.SESSION\_SCOPE);

pageContext.setAttribute("data3", "hangzhouguigu111",pageContext.REQUEST\_SCOPE);

out.write(data);

out.write("<br/>");

out.write((String)(session.getAttribute("data2")));

out.write("<br/>");

out.write((String)(request.getAttribute("data3")));

%>

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

pageContext.setAttribute("username", "guigu");

request.setAttribute("age", "13");

session.setAttribute("sex", "male");

application.setAttribute("addr", "杭州西湖");

%>

<!-- 在本页直接获取四大域对象的数据 -->

<%

String pd=(String)pageContext.getAttribute("username");

String rd=(String)request.getAttribute("age");

String sd=(String)session.getAttribute("sex");

String ad=(String)application.getAttribute("addr");

out.println(pd);

out.println(rd);

out.println(sd);

out.println(ad);

%>

<a href=*"3.jsp"*>3.jsp</a>

</body>

</html>

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

String pd=(String)pageContext.getAttribute("username");

String rd=(String)request.getAttribute("age");

String sd=(String)session.getAttribute("sex");

String ad=(String)application.getAttribute("addr");

out.println(pd);

out.println(rd);

out.println(sd);

out.println(ad);

%>

</body>

</html>

使用application实现网页计数器

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

**if**(application.getAttribute("counter")==**null**){

application.setAttribute("counter", "1");

}**else**{

String num =application.getAttribute("counter").toString();

**int** count =Integer.parseInt(num);

count++;

application.setAttribute("counter", String.valueOf(count));

}

%>

欢迎访问本网站,您是第<%=application.getAttribute("counter").toString() %>访问者

</body>

</html>

### JSP标签

* JSP标签也称之为Jsp Action(JSP动作)元素，它用于在Jsp页面中提供业务逻辑功能，避免在JSP页面中直接编写java代码，造成jsp页面难以维护。
* <jsp:include>标签
* <jsp:forward>标签
* <jsp:param>标签

#### jsp:include

* <jsp:include>标签用于把另外一个资源的输出内容插入进当前JSP页面的输出内容之中，这种在JSP页面执行时的引入方式称之为动态引入。
* 语法：

<jsp:include page="relativeURL | <%=expression%>" flush="true|false" />

* + page属性用于指定被引入资源的相对路径，它也可以通过执行一个表达式来获得。
  + flush属性指定在插入其他资源的输出内容时，是否先将当前JSP页面的已输出的内容刷新到客户端。

***Tip：* <jsp:include>与include指令的比较**

* <jsp:include>标签是动态引入， <jsp:include>标签涉及到的2个JSP页面会被翻译成2个servlet，这2个servlet的内容在执行时进行合并。
* 而include指令是静态引入，涉及到的2个JSP页面会被翻译成一个servlet，其内容是在源文件级别进行合并。
* 不管是<jsp:include>标签，还是include指令，它们都会把两个JSP页面内容合并输出，所以这两个页面不要出现重复的HTML全局架构标签，否则输出给客户端的内容将会是一个格式混乱的HTML文档。

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<jsp:include page=*"head.jsp"*></jsp:include>

<br/> <br/>

我是本网页的内容 body <br/>

<jsp:include page=*"foot.jsp"*></jsp:include>

</body>

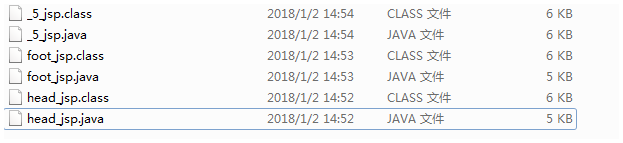
</html>

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

<h1> 使用jsp的标签include head的头部</h1>

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

<h1> 使用jsp的标签include foot底部</h1>



#### jsp:forward

* **<jsp:forward>标签用于把请求转发给另外一个资源。**
* **语法：**

<jsp:forward page="relativeURL | <%=expression%>" />

* + page属性用于指定请求转发到的资源的相对路径，它也可以通过执行一个表达式来获得。

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

String url="../login.html";

%>

<jsp:forward page=*"*<%=url %>*"*></jsp:forward>

</body>

</html>

#### jsp:param

* **当使用<jsp:include>和<jsp:forward>标签引入或将请求转发给其它资源时，可以使用<jsp:param>标签向这个资源传递参数。**
* **语法1：**

<jsp:include page="relativeURL | <%=expression%>">

<jsp:param name="parameterName" value="parameterValue|<%= expression %>" />

</jsp:include>

* **语法2：**

<jsp:forward page="relativeURL | <%=expression%>">

<jsp:param name="parameterName" value="parameterValue|<%= expression %>" />

</jsp:include>

* **<jsp:param>标签的name属性用于指定参数名，value属性用于指定参数值。在<jsp:include>和<jsp:forward>标签中可以使用多个<jsp:param>标签来传递多个参数。**

<%

String url="LoginServlet";

%>

<jsp:forward page=*"*<%=url %>*"*>

<jsp:param value=*"aaa"* name=*"username"*/>

<jsp:param value=*"123a"* name=*"password"*/>

</jsp:forward>

# JavaBean与JSP

## 基础概念

### JavaBean的概念

* JavaBean是一个遵循特定写法的Java类，它通常具有如下特点：
  + 这个Java类必须具有一个无参的构造函数
  + 属性必须私有化。
  + 私有化的属性必须通过public类型的方法暴露给其它程序，并且方法的命名也必须遵守一定的命名规范。
* 虽然Sun公司在定义JavaBean规范时，允许Java开发人员把JavaBean设计得可以像Swing组件一样功能强大，但在实际的J2EE开发中，通常只使用到以上JavaBean最基本的特性。
* JavaBean在J2EE开发中，通常用于封装数据，对于遵循以上写法的JavaBean组件，其它程序可以通过反射技术实例化JavaBean对象，并且通过反射那些遵守命名规范的方法，从而获知JavaBean的属性，进而调用其属性保存数据。

### JavaBean的属性

* JavaBean的属性可以是任意类型，并且一个JavaBean可以有多个属性。每个属性通常都需要具有相应的setter、 getter方法，setter方法称为属性修改器，getter方法称为属性访问器。
* 属性修改器必须以小写的set前缀开始，后跟属性名，且属性名的第一个字母要改为大写，例如，name属性的修改器名称为setName，password属性的修改器名称为setPassword。
* 属性访问器通常以小写的get前缀开始，后跟属性名，且属性名的第一个字母也要改为大写，例如，name属性的访问器名称为getName，password属性的访问器名称为getPassword。
* 一个JavaBean的某个属性也可以只有set方法或get方法，这样的属性通常也称之为只写、只读属性。

## 在jsp中使用JavaBean

* JSP技术提供了三个关于JavaBean组件的动作元素，即JSP标签，它们分别为：
* <jsp:useBean>标签：用于在JSP页面中查找或实例化一个JavaBean组件。
* <jsp:setProperty>标签：用于在JSP页面中设置一个JavaBean组件的属性。
* <jsp:getProperty>标签：用于在JSP页面中获取一个JavaBean组件的属性。

<jsp:useBean id=*"p"* class=*"com.guigu.jspjavabean.Person"* scope=*"page"*></jsp:useBean>

<!-- 方式一 name是针对哪个javabean进行赋值 property是针对javabean中的哪个属性进行赋值 -->

<jsp:setProperty property=*"id"* name=*"p"* value=*"1"*/>

<jsp:setProperty property=*"name"* name=*"p"* value=*"guigu"*/>

<jsp:setProperty property=*"age"* name=*"p"* value=*"13"*/>

<jsp:setProperty property=*"birthday"* name=*"p"* value=*"*<%=**new** Date() %>*"*/>

<!-- 方式二 通过请求参数进行赋值 -->

<%-- <jsp:setProperty property="name" name="p" param="username"/> --%>

<!-- 对javabean中所有的属性都通过参数的形式进行赋值 -->

<!-- http://localhost:8080/JavaWEB/javabean/1.jsp?id=12&username=hzguigu&age=99 -->

<jsp:setProperty name=*"p"* property=*"\*"*/>

<%

out.println(p.getId());

out.println(p.getName());

out.println(p.getAge());

out.println(p.getBirthday());

%>

<br/>

<br/>

<!-- 使用javavbean进行获取值 -->

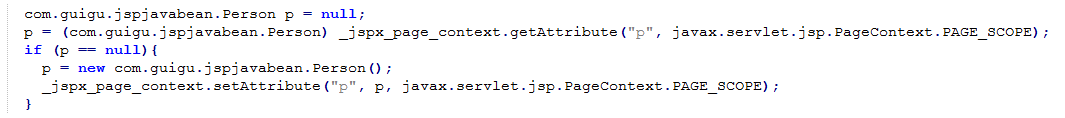
<jsp:getProperty property=*"id"* name=*"p"*/>

<jsp:getProperty property=*"name"* name=*"p"*/>

## <jsp:useBean>执行原理

<jsp:useBean id="p" class="com.guigu.jspjavabean.Person" scope="page"></jsp:useBean>

最终翻译为servlet代码如下：



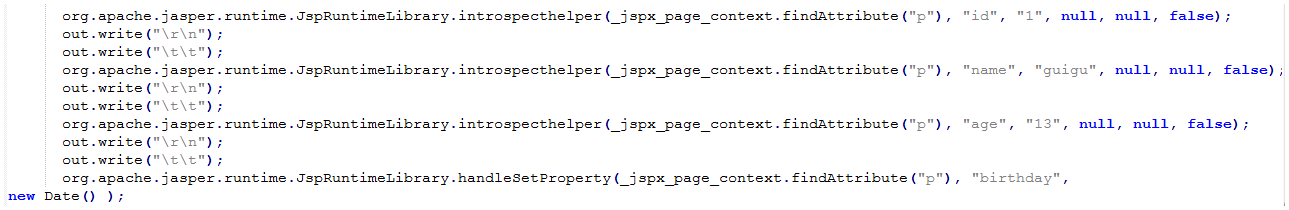
设置属性 最终翻译源码以下：

<jsp:setProperty property=*"id"* name=*"p"* value=*"1"*/>

<jsp:setProperty property=*"name"* name=*"p"* value=*"guigu"*/>

<jsp:setProperty property=*"age"* name=*"p"* value=*"13"*/>

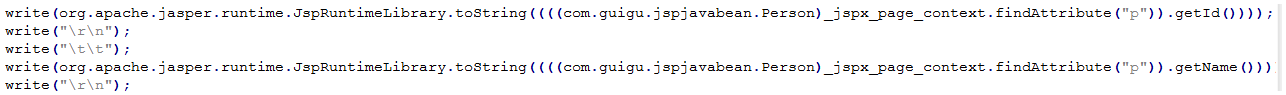
<jsp:setProperty property=*"birthday"* name=*"p"* value=*"*<%=**new** Date() %>*"*/>



获取属性最终翻译源码如下：

<jsp:getProperty property=*"id"* name=*"p"*/>

<jsp:getProperty property=*"name"* name=*"p"*/>



## JSP是开发模式

* SUN公司推出JSP技术后，同时也推荐了两种web应用程序的开发模式，一种是JSP+JavaBean模式，一种是Servlet+JSP+JavaBean模式。
* JSP+JavaBean模式适合开发业务逻辑不太复杂的web应用程序，这种模式下，JavaBean用于封装业务数据，JSP即负责处理用户请求，又显示数据。
* Servlet+JSP+JavaBean(MVC)模式适合开发复杂的web应用，在这种模式下，servlet负责处理用户请求，jsp负责数据显示，javabean负责封装数据。 Servlet+JSP、JavaBean模式程序各个模块之间层次清晰，web开发推荐采用此种模式。

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>简易网页计算器</title>

</head>

<body style="text-align: *center*;">

<jsp:useBean id=*"calculatorBean"* class=*"com.guigu.jspmvc.CalculatorBean"*></jsp:useBean>

<jsp:setProperty property=*"\*"* name=*"calculatorBean"*/>

<%

calculatorBean.calculate();

%>

计算结果是：

<jsp:getProperty property=*"firstNum"* name=*"calculatorBean"*/>

<jsp:getProperty property=*"operator"* name=*"calculatorBean"*/>

<jsp:getProperty property=*"secondNum"* name=*"calculatorBean"*/>

=

<jsp:getProperty property=*"result"* name=*"calculatorBean"*/>

<form action=*"/JavaWEB/jsp3/calc.jsp"* method=*"post"*>

<table border=*"1"* width=*"50%"*>

<tr>

<td colspan=*"2"*>简易计算器</td>

</tr>

<tr>

<td>第一个参数</td>

<td>

<input type=*"text"* name=*"firstNum"*>

</td>

</tr>

<tr>

<td>运算符</td>

<td>

<select name=*"operator"*>

<option value=*"+"*>+</option>

<option value=*"-"*>-</option>

<option value=*"\*"*>\*</option>

<option value=*"/"*>/</option>

</select>

</td>

</tr>

<tr>

<td>第二个参数</td>

<td>

<input type=*"text"* name=*"secondNum"*>

</td>

</tr>

<tr>

<td colspan=*"2"*>

<input type=*"submit"* value=*"计算"*>

</td>

</tr>

</table>

</form>

</body>

</html>

**package** com.guigu.jspmvc;

**import** java.math.BigDecimal;

/\*\*

\* <p>project\_name:JavaWEB</p>

\* <p>package\_name:com.guigu.jspmvc.CalculatorBean</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月3日上午8:43:19 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** CalculatorBean {

**private** **double** firstNum;

**private** **double** secondNum;

**private** **char** operator = '+';

**private** **double** result;

**public** **double** getFirstNum() {

**return** firstNum;

}

**public** **void** setFirstNum(**double** firstNum) {

**this**.firstNum = firstNum;

}

**public** **double** getSecondNum() {

**return** secondNum;

}

**public** **void** setSecondNum(**double** secondNum) {

**this**.secondNum = secondNum;

}

**public** **char** getOperator() {

**return** operator;

}

**public** **void** setOperator(**char** operator) {

**this**.operator = operator;

}

**public** **double** getResult() {

**return** result;

}

**public** **void** setResult(**double** result) {

**this**.result = result;

}

**public** **void** calculate() {

**switch** (operator) {

**case** '+':

result=firstNum+secondNum;

**break**;

**case** '-':

result=firstNum-secondNum;

**break**;

**case** '\*':

result=firstNum\*secondNum;

**break**;

**case** '/':

**if**(secondNum==0) {

**throw** **new** RuntimeException("被除数不能为0！！");

}

result=firstNum/secondNum;

result =**new** BigDecimal(result).setScale(2, BigDecimal.***ROUND\_HALF\_UP***).doubleValue();

**break**;

**default**:

**throw** **new** RuntimeException("对不起,您传入的参数非法");

}

}

}

# EL表达式和JSTL表达式

## EL表达式语言

* EL 全名为Expression Language。EL主要作用：
* 获取数据：
  + EL表达式主要用于替换JSP页面中的**脚本表达式，以从各种类型的web域 中检索java对象、**获取数据。(某个web域 中的对象，访问javabean的属性、访问list集合、访问map集合、访问数组)
* 执行运算：
  + 利用EL表达式可以在JSP页面中执行一些基本的关系运算、逻辑运算和算术运算，以在JSP页面中完成一些简单的逻辑运算。${user==null}
* 获取web开发常用对象
  + EL 表达式定义了一些隐式对象，利用这些隐式对象，web开发人员可以很轻松获得对web常用对象的引用，从而获得这些对象中的数据。
* 调用Java方法
  + EL表达式允许用户开发自定义EL函数，以在JSP页面中通过EL表达式调用Java类的方法。

### 获取数据：

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>EL表达式的使用</title>

</head>

<body>

<%

out.print(request.getAttribute("username"));

%>

<!--

${username} $代表从域对象中搜索数据

等价于

pageContext.findAttribute("username");

findAttribute 会依次在四大域对象中搜索数据 page--request--session--application

-->

${username}

<!-- 获取javabean对象的数据 -->

${person.id}<br/>

${person.name}<br/>

${person.age}<br/>

${person.address.name }<br/>

<!-- 获取list集合中的数据 -->

${list[1].name }<br/>

${list[1].age }<br/>

<!-- 获取map集合中的数据 -->

${map.a }<br/>

${map.b }<br/>

<!-- 利用el表达式获取web应用 -->

${pageContext.request.contextPath}

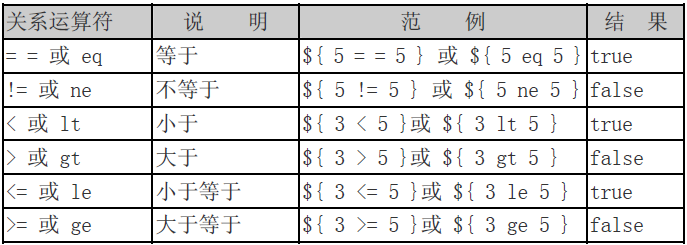
<a href=*"*${pageContext.request.contextPath}*/jsp3/calc.jsp"*>网页计算器</a>

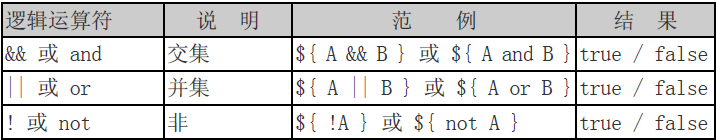
</body>

</html>

### 执行运算：

* 语法：${运算表达式}，EL表达式支持如下运算符：





* empty运算符：检查对象是否为null或“空”，**很好用！！！**
* 二元表达式：${user!=null?user.name : “”} ，**很好用！！！**
* [ ] 和 . 号运算符

**package** com.guigu.el;

/\*\*

\* <p>project\_name:JavaWEB</p>

\* <p>package\_name:com.guigu.el.User</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月3日上午10:34:42 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** User {

**private** **int** id;

**private** String name;

**private** **int** age;

**private** String gender;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

}

**package** com.guigu.el;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/ElServletDemo3")

**public** **class** ElServletDemo3 **extends** HttpServlet {

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

User user =**new** User();

user.setId(1);

user.setGender("female");

user.setName("张三");

request.setAttribute("user", user);

request.getRequestDispatcher("/el/el3.jsp").forward(request, response);

}

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

doGet(request, response);

}

}

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

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<form action=*""* method=*"post"*>

<table border=*"1"*>

<tr>

<td>用户名</td>

<td>

<input type=*"text"* name=*"name"* value=*"*${user.name}*"*>

</td>

</tr>

<tr>

<td>性别</td>

<td>

<input type=*"radio"* name=*"gender"* value=*"male"* ${user.gender=='male'?'checked':''}>男

<input type=*"radio"* name=*"gender"* value=*"female"* ${user.gender=='female'?'checked':''}>女

</td>

</tr>

</table>

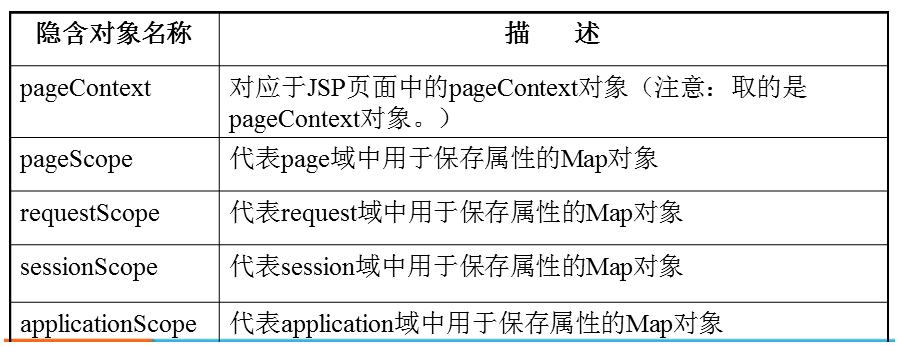
</form>

</body>

</html>

### 获取web开发常用对象

* EL表达式语言中定义了11个隐含对象，使用这些隐含对象可以很方便地获取web开发中的一些常见对象，并读取这些对象的数据。
* 语法：${隐式对象名称} ：获得对象的引用



**package** com.guigu.el;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/ElServletDemo4")

**public** **class** ElServletDemo4 **extends** HttpServlet {

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

request.setAttribute("username", "张三");

request.getSession().setAttribute("username", "李四");

request.getServletContext().setAttribute("username", "王五");

request.getRequestDispatcher("el/el4.jsp").forward(request, response);

}

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

doGet(request, response);

}

}

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<!-- 获取不用域对象中相同的key值的数据 -->

<!-- 默认从最小的域对象中取数据 -->

${username }<br/>

<!-- 获取指定的session对象的数据 -->

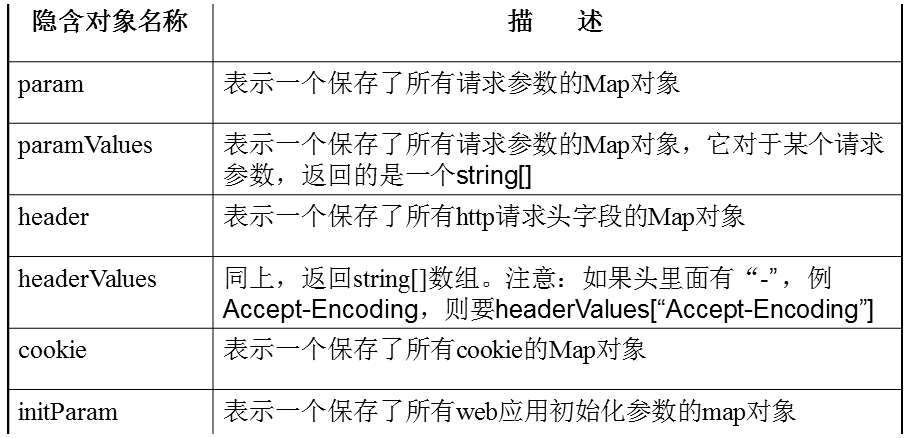
${requestScope.username } <br/>

${sessionScope.username }<br/>

${applicationScope.username } <br/>

</body>

</html>



## JSTL表达式

* 由于在JSP页面中显示数据时，经常需要对显示的字符串进行处理，SUN公司针对于一些常见处理定义了一套EL函数库供开发者使用。
* 这些EL函数在JSTL开发包中进行描述，因此在JSP页面中使用SUN公司的EL函数库，需要导入JSTL开发包，并在页面中导入EL函数库，如下所示：
  + 在页面中使用JSTL定义的EL函数：

**<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>**

<%@page import=*"java.util.Arrays"*%>

<%@page import=*"java.util.ArrayList"*%>

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

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

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/functions"* prefix=*"fn"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

${fn:toLowerCase("AbcDEF") }

<br/>

${fn:toUpperCase("AbcDEF")}

<br/>

${fn:length("abcde") }

<%

List list =Arrays.asList("1","2","4");

request.setAttribute("list1", list);

%>

${fn:length(list1) }

<br/>

${fn:split("aaa,bbb,cc",",")[1] }

<br/>

${fn:join(fn:split("www,hzguigu,com",","),".") }

</body>

</html>

利用jstl实现复选按钮的回显

user.setName("张三");

String [] likes= {"sing","dance","qdm"};

user.setLikes(likes);

<td>

<input type=*"checkbox"* name=*"likes"* value=*"sing"* ${fn:contains(fn:join(user.likes,","),"sing")?'checked':''}>唱歌

<input type=*"checkbox"* name=*"likes"* value=*"dance"* ${fn:contains(fn:join(user.likes,","),"dance")?'checked':''}>跳舞

<input type=*"checkbox"* name=*"likes"* value=*"music"* ${fn:contains(fn:join(user.likes,","),"music")?'checked':''}>音乐

<input type=*"checkbox"* name=*"likes"* value=*"game"* ${fn:contains(fn:join(user.likes,","),"game")?'checked':''}>游戏

<input type=*"checkbox"* name=*"likes"* value=*"qdm"* ${fn:contains(fn:join(user.likes,","),"qdm")?'checked':''}>敲代码

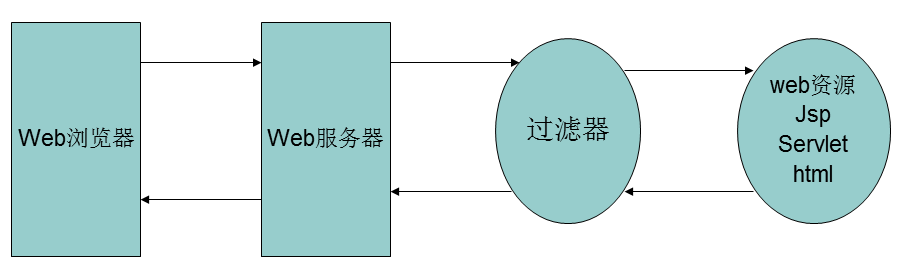
</td>

</tr>

# 过滤器

## 基础知识

* Filter也称之为过滤器，它是Servlet技术中最激动人心的技术，WEB开发人员通过Filter技术，对web服务器管理的所有web资源：例如Jsp, Servlet, 静态图片文件或静态 html 文件等进行拦截，从而实现一些特殊的功能。例如实现URL级别的权限访问控制、过滤敏感词汇、压缩响应信息等一些高级功能。
* Servlet API中提供了一个Filter接口，开发web应用时，如果编写的Java类实现了这个接口，则把这个java类称之为过滤器Filter。通过Filter技术，开发人员可以实现用户在访问某个目标资源之前，对访问的请求和响应进行拦截，如下所示：



* Filter接口中有一个doFilter方法，当开发人员编写好Filter，并配置对哪个web资源进行拦截后，WEB服务器每次在调用web资源的service方法之前，都会先调用一下filter的doFilter方法，因此，在该方法内编写代码可达到如下目的：
  + 调用目标资源之前，让一段代码执行
  + 是否调用目标资源（即是否让用户访问web资源）。
    - web服务器在调用doFilter方法时，会传递一个filterChain对象进来，filterChain对象是filter接口中最重要的一个对象，它也提供了一个doFilter方法，开发人员可以根据需求决定是否调用此方法，调用该方法，则web服务器就会调用web资源的service方法，即web资源就会被访问，否则web资源不会被访问。
  + 调用目标资源之后，让一段代码执行

filterChain可以决定是否放行到目标资源

* Filter开发分为二个步骤：
  + 编写java类实现Filter接口，并实现其doFilter方法。
  + 在 web.xml 文件中使用<filter>和<filter-mapping>元素对编写的filter类进行注册，并设置它所能拦截的资源。（动手实验）
* Filter链
  + 在一个web应用中，可以开发编写多个Filter，这些Filter组合起来称之为一个Filter链。
  + web服务器根据Filter在web.xml文件中的注册顺序，决定先调用哪个Filter，当第一个Filter的doFilter方法被调用时，web服务器会创建一个代表Filter链的FilterChain对象传递给该方法。在doFilter方法中，开发人员如果调用了FilterChain对象的doFilter方法，则web服务器会检查FilterChain对象中是否还有filter，如果有，则调用第2个filter，如果没有，则调用目标资源。
  + Filter链实验（查看filterChain API文档）

**package** com.guigu.filter;

**import** java.io.IOException;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

/\*\*

\* <p>project\_name:JavaFilter</p>

\* <p>package\_name:com.guigu.filter.SimpleFilter1</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月4日下午2:26:47 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** SimpleFilter1 **implements** Filter {

@Override

**public** **void** init(FilterConfig filterConfig) **throws** ServletException {

System.***out***.println("进行初始化过滤器1....");

}

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)**throws** IOException, ServletException {

System.***out***.println("在访问目标资源之前执行某些事情.........aaaa");

//决定是否放行到目标资源

chain.doFilter(request, response);

System.***out***.println("在访问目标资源之后执行某些事情......bbbb");

}

@Override

**public** **void** destroy() {

}

}

**package** com.guigu.filter;

**import** java.io.IOException;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

/\*\*

\* <p>project\_name:JavaFilter</p>

\* <p>package\_name:com.guigu.filter.SimpleFilter2</p>

\* <p>description：</p>

\* <p>@author：刘老师<p>

\* <p> date:2018年1月4日下午2:36:55 </p>

\* <p>comments： </p>

\* <p>@version jdk1.8</p>

\*

\* <p>Copyright (c) 2018, 980991634@qq.com All Rights Reserved. </p>

\*/

**public** **class** SimpleFilter2 **implements** Filter {

@Override

**public** **void** init(FilterConfig filterConfig) **throws** ServletException {

System.***out***.println("进行初始化过滤器2....");

}

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

**throws** IOException, ServletException {

System.***out***.println("在访问目标资源之前执行某些事情.........cccc");

// 决定是否放行到目标资源

// chain.doFilter(request, response);

System.***out***.println("在访问目标资源之后执行某些事情......ddddd");

}

@Override

**public** **void** destroy() {

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"* id=*"WebApp\_ID"* version=*"3.1"*>

<display-name>JavaFilter</display-name>

<filter>

<filter-name>filter1</filter-name>

<filter-class>com.guigu.filter.SimpleFilter1</filter-class>

</filter>

<filter-mapping>

<filter-name>filter1</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter>

<filter-name>filter2</filter-name>

<filter-class>com.guigu.filter.SimpleFilter2</filter-class>

</filter>

<filter-mapping>

<filter-name>filter2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

</web-app>

**package** com.guigu.filter;

**import** java.io.IOException;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebFilter;

@WebFilter("/\*")

**public** **class** SimpleFilter3 **implements** Filter {

**public** **void** destroy() {

}

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain) **throws** IOException, ServletException {

System.***out***.println("hello filter3....");

chain.doFilter(request, response);

System.***out***.println("hello filter3333....");

}

**public** **void** init(FilterConfig fConfig) **throws** ServletException {

}

}

## 过滤器的应用

### 解决全站乱码问题

**package** com.guigu.filter;

**import** java.io.IOException;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebFilter;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebFilter("/\*")

**public** **class** CharseterEncodingFilter **implements** Filter {

**private** FilterConfig config=**null**;

**private** String defaultCharset="UTF-8";

**public** **void** destroy() {

}

**public** **void** doFilter(ServletRequest req, ServletResponse res, FilterChain chain) **throws** IOException, ServletException {

HttpServletRequest request =(HttpServletRequest) req;

HttpServletResponse response=(HttpServletResponse) res;

//得到系统指定的编码集

String charset=config.getInitParameter("charset");

**if**(charset==**null**) {

charset=defaultCharset;

}

request.setCharacterEncoding(charset);

response.setCharacterEncoding(charset);

response.setContentType("text/html;charset="+charset);

chain.doFilter(request, response);

}

**public** **void** init(FilterConfig fConfig) **throws** ServletException {

**this**.config=fConfig;

}

}

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"CharsetServlet"* method=*"post"*>

城市:<input type=*"text"* name=*"city"*>

<input type=*"submit"* value=*"提交"*>

</form>

</body>

</html>

**package** com.guigu.servlet;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/CharsetServlet")

**public** **class** CharsetServlet **extends** HttpServlet {

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

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

System.***out***.println(city);

}

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

doGet(request, response);

}

}

### 脏话过滤器

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

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"*${pageContext.request.contextPath }*/DirtyServlet"* method=*"post"*>

留言:<br/>

<textarea rows=*"8"* cols=*"60"* name=*"message"*></textarea>

<input type=*"submit"* value=*"发表"*>

</form>

</body>

</html>

**package** com.guigu.servlet;

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/DirtyServlet")

**public** **class** DirtyServlet **extends** HttpServlet {

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

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

response.getWriter().write("您上次的留言是:<br/>"+message);

}

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

// **TODO** Auto-generated method stub

doGet(request, response);

}

}

**package** com.guigu.filter;

**import** java.io.IOException;

**import** java.util.Arrays;

**import** java.util.List;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebFilter;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletRequestWrapper;

**import** javax.servlet.http.HttpServletResponse;

@WebFilter("/\*")

**public** **class** DirtyFilter **implements** Filter {

**public** **void** destroy() {

}

**public** **void** doFilter(ServletRequest req, ServletResponse res, FilterChain chain) **throws** IOException, ServletException {

// 从数据库加载脏话字典

HttpServletRequest request =(HttpServletRequest) req;

HttpServletResponse response =(HttpServletResponse) res;

DirtyRequest dirtyRequest =**new** DirtyRequest(request);

chain.doFilter(dirtyRequest, response);

}

**public** **void** init(FilterConfig fConfig) **throws** ServletException {

}

}

//利用装饰者模式 对request进行增强

**class** DirtyRequest **extends** HttpServletRequestWrapper{

**private** List<String> dirtyWords=Arrays.*asList*("龟孙","傻子","畜生","王八");

**private** HttpServletRequest request;

**public** DirtyRequest(HttpServletRequest request) {

**super**(request);

**this**.request=request;

}

@Override

**public** String getParameter(String name) {

//获取用户的请求参数

String value =**this**.request.getParameter(name);

//把获取的value的数据进行替换即可

**if**(value==**null**) {

**return** **null**;

}

**for** (String str : dirtyWords) {

**if**(value.contains(str)) {

value=value.replace(str, "\*\*\*\*\*");

}

}

**return** value;

}

}

### HTML转义

<script type=*"text/javascript"*>

**while**(**true**){

alert("这个网站是个傻子");

}

</script>