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。

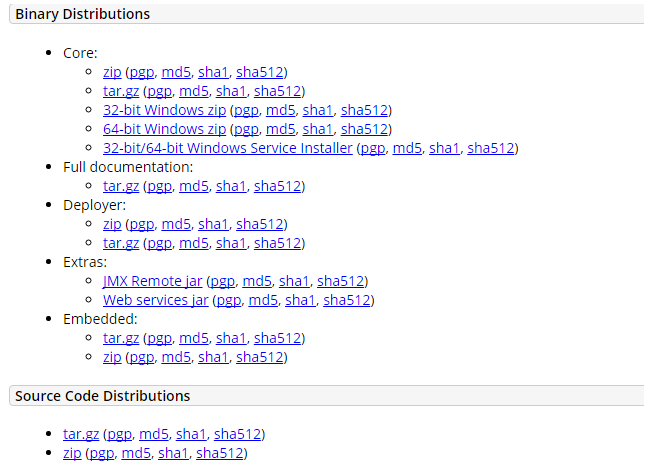
## 常见的web服务器

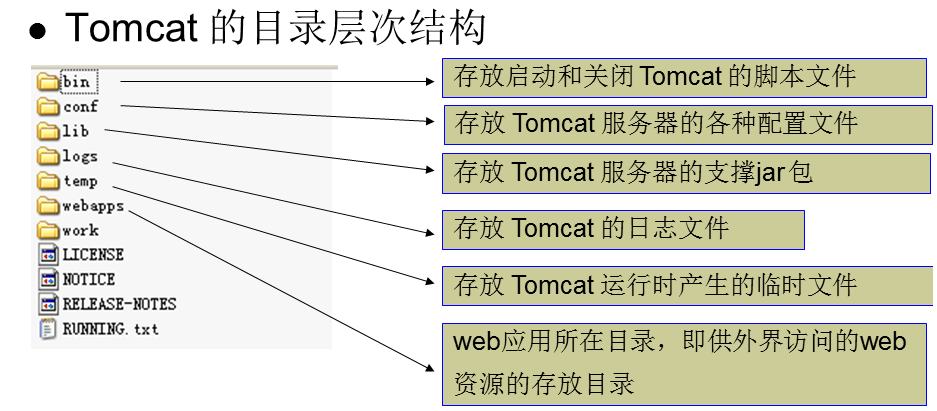
* WebLogic是BEA公司的产品，是目前应用最广泛的Web服务器，支持J2EE规范，而且不断的完善以适应新的开发要求，启动界面如图
* 另一个常用的Web服务器是IBM公司的WebSphere，支持J2EE规范，启动界面如图
* **在小型的应用系统或者有特殊需要的系统中，可以使用一个 免费的Web服务器：Tomcat，该服务器支持全部JSP以及Servlet规范，启动界面如图**

## 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**



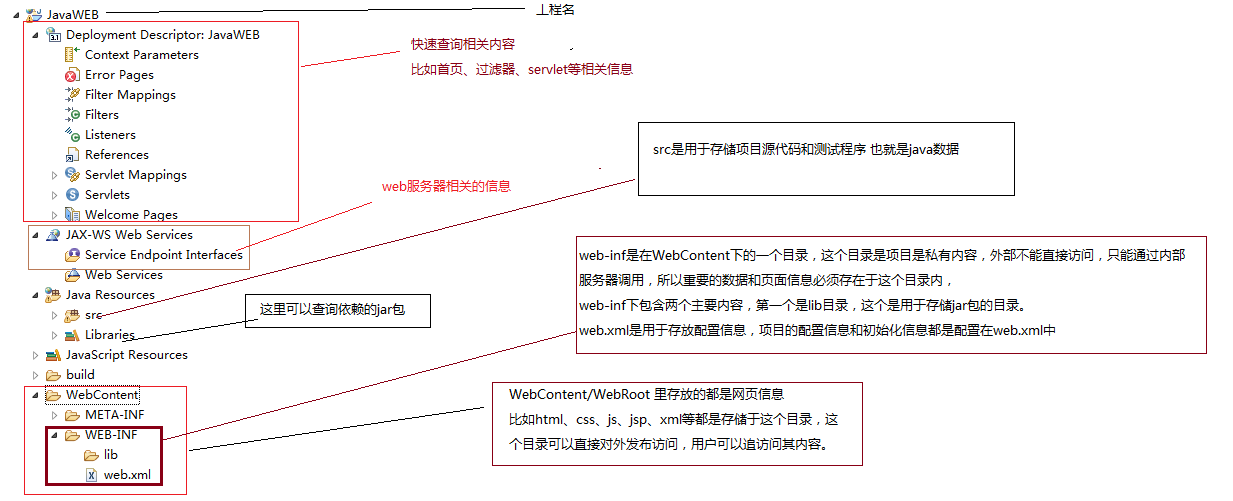


Eclipseee中配置tomcat

# 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

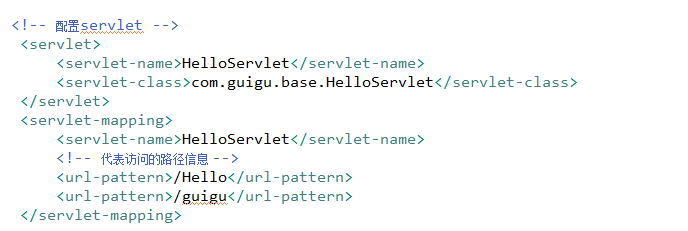
**protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

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**

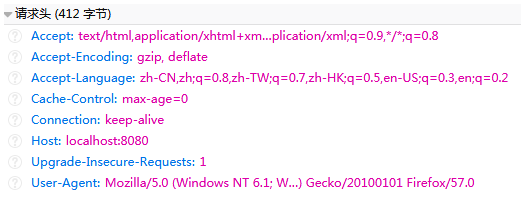


## Servlet的运行流程

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**

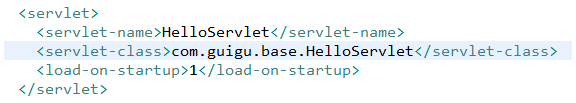
Servlet的的声明周期分为三个阶段，1.初始化阶段。2响应客户阶段。3.终止阶段

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

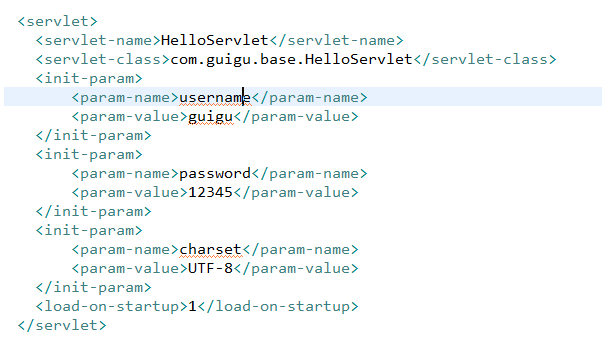
当前终止的时候回调用destroy方法进行终止。

**连接源码**

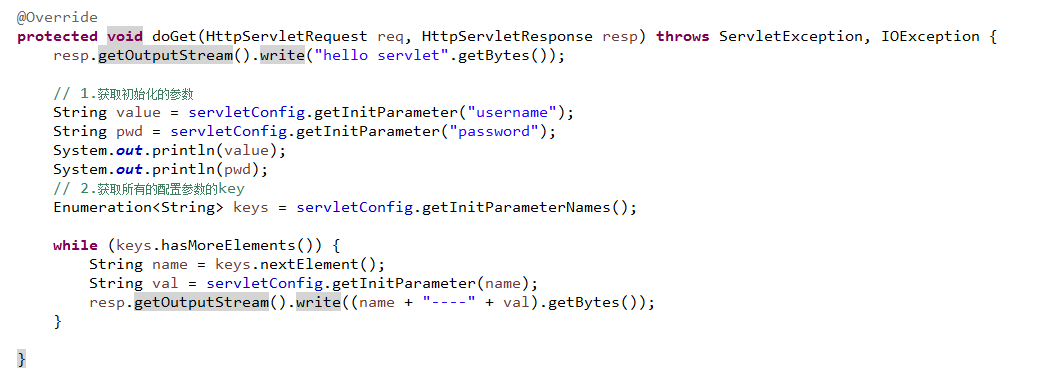
## Servlet相关的对象

### 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**



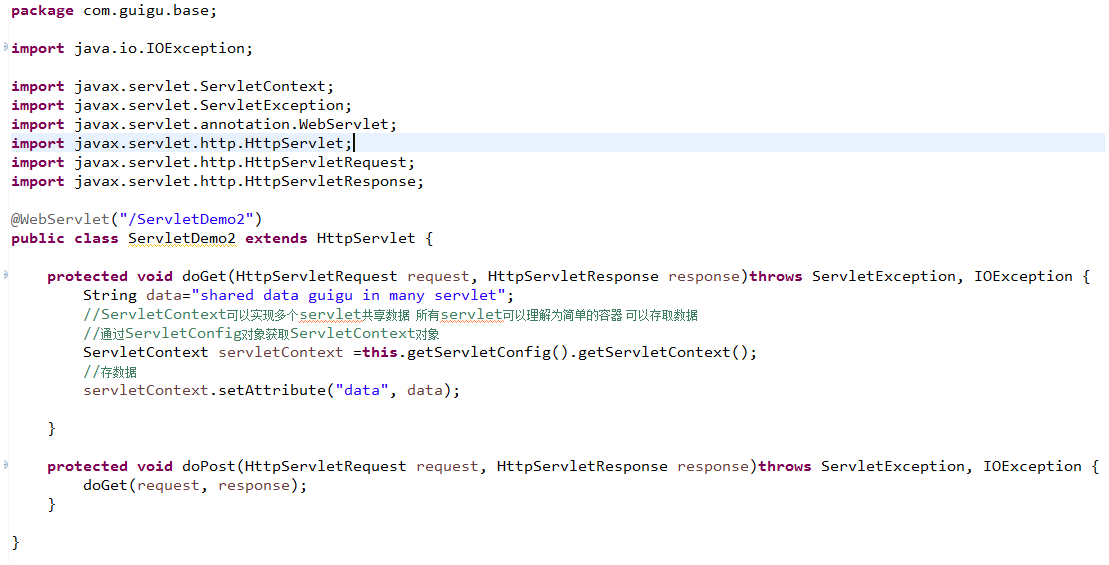
方式二在注解开发中使用配置参数



### ServletContext

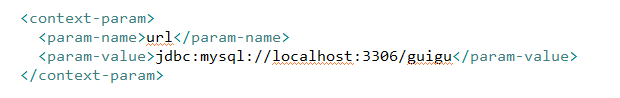
* WEB容器在启动时，它会为每个WEB应用程序都创建一个对应的ServletContext对象，它代表当前web应用。
* ServletConfig对象中维护了ServletContext对象的引用，开发人员在编写servlet时，可以通过ServletConfig.getServletContext方法获得ServletContext对象。
* 由于一个WEB应用中的所有Servlet共享同一个ServletContext对象，因此Servlet对象之间可以通过ServletContext对象来实现通讯。ServletContext对象通常也被称之为**context域对象**。
* 查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

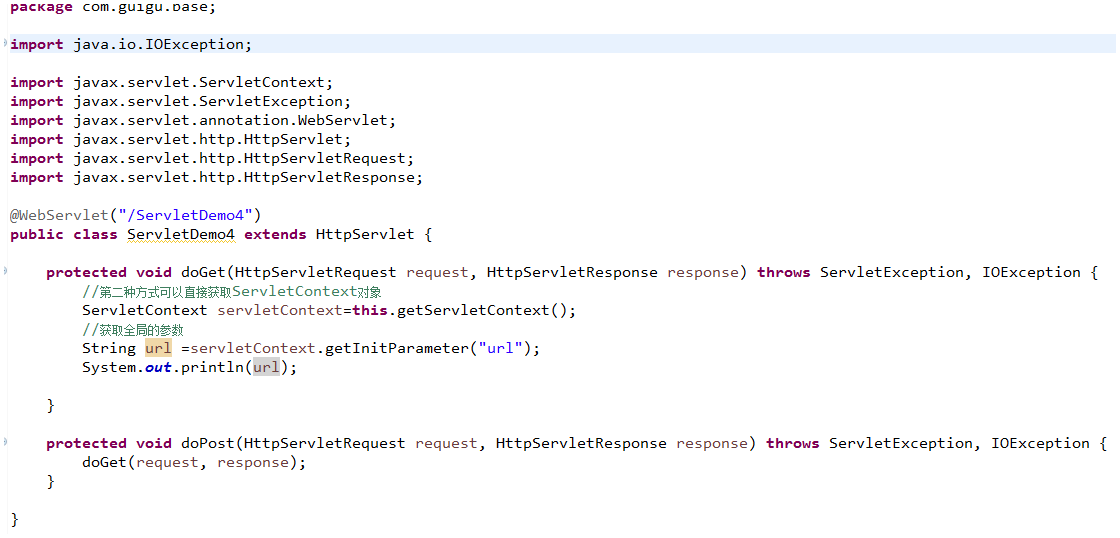
**第一个案例 多个servlet共享数据**



**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

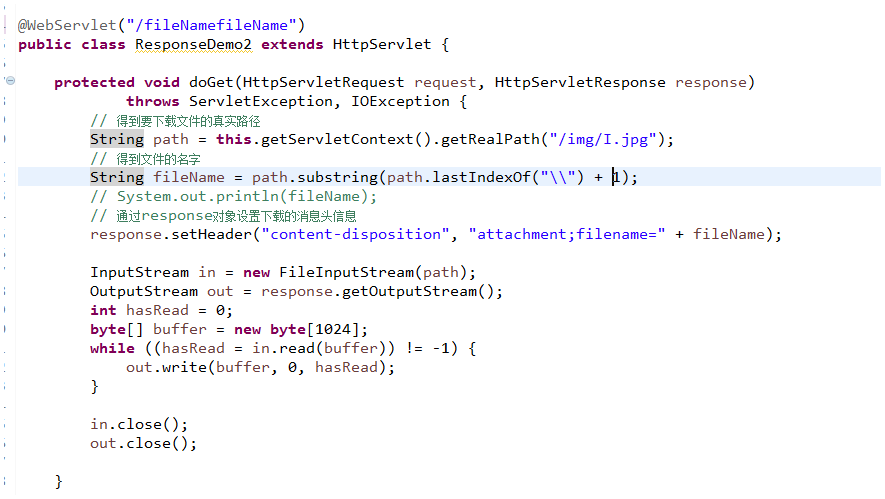
* Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的request对象、和代表响应的response对象。
* request和response对象即然代表请求和响应，那我们要获取客户机提交过来的数据，只需要找request对象就行了。要向客户机输出数据，只需要找response对象就行了。

### 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**">

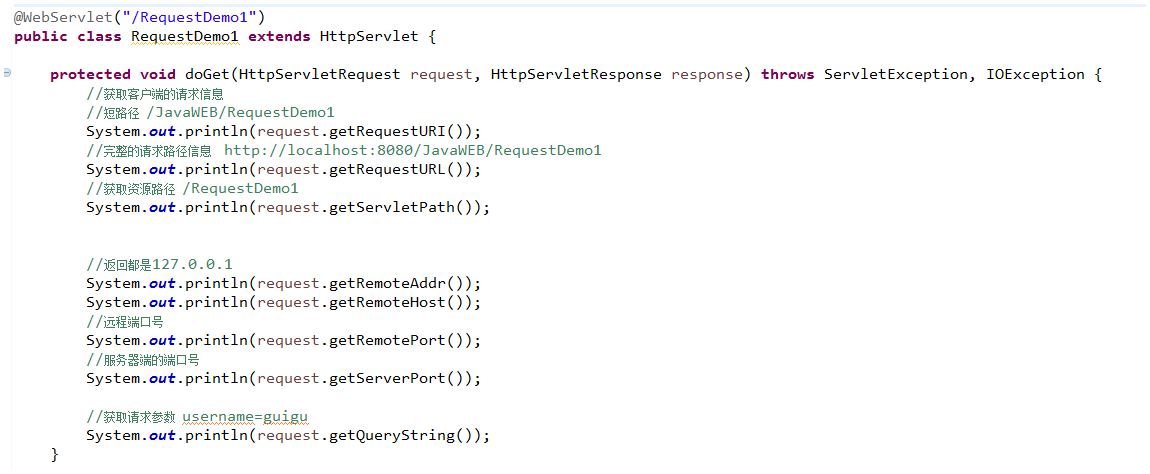
<input type=*"button"* value=*"换个图片"* onclick="reloadImage()" id=*"btn"*>

</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 {

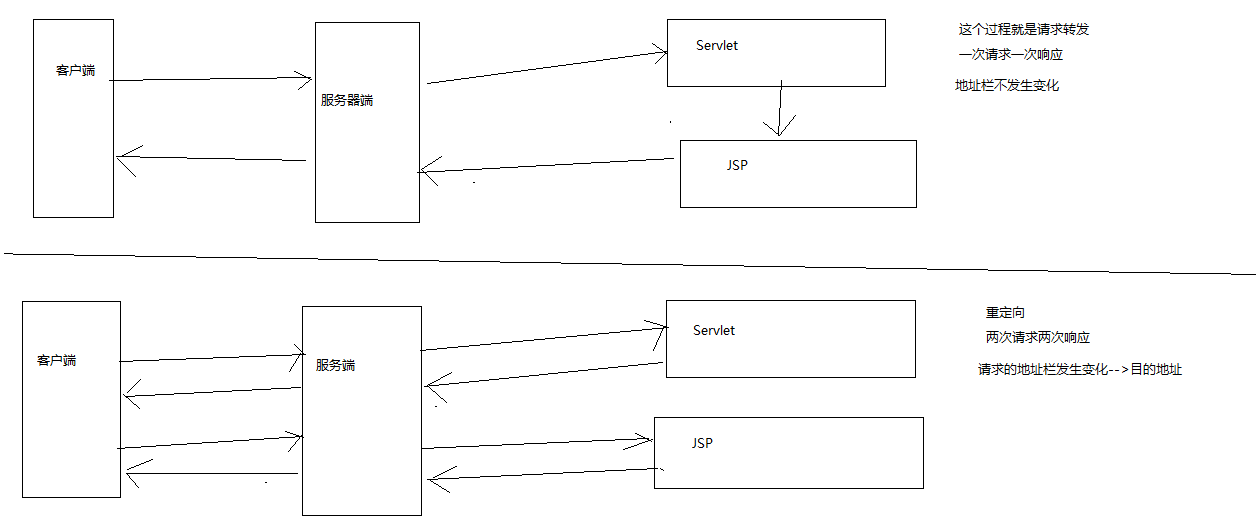
doGet(request, response);

}

}

### 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。

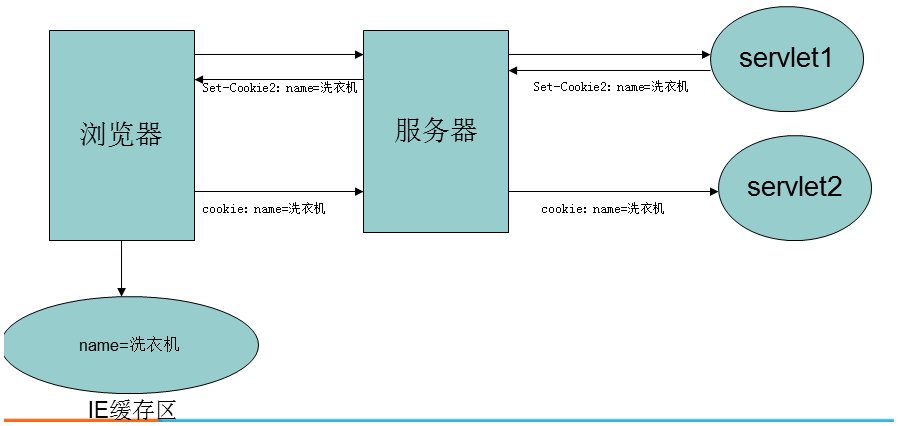
# 会话管理

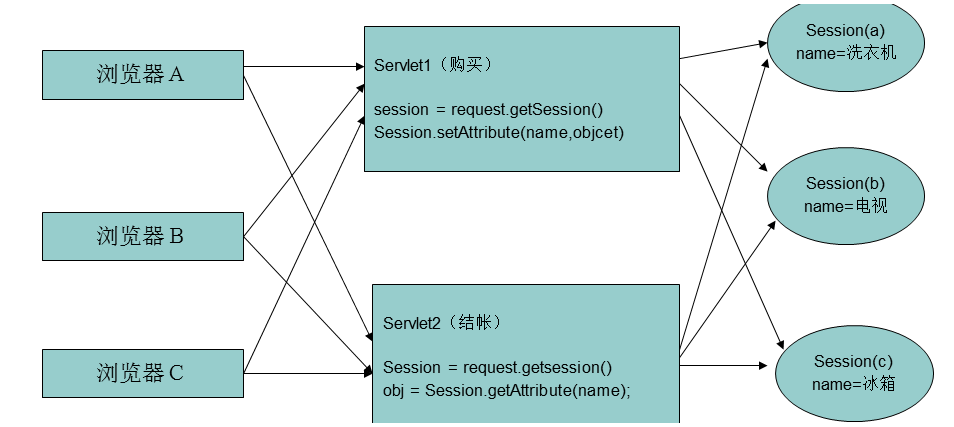
* 什么是会话？
  + 会话可简单理解为：用户开一个浏览器，点击多个超链接，访问服务器多个web资源，然后关闭浏览器，整个过程称之为一个会话。
* 会话过程中要解决的一些问题？
  + 每个用户在使用浏览器与服务器进行会话的过程中，不可避免各自会产生一些数据，程序要想办法为每个用户保存这些数据。
  + 例如：用户点击超链接通过一个servlet购买了一个商品，程序应该想办法保存用户购买的商品，以便于用户点结帐servlet时，结帐servlet可以得到用户购买的商品为用户结帐。
  + 思考：用户购买的商品保存在request或servletContext中行不行？

***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);

}

}

**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