Servlet注册 url-pattern支持的语法

使用通配符 \*

类似\*.do

类似/\* 匹配任意内容，包括静态资源（引起静态资源不能正常访问）

/ 和 /\*类似，不会匹配jsp资源

Servlet的生命周期

创建、初始化、service、销毁

默认情况下，第一次访问servlet时，创建对象

<load-on-startup>

Servlet的线程安全

同步代码块

使用支持线程安全的结构存储数据

ServletConfig 获取servlet中配置的初始化参数，每个servlet对应一个ServletConfig对象

ServletContext 一个web应用对象ServletContext对象

获取web应用的路径

获取应用的初始化参数

作为域对象，存储数据，实现资源间数据的共享

转发和重定向

转发 request.getRequestDiapatcher().forward()

重定向 response.sendRedirect()

# 一 什么是会话

## 1 生活中会话

A:商量个事儿呗

B：嘛事

A：借点钱呗

B: 喂…喂…我这里信号不好…

## 2 软件中会话

打开浏览器-》访问应用-》关闭浏览器

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，服务器要想办法为每个用户保存这些数据。

例如：多个用户点击超链接通过一个servlet各自购买了一个商品，服务器应该想办法把每一个用户购买的商品保存在各自的地方，以便于这些用户点结帐servlet时，结帐servlet可以得到用户各自购买的商品为用户结帐。

## 3 会话技术

1)Cookie:将数据保存在浏览器客户端的技术

服务器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

2)Session:将数据保存在服务端的技术

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

# 二 Cookie

## 1 原理

1. 服务端创建cookie对象
2. 发送cookie信息到浏览器

例如，响应头中增加 Set-Cookie: name=zhangsan

1. 浏览器将得到的cookie信息保存在浏览器端
2. 通过浏览器下次访问web应用时，请求数据中会带上cookie信息例如，请求头中 Cookie: name=zhangsan
3. 服务器端收到浏览器发送的cookie信息

## 2 使用

Cookie(String name, String value) 创建cookie对象的构造方法，value是字符串类型，不能使用中文

public String getName() 取得Cookie的名字

public String getValue() 取得Cookie的值

public void setValue(String newValue)设置Cookie的值

public void setMaxAge(int expiry)设置Cookie的最大保存时间，单位秒，即cookie的有效期，默认会话结束，cookie失效

public int getMaxAge()获取Cookies的有效期

public void setPath(String uri) 设置cookie的有效路径，比如把cookie的有效路径设置为"/aaa"，那么浏览器访问"aaa"的web资源时，就会带上cookie

public String getPath() 获取cookie的有效路径

**public** **void** cookie1(HttpServletRequest request, HttpServletResponse response){

//创建cookie对象

Cookie cookie = **new** Cookie("name", "zhangsan");

//如果有相同name值得cookie，后一个会替换前一个cookie内容

//Cookie cookie1 = new Cookie("name", "lisi");

//cookie中的name和value都是字符串类型

//Cookie cookie1 = new Cookie("age", "10");

//cookie中不能出现中文

//Cookie cookie1 = new Cookie("name", "张三");

//通过响应将cookie数据发送到浏览器端

response.addCookie(cookie);

//response.addCookie(cookie1);

//获取请求中的cookie数据

//ctrl + 1，Enter

//ctrl + 2, l

Cookie[] cookies = request.getCookies();

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

**for** (Cookie cookie2 : cookies) {

// 获取cookie的name和对应的value

System.***out***.println(cookie2.getName());

System.***out***.println(cookie2.getValue());

}

}

}

**public** **void** cookieTime(HttpServletRequest request, HttpServletResponse response){

Cookie cookie = **new** Cookie("name", "zhangsan");

//设置cookie的过期时间

//单位秒

// cookie.setMaxAge(60 \* 60);

//可以实现删除已有的cookie

// cookie.setMaxAge(0);

//过期时间：会话结束时

cookie.setMaxAge(-1);

response.addCookie(cookie);

}

**public** **void** cookiePath(HttpServletRequest request, HttpServletResponse response){

Cookie cookie = **new** Cookie("name", "lisi");

//路径不同，name相同，不会覆盖原来的cookie

//实现将cookie传给另一个web应用

//设置cookie的路径

cookie.setPath("/W08\_3\_T");

cookie.setMaxAge(60 \* 60);

response.addCookie(cookie);

}

## 3 特点

一个cookie不能超过4k;一个浏览器最多存300个cookie；一个站点最多存20个cookie

Cookie只能存字符串数据，不支持中文

Tomcat9中，cookie中不能有空格，会报错

# 三 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），可用用于保存客户端用户的访问状态。注意：**一个浏览器独占一个session对象**(默认情况下)。

Session依赖于cookie

Session也属于域对象

## 1 原理

1. 访问应用时，如果创建session，每个session对象会分配一个id,称谓JSESSIONID
2. 服务器会将session的id以cookie的形式发送给浏览器端，浏览器端会存储该id
3. 以后再访问应用时，浏览器将session的id发送给服务端
4. 服务端收到session的id后，会根据id查找对应的session对象，如果没有找到，创建一个新的session对象

## 2 使用

常用方法：

request.getSession() 获取session对象，如果不存在，创建

public void setAttribute(String name,String value)设定指定名字的属性的值，并将它添加到session会话范围内

public Object getAttribute(String name)在会话范围内获取指定名字的属性的值，返回值类型为object，如果该属性不存在，则返回null。

public void removeAttribute(String name)，删除指定名字的session属性

public void invalidate（），使session失效。可以立即使当前会话失效，原来会话中存储的所有对象都不能再被访问。

public String getId( )，获取当前的会话ID。每个会话在服务器端都存在一个唯一的标示sessionID，session对象发送到浏览器的唯一数据就是sessionID，它一般存储在cookie中。

public void setMaxInactiveInterval(int interval) 设置会话的最大持续时间，单位是秒，负数表明会话永不失效。

public int getMaxInActiveInterval（）,获取会话的最大持续时间。

//会根据session的id 查找服务端是否有对应的session对象，如果有，返回session对象;如果没有创建session对象

//创建session对象后，会将JSESSIONID以cookie的形式存在浏览器端

HttpSession session = request.getSession();

//session的id，唯一的 JSESSIONID

String id = session.getId();

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

//默认情况下，JSESSIONID的过期时间是会话结束时，为了能正常访问session对象，需要手动将JSESSIONID存储到cookie中，并设置一个合适的过期时间

Cookie cookie = **new** Cookie("JSESSIONID", id);

cookie.setMaxAge(30 \* 60);

response.addCookie(cookie);

//设置session的过期时间，单位秒，默认30分钟

session.setMaxInactiveInterval(10 \* 60);

//向session域中存入数据

session.setAttribute("name", "张三");

-----------------------

//根据JSESSIONID找session对象，如果存在，返回；如果不存在，返回null

HttpSession session = request.getSession(**false**);

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

//获取域中的数据

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

//获取域中所有的属性值

//session.getAttributeNames();

}

访问一个资源时，如果没有登录过，跳转到登录界面。如果登录过，在一定时间内，可以直接访问该资源

## 3 禁用cookie的处理

session依赖cookie，因为服务器需要根据sessionId，然后找到客户端的session对象，如果浏览器禁用了cookie，就需要对URL进行重写，这样服务器收到的请求中就会带有sessionId。

使用方法：response.encodeURL(String url)

它会判断客户端浏览器是否禁用了Cookie，如果禁用了，那么这个方法会在URL后面追加jsessionid，否则不会追加。