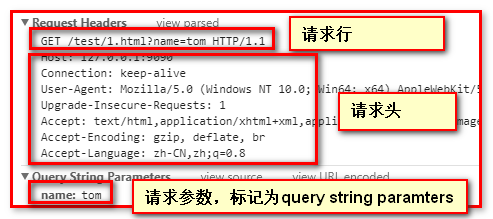
HTTP（超文本传输协议）永远是客户端（浏览器或者手机等）发送请求，服务器回送响应，其请求方式主要有GET，POST。

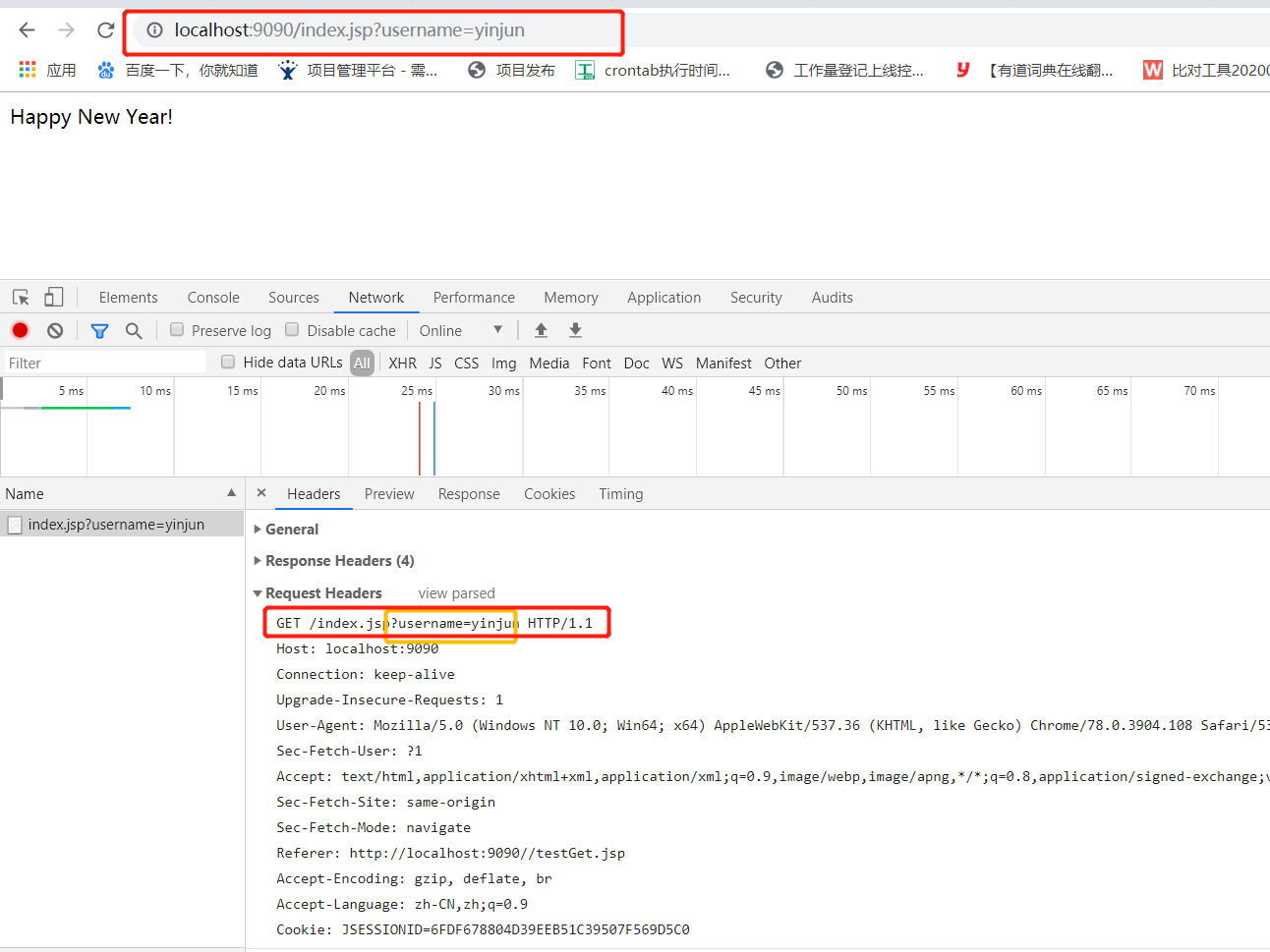
* 1. GET：请求行，请求头

A标签和直接地址栏输入都是get请求

----暴露请求参数在地址栏不安全

-----有长度限制，所谓的请求长度限制是由浏览器和 web 服务器决定和设置的

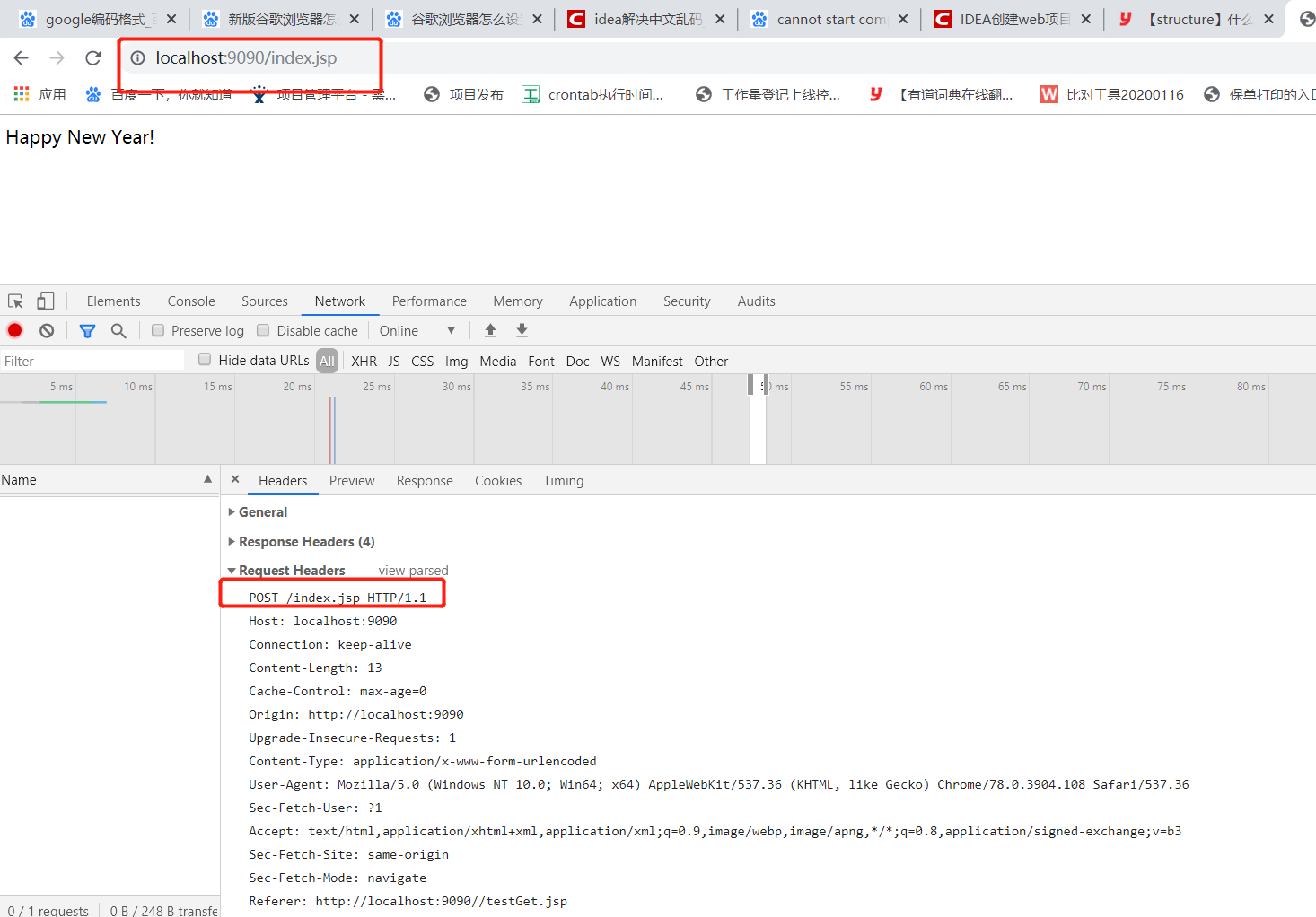


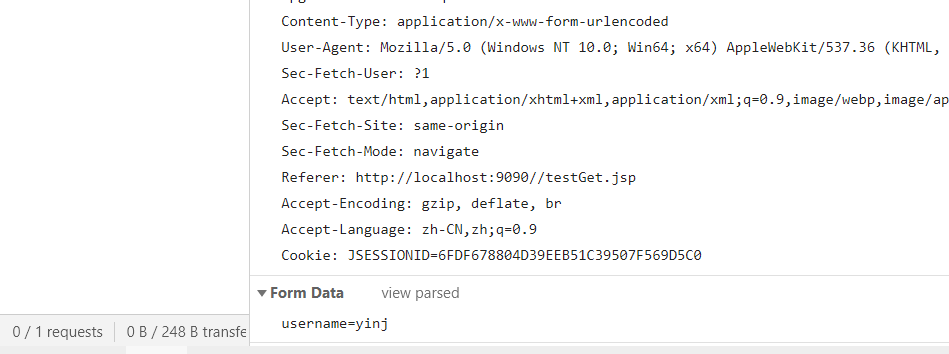


* 1. POST：请求行，请求头，请求体；参数放于请求体中

使用form表单method=post就是post请求







# 2、

响应行中的各种响应码含义：

200 - 请求成功

302 – 请求重定向 （response）

404 - 请求的资源（网页等）不存在

500 - 内部服务器错误（如果遇到500的情况，表示的是java代码出错误）

# 3、

（1）地址栏输入中文出错，在tomcat servert.xml配置端口处加上URIEncoding="utf8"

1. 浏览器输入中文汉字，java控制台输入乱码，

req.setCharacterEncoding(**"utf-8"**);

从页面上获取中文不会乱码了。

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

向页面上返回中文不会乱码了。

1. servlet的生命周期

生命周期：是说一个对象从创建到销毁的整个过程。称为整个对象的一个生命周期。

Servlet生命周期，根据功能不同，大致分为三个阶段，分别是 初始化阶段，运行阶段（最频繁） 和 销毁阶段。

1. 初始化阶段：当客户端发出请求到tomcat时，tomcat看内存里有没有sevlet对象。有：直接用；没有：调用init方法实现初始化。注意：整个过程init只调用一次。
2. 运行阶段：就是调用service()方法阶段，而doget和dopost都是service的子类。
3. 销毁阶段：当服务器关闭时，servlet会随着Web应用的销毁而销毁。
4. servletconfig的学习
   1. 每一个servlet都跟着一个独立的servletconfig对象，通过在xml中进行配置，servlet初始化调用init方法时创建servletconfig对象。

（1）getInitParameter、 getInitParameterNames用于获取Web.xml中的参数名、参数值。

（2）getServletName 获取 Web.xml中的 Servlet-name。

 （3）ServletContext.getServletContext();

(b)要在init方法中传入configsertvlet,并为servletconfig全局变量赋值，传入的值就是xml中的值。

# Servletcontext

相当于一个全局变量，整个工程共用，比如统计登录人数。

1. getAttribute()，setAttribute(String,Object)，removeAttribute()
2. ServletContext aServletContext = this.getServletContext();

由于在init方法里没有写**super**.init(servletConfig);

导致this.getServletContext();值为空。 如果不写，用reauest.getServletContext();可以得到值。

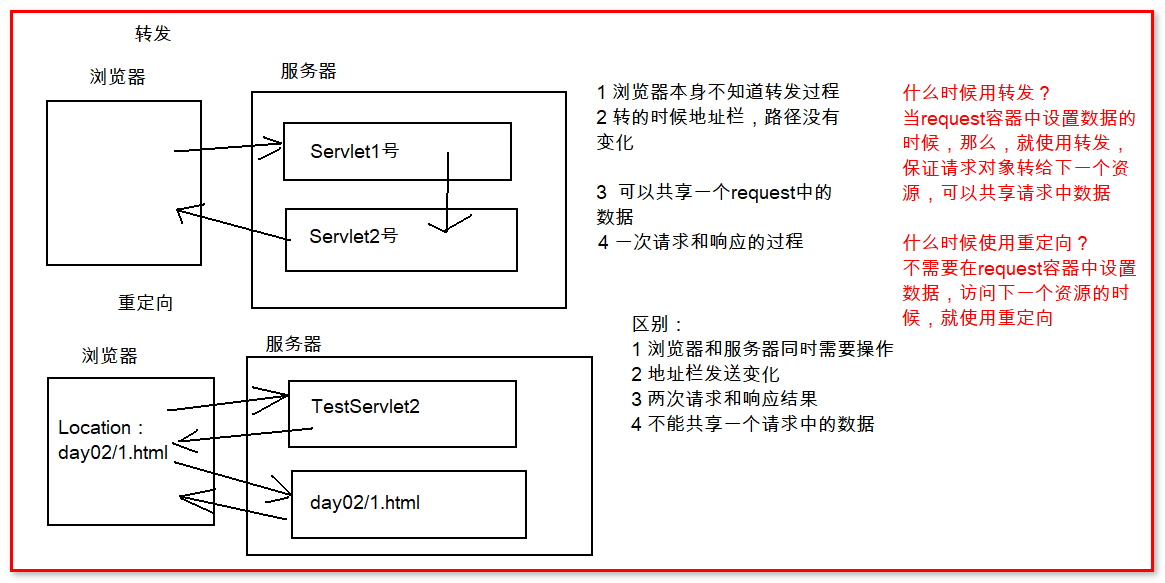
（3）响应重定向和请求转发的区别：

1 转发操作发生在服务器内部，重定向是浏览器执行操作

2 转发地址栏不变，重定向，地址栏变化（只有浏览器自己请求的页面，地址栏才会显示）

3 转发在一次请求中完成，重定向是两次请求

4 转发可以在一次请求中共享数据，重定向不行。



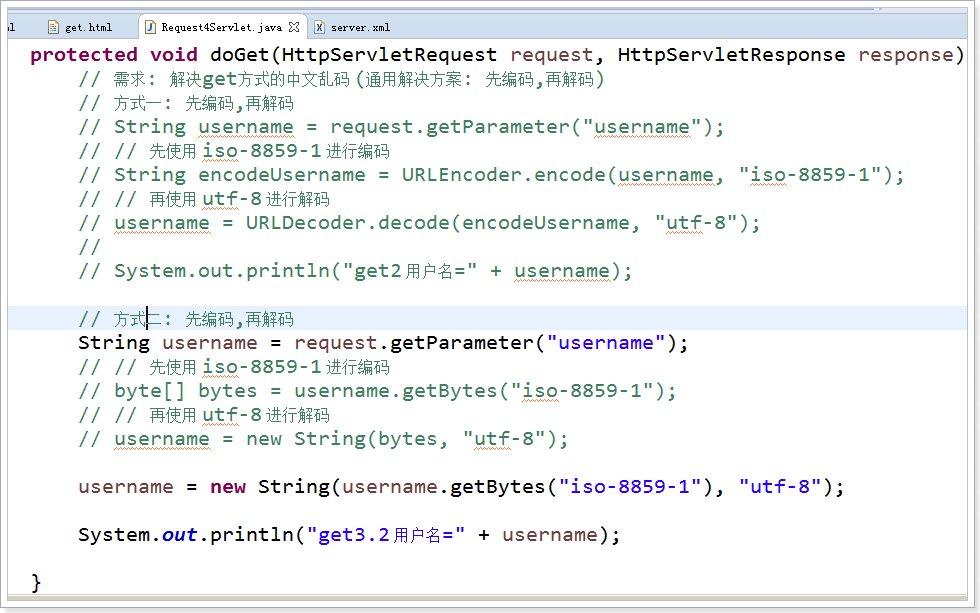
（5）String requestURI = req.getRequestURI();  
StringBuffer requestURL = req.getRequestURL();

requestURI = /registerTest

requestURL = <http://localhost:9090/registerTest>

# Request

### Get 提交的中文乱码解决：



### POST提交的中文乱码解决：

req.setCharacterEncoding(**"utf-8"**);

从页面上获取中文不会乱码了。

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

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

***Request.setAttribute(“User”,user);***

1 使用request对象，可以获取请求行

2 使用request对象，可以获取请求头

3 使用request对象，可以处理中文乱码

4 使用request对象，调用下一个servlet（请求转发）

5 使用request对象，在一个请求转发过程中，让两个servlet共享数据。

2、Beanutils

# Cookie(浏览器端)

问题1: cookie可以记录数据，但是数据需要一直记录永远不删除吗？

答：数据不删除，数据量越来越大，影响存储空间，数据量越大网络传输cookie慢，服务器解析cookie，也是效率不高。

问题2：那么如何删除cookie数据？

答：我们在服务器创建cookie给浏览器，但是我们无法操作浏览器，因此，在cookie创建的时候设置cookie的生存时间，时间一到自动去死

如果不设置过期时间，默认是多少？

答：浏览会话结束时——浏览器关闭的时候



Cookie技术可以将用户的信息保存在各自的浏览器中，

优点：很明显实现了同一个用户不同请求中数据共享。

缺点：黑客可以利用脚本等手段 窃取cookie中的重要数据，从而泄漏个人的隐私，存在巨大的安全隐患。

# 7、Session（服务器端的会话技术）

1、session，在服务器端保存用户的数据。（注意：session技术，还是依赖cookie技术）。

2、会话：从第一次请求服务器开始，一直到关闭浏览器，这一段操作，称之为：会话。

相当于平时打电话。

3、总结：

1 session是服务器开辟的一个用来存储数据的空间

2 服务器为每个浏览器单独开辟一个session

3 服务器根据浏览器发送过来的cookie，来确认当前浏览器使用哪个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（服务器端）。

* 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。

Session对象的创建时间：

当第一次调用request.getSession()的时候创建session容器.

如果第一次访问jsp页面,也会创建session容器

Session的销毁时间：

1. （自动去死）Session在服务器的存活时间。Session对象在服务器内部有默认的存活的时间。一般默认是30分钟。如果在30分钟内，用户没有再对当前这个服务器中的资源进行任何的访问操作，这时只要时间到达30分钟，服务器会自动的销毁这个session。

Session的存活时间我们可以在当前这个项目的web.xml中配置：

# 8、JavaWEB中三种数据范围（三种容器）

Request ServletContext Session

三个对象都有：setAttribute getAttribute方法，都可以存取数据

回答面试官问题套路：先回答基本概念，返回，跟上使用案例。

问题：什么时候使用request对象，保存数据？

答：一次请求中需要使用的数据，就保存在request。举例：商品数据，就存入请求中。

问题：什么时候使用ServletContext 对象，保存数据？

答：全局使用的数据，整个项目需要使用的数据，就是，要存入ServletContext 。举例：在线人数，存入ServletContext 。

问题：什么时候使用session对象，保存数据？

答：一次会话（多次请求——都发生在一次开启浏览器和关闭浏览器过程中）中需要使用的数据，都存入session。

举例：点餐商品

# JSP

1. 格式如下：<%= java代码 %>
2. 注释问题

由于JSP中可以书写html、Java代码，因此在JSP中可以嵌入html注释，还可以嵌入Java注释，还可以书写JSP自己的注释

1. html注释：<!—注释 -->
2. java注释：Java的注释必须嵌入在上面介绍的三个脚本中，不能在jsp其他位置书写。
3. jsp自己的注释：<%-- 注释--%>

<!--

jsp作用：简化书写html标签，动态的生成网页内容（jsp中可以书写java代码）

演示，jsp文件中的各种注释

-->

<%-- jsp文件中注释 ：不会翻译到 .java文件中--%>

<%-- html注释，无法注释掉java代码，但是，在html文件中依然起作用 --%>

<%-- java注释会原封不动的翻译到.java文件中 --%>

<!--11111 <% //int i = 0; %> -->

<%-- 总结：希望在哪个类型的文件中，看到注释，就使用哪种类型的注释 --%>

1. JSP中的page指令（重点：必须掌握）

例子：

<%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

第一个记住：格式为：<%@ 指令的名字 key=value key=value …….%>就是jsp指令

Key:属性名称

Value：属性值

Key和value值是用来设置jsp页面（例子：<a href=”http://www.baidu.com”></a>）

import="java.util.\*"

pageEncoding="UTF-8"

errorPage="500.jsp"

### JSP中的include指令

JSP中的include指令作用：引入其他的页面（头页面和尾），合并成一个页面，展示。这种引入方式称为静态引入。

<%@include file=*"footer.jsp"* %> 静态引入，翻译成一个.java文件

<jsp:include page=*"header.jsp"*/><br> 动态引入 翻译成不同的.JAVA，最后拼接在一块

### 4、JSP中的taglib指令（重点：必须掌握）

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

### 5、JSP的内置对象9个：

HttpServletRequest request 请求

HttpServletResponse response 响应

HttpSession session 会话

ServletContext application 表示当前项目对象

ServletConfig config 专门获取当前这个Servlet的配置信息

PageContext pageContext它表示的是当前jsp页面的上下文对象

作用：它的主要功能就是可以获取到JSP页面上的其他八个内置对象。

Throwable exception 主要是保存JSP页面上的异常信息的对象

JspWriter out 它相当于我们在Servlet中使用的response.getWriter

Page

总结：web中的四个域对象（容器对象）

Page(当前页面) （当前项目的根路径） < request（一次请求）（商品） < session（一次会话，多次请求）（昵称） < ServletContext(整个项目）在线人数

### 6、 jsp:forward标签

它主要是起到我们在Servlet中学习的request的请求转发的作用。

<jsp:forward page=*"index.jsp"*></jsp:forward>

# 注解

1. @WebServlet(name = "HelloServlet",urlPatterns = "/hello")
2. **Spring:**

(1)@Repository(name=”userDao”)

(2)@Service(name=”userService”)

@Resources(name=”userDao”) 或 @Autowired @Qualifier("userDao")

(3)@Value(value="#{customerDao}")