# JSP对象

九对象：request、response、session、out、pagecontext、page、exception、application、config。

**加上一个Cookie对象**

JVM只能识别java类，并不能识别JSP代码。Servlet就是用Java编写的服务器端程序。狭义的Servlet是指Java语言实现的一个接口，广义的Servlet是指任何实现了这个Servlet接口的类。JSP的本质就是Servlet，Servlet是JSP的前身，JSP是Servlet的扩展；当JSP第一次被访问时，Web容器将JSP代码编译成JVM能够识别的Java类，JSP经编译后就变成了Servlet。（JSP与Servlet联系）

JSP中的内置对象都是通过HttpServletRequest对象，HttpServletResponse对象以及HttpServlet对象得到。

JSP是HTML中内嵌Java代码，侧重页面显示；而Servlet是HTML代码与Java代码分开，侧重逻辑控制。（JSP与Servlet区别）

**1、request对象（转译后对应javax.servlet.http.HttpServletRequest对象）：**

request对象代表了客户端的请求信息，主要用于接受通过HTTP协议传送到服务器的数据（包括头信息、系统信息、请求方式以及请求参数等）。

request对象的生命周期：一次请求。

request对象的作用域：JSP页面内有效。

request常用的方法：

1 object getAttribute(String name) ----返回指定属性的属性值

2 Enumeration getAttributeNames() ----返回所有可用属性名的枚举

**3 String getCharacterEncoding() ----返回字符编码方式**

4 int getContentLength() ----返回请求体的长度（以字节数）

5 String getContentType() ----得到请求体的MIME类型（多用途互联网邮件扩展类型）

6 ServletInputStream getInputStream() ----得到请求体中一行的二进制流

**7 String getParameter(String name) ----返回name指定参数的参数值**

8 Enumeration getParameterNames() ----返回可用参数名的枚举

9 String[] getParameterValues(String name) ---返回包含参数name的所有值的数组

10 String getProtocol() ----返回请求用的协议类型及版本号

11 String getScheme() ----返回请求用的计划名,如:http.https及ftp等

12 String getServerName() ----返回接受请求的服务器主机名

13 int getServerPort() ----返回服务器接受此请求所用的端口号

14 BufferedReader getReader() ----返回解码过了的请求体

15 String getRemoteAddr() ----返回发送此请求的客户端IP地址

16 String getRemoteHost() ----返回发送此请求的客户端主机名

17 void setAttribute(String key,Object obj) ---设置属性的属性值

18 StringgetRealPath(String path) ----返回一虚拟路径的真实路径

19 String request.getContextPath() ----返回上下文路径

**2、response对象 （转译后对应javax.servlet.http.HttpServletResponse对象）：**

response对象表示服务器端对客户端的回应。主要用于设置头信息、跳转、Cookie等。

response的生命周期：一次响应。

response的作用域：只在JSP页面内有效。

response常用方法：

**public void setHeader(String name, String value) ----设置头信息的名字和内容**

**public void sendRedirect(String location) ----跳转，由一个页面跳往另外一个页面**

Public Void addCookie (Cookie cookie) ----向客户端加入cookie

public void setContentType(java.lang.String type) ----设置内容的返回类型

**（request）请求转移与请求重定向（response）的区别：**

请求重定向：response.sendRedirect(), 客户端行为，从本质上讲等同于两次请求，前一次的请求对象不会保持，地址栏的URL地址会改变。

请求转发：request.getRequsetDispatcher().forward(requset,response);服务器行为，是一次请求，转发后请求对象会保存，地址栏的URL地址不会改变。（服务器内部转发，所有客户端看不到地址栏的改变）。

1. **session对象 （转译后对应javax.servlet.http.HttpSession对象）**

session对象用于存储特定的用户会话所需的信息 。

session对象生命周期：从存入数据时开始，默认闲置30分钟后失效。

session对象作用域：会话内有效。（从一个客户打开浏览器并连接到服务器开始，到客户关闭浏览器离开这个服务器结束，被称为一个会话；因为http是无状态的，Session需要使用Cookie作为识别标志。该Cookie为服务器自动生成的，它的maxAge属性一般为-1，表示仅当前浏览器内有效，并且各浏览器窗口间不共享，关闭浏览器就会失效。）

session常用方法：

public String getId() ----获取Session对象编号。

**public void setAttribute(String key,Object obj) ----将参数Object指定的对象obj添加Session对象中，并为添加的对象指定一个索引关键字。**

**public Object getAttribute(String key) ----获取Session对象中含有关键字的对象。**

public Boolean isNew() ----判断是否是一个新session

1. **application对象（转译后对应javax.servlet.ServletContext对象）：**

application对象用于存储和访问来自任何页面的变量，类似于 session 对象。不同之处在于，所有的用户分享一个 Application 对象，类似于“全局变量”，而 session 对象和用户的关系是一一对应的。

application对象生命周期：服务器启动发送第一个请求时就产生了Application对象，直到服务器关闭。

application对象常用方法：

**setAttribute(String key,Object obj) ----将参数Object指定的对象obj添加到Application对 象中，并为添加的对象指定一个索引关键字。**

**getAttribute(String key) ----获取Application对象中含有字的对象。**

1. **out 对象（转译后对应javax.servlet.jsp.jspWriter对象）：**

out 对象用于在Web浏览器内输出信息，并且管理应用服务器上的输出缓冲区。（注意要及时关闭输出流）

其主要方法如下：

clear() ----清除缓冲区中的数据，若缓冲区已经是空的，则会产生IOException异常；

clearBuffer() ----清除缓冲区的数据，若缓冲区为空，不会产生IO异常；

flush() -----直接将目前暂存于缓冲区的数据输出；

getBufferSize() ----返回缓冲区的大小；

getRemaining() ----返回缓冲区的剩余空间大小；

isAutoFlush() ----返回布尔值表示是否自动输出缓冲区的数据；

输出数据的一些方法：

newLine() ----输出换行；

print(datatype data) ----输出不同数据类型的数据；

println(datatype data) ----输出不同数据类型的数据，并自动换行；

1. **pageContext 对象（转译后对应javax.servlet.jsp.PageContext对象）：**

pageContext对象能够存取其他隐含对象，如request、reponse、session、application 等对象。**（实际上，pageContext对象提供了对JSP页面所有的对象及命名空间的访问。）**

pageContex常用方法：

void setAttribute(String name, Object value, int scope)

----pageContext对象存取其他隐含对象属性的方法，此时需要指定范围的参数：范围参数有四个，分别代表四种范围：PAGE\_SCOPE、REQUEST\_SCOPE、SESSION\_SCOPE、APPLICATION\_SCOPE

pageContext对象取得其他隐含对象的方法：

Exception getException( ) ----回传网页的异常

JspWriter getOut( ) ----回传网页的输出流

Object getPage( ) ----回传网页的Servlet 实体

ServletRequest getRequest( ) ----回传网页的请求

ServletResponse getResponse( ) ----回传网页的响应

ServletConfig getServletConfig( ) ----回传此网页的ServletConfig 对象

ServletContext getServletContext( ) ----回传此网页的执行环境

HttpSession getSession( ) ----回传和网页有联系的会话

1. **PageContext对象提供取得属性的方法**

Object getAttribute(String name, int scope)

----回传name 属性，范围为scope的属性对象，回传类型为Object

Enumeration getAttributeNamesInScope(int scope)

----回传所有属性范围为scope 的属性名称，回传类型为Enumeration

int getAttributesScope(String name) ----回传属性名称为name 的属性范围

void removeAttribute(String name) ----移除属性名称为name 的属性对象

void removeAttribute(String name, int scope)

----移除属性名称为name，范围为scope 的属性对象

void setAttribute(String name, Object value, int scope)

----指定属性对象的名称为name、值为value、范围为scope

Object findAttribute(String name)

----寻找在所有范围中属性名称为name 的属性对象

1. **config 对象（转译后对应javax.servlet.ServletConfig对象）：**

config对象的主要作用是取得服务器的配置信息。

config对象常用方法：

getServletContext() ----返回 一个含有服务器相关信息的ServletContext对象。

getIntParameter(String name) ----返回初始化参数的值。

getIntParameterNames()----返回包含了Servlet初始化所需要的所有参数返回类型是枚举型

1. **page对象 （转译后对应this）：**

page对象代表JSP本身，只有在JSP页面内才是合法的。page对象有点类似于Java编程中的this指针，就是指当前JSP页面本身。

1. **exception对象（转译后对应java.lang.Throwable对象）：**

exception对象的作用是显示异常信息，必须在page 指令中设定< %@ page isErrorPage="true" %>才能使用，在一般的JSP页面中使用该对象将无法编译JSP文件。

exception对象常用方法：

getMessage( ) ----该方法返回错误信息。

**printStackTrace( ) ---- 该方法以标准错误的形式输出一个错误和错误的堆栈。**

**toString() ----该方法以字符串的形式返回一个对异常的描述。**

1. **Cookie对象：会话层跟踪机制【非内置对象】**

用来记录用户登录的用户名、密码 、登陆时间、个性化服务等

1举例： Cookie cookie=new Cookie（“username”,”Jack”）;

respone.addCookie(cookie);

//注意：Cookie值不能包含空格、方括号、圆括号、等号、逗号、双引号、斜杠、问号、@、冒号、分号。

1. JSP通过调用request.getCookies()从客户端读入Cookie对象数组。再循环访问各个 Cookie元素对象，然后调用getName()方法逐个检查Cookie的名字，直到找到目标 Cookie，然后对该Cookie对象调用getValue（）方法与指定名字有关的值
2. Save—Read：

保存时，使用Java.net.URLEncoder.encode(String s,String enc)对中文进行编码；

读取时，使用java.net.URLDecoder.decode(String s,String enc)进行解码。

**存时候**

Cookie cookie=**new** Cooki**e(URLEncoder.encode("姓名","utf-8"),URLEncoder.encode("杰克","utf-**8"));

cookie=**new** Cookie("userIP" ,userIP);//不能这么写

cookie.setMaxAge(60\*60);设定该cookie再用户机器上硬盘上的存活期为一小时

response.addCookie(cookie);//添加进Cookie

**取时候：**

String name=URL**Decoder.decode(cookie.getName(),"ut**f-8");

String value=URLDecoder.decode(cookie.getValue(),"utf-8");

【若前期使用cookie=new Cookie("userIP" ,userIP);读取时要String

value=URLDecoder.decode(cookie.getValue().replaceAll("%","%25"),"utf-8");】

1.错误：Exception in thread "main" java.lang.IllegalArgumentException: URLDecoder: Incomplete trailing escape (%) pattern  
原因是字符串中包含有%字符

解决方法如下 example：

String str = "hello该字符串中包含%";

System.out.println(URLDecoder.decode(str.replaceAll("%", "%25"), "UTF-8"));

1. java.lang.IllegalArgumentException: Cookie name "姓名" is a reserved token

//把中文换成英文、 在servlet中创建Cookie必须遵守RFC 2109规范。规范对于 cookie name的原文描述如下：The name must conform to RFC 2109. That means it can contain only ASCII alphanumeric characters and cannot contain commas, semicolons, or white space or begin with a $ character. 大致中文含义就是name 只能是ascii字符并且不能包含逗号，分号，空格，同时不能以$开头。这是cookie 的标准，和你servlet的request/response用什么编码没有任何关系