第5讲 JSP技术

1. Java WEB开发环境

- JDK：Java Development Kit
- 集成开发环境IDE：Eclipse
- HTTP服务器：Tomcat
- 数据库服务器：MySQL

2. JSP简介

- JSP（Java Server Pages）
- 在HTML代码中嵌入Java代码片段和JSP标签，形成JSP网页(*.jsp)
- 在接收到用户请求时，服务器会处理Java代码片段和JSP标签，然后生成处理结果的HTML页面返回给客户端，客户端的浏览器将呈现最终页面效果

JSP文件内容的构成

- JSP原始代码中包含了模版元素和JSP元素构成
- 模版元素指的是JSP引擎不处理的部分, HTML + Javascript + CSS等 。
- JSP元素则指的是由JSP引擎直接处理的部分，这一部分必须符合JSP语法，否则会导致编译错误。
  - 代码元素：声明、代码段、表达式
  - 注释元素：HTML注释；JSP隐藏注释；java注释
  - 指令元素：page、include、taglib、tag等
  - 动作元素：jsp:include、jsp:forward等

3. JSP语法

- 脚本代码：可以编写单行或多行的Java代码，其数据类型、语法格式、控制结构与Java代码格式完全相同
- HTML代码中包含脚本代码的语法格式：

  <% 脚本程序代码片段 %>

- 例5-1

JSP声明

- 一个声明语句可以声明一个或多个全局变量或全局方法，供后面的Java代码使用
- JSP声明的语法格式：
  <%!变量或方法 %>
- 全局变量是属于类变量，首次调用时进行初始化，在该网站生存期内始终存在。
- 全局方法属于类方法
- 例5-2

JSP表达式

- JSP表达式可以直接把Java的表达式结果输出到JSP页面中。
- 表达式的最终运算结果将被转换为字符串类型，并插入到表达式出现的地方。
- JSP表达式的语法格式如下：

  <%= 表达式 %>

JSP注释

- JSP注释：注释内容不会被服务器编译运行，也不会被发送到浏览器

  <%-- 注释文本 --%>
- HTML注释：注释内容会直接发送到浏览器，用户可以通过查看网页源代码看到注释内容

  <!-- 注释文本 -->
- 单行注释：//注释内容
- 多行注释：/*注释文本*/
- JavaDoc注释：/**JavaDoc注释文本*/

4. 指令元素

- 指令元素不会产生任何内容输出到网页中，主要用于定义整个JSP页面的相关信息。例如：使用的语言，导入的类包，指定错误处理页面等。
- 其语法格式如下：

<%@指令名称 属性=“属性值” ... %>

➢三种指令标签

- <%@page...%>：定义页面的依赖属性
- <%@include...%>：包含其他文件
- <%@taglib...%>：引入标签库的定义

(1) page指令

- language属性：用于定义页面所用的脚本语言，默认是Java
- contentType属性：用于定义当前页面的字符文本的类型和字符编码，默认值是“text/html; charset=ISO-8859-1”
- pageEncoding属性：用于定义当前页面的编码方式，默认为ISO-8859-1

- 在JSP标准的语法中，如果pageEncoding属性存在，那么JSP页面的字符编码方式就由pageEncoding决定，否则就由contentType属性中的charset决定，如果charset也不存在，JSP页面的字符编码方式就采用默认的ISO-8859-1。
- 如果页面中含有中文，一般将如下指令放到jsp页面的最上方

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

- import属性：导入要使用的Java包和类
  - 在Java中，如要载入多个包，需用import分别指明，在JSP里，可用import指明多个包，之间用逗号隔开。

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

    或者分多行写

    <%@ page import="java.text.*"%>

    <%@ page import=" java.util.*"%>
  - JSP默认引入的包：

    java.lang.*

    javax.servlet.*

    javax.servlet.jsp.*

    javax.servlet.http.*

- extends属性：用于设置JSP页面继承的Java类
  - 所有JSP页面在执行之前都会被服务器解析成Servlet，而Servlet是由Java类定义的，所以JSP和Servlet都可以继承指定的父类。
  - 该属性并不常用，而且有可能影响服务器的性能优化。

- errorPage属性：指定当JSP页面发生异常时需要转向的错误处理页面
- isErrorPage属性：指定当前页面是否可以作为另一个JSP页面的错误处理页面

(2) include指令

- include指令用于文件包含。
- 该指令可以在JSP页面中包含另一个文件的内容，被包含文件中的所有内容都被原样包含到该JSP页面中
- 如果被包含文件中有代码，这部分代码将被包含到JSP页面之后，再一起被执行。
- 被包含的文件可以是一段Java代码、HTML代码或者是另一个JSP页面。
- 例5-3

(3) taglib指令

- 该指令用于加载用户自定义标签
- 使用该指令加载后的标签可以直接在JSP页面中使用。
- 其语法格式如下：

```
<%@ taglib uri="uri" prefix="prefixOfTag" %>
```

5. 动作元素

- JSP动作元素在请求处理阶段起作用
- 动作元素基本上都是预定义的函数，JSP规范定义了一系列的标准动作，它用JSP作为前缀。
- 语法：

```
<jsp:action_name attribute="value" />
```

(1) jsp:include动作

- 作用：在当前的JSP页面中加入（包含）静态和动态的资源。
  - 静态网页：直接将内容加入到JSP网页中。
  - 动态网页：编译运行该文件加入到JSP网页中。

- 语法格式：

```
<jsp:include page=”相对URL地址” flush=”true”/>
```

或者

```
<jsp:include page=”相对URL地址”  flush=”true”>
    <jsp:param name=”参数名” value=”参数值”/>
    ……
</jsp:include>
```

比较include动作和指令两种包含方式

- 执行时间上
  - 指令<%@ include file=“”%> ： 编译时包含文件
  - 动作<jsp:include page=“”/> ： 运行时包含文件
- 引入内容的不同
  - <%@ include file=”relativeURI”%>：引入静态文本(html, jsp),在JSP页面被转化成servlet之前和它融和到一起
  - <jsp:include page=”relativeURI” flush=”true” />：引入执行页面或servlet所生成的结果

(2) jsp:forward动作

作用：把当前的JSP页面重新定向到另一个页面上

语法格式：

```
<jsp:forward page=”重定向的页面”>
    <jsp:param name=”name” value=”value”/>
    …
</jsp:forward>
```

注意：<jsp:forward>标记后面的代码不会被执行。

用户看到的地址是当前页面的地址，而内容则是另一个页面的。

(3) jsp:param动作

- jsp:param：用来向目的页面提供key/value的信息，可以与<jsp:include>，<jsp:forward>，<jsp:plugin>一起搭配使用
- 例5-4

5. JSP隐式对象

- JSP隐式对象是JSP容器为每个页面提供的Java对象，开发者可以直接使用它们而不用显式声明。
- JSP隐式对象也被称为预定义变量。

JSP所支持的九大隐式对象

- request：HttpServletRequest 接口的实例
- response：HttpServletResponse 接口的实例
- out：JspWriter类的实例，用于把结果输出至网页上
- session：HttpSession类的实例
- application：ServletContext类的实例，与应用上下文有关
- config：ServletConfig类的实例
- pageContext：PageContext类的实例，提供对JSP页面所有对象以及命名空间的访问
- page：类似于Java类中的this关键字
- Exception：Exception类的对象，代表发生错误的JSP页面中对应的异常对象

6. http协议

- HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW: World Wide Web ）服务器传输超文本到本地浏览器的传送协议。。
- HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

http工作原理

- HTTP协议工作于客户端-服务端架构上。
- 浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。
- Web服务器根据接收到的请求后，向客户端发送响应信息。
- HTTP默认端口号为80。

http协议三个特点

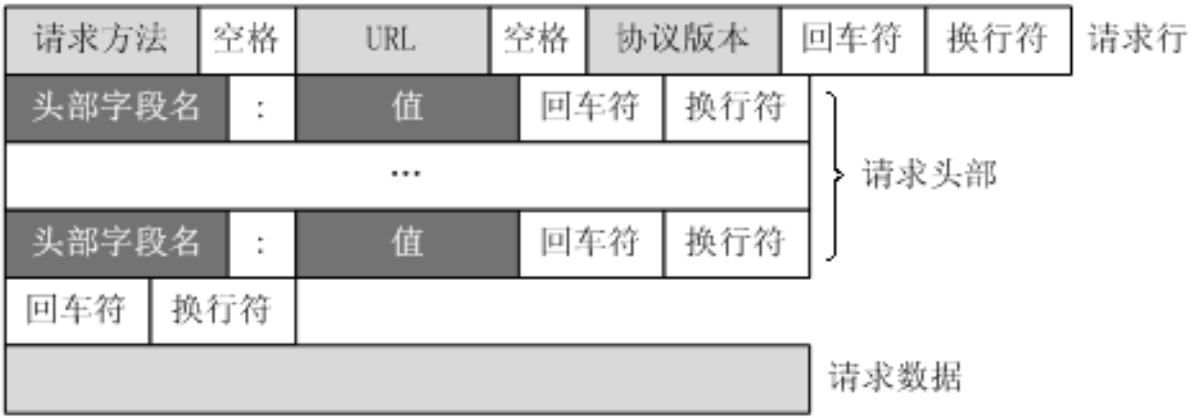
- 无连接：每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。
- 媒体无关：任何类型的数据（如文本、文档、图片、视频、音频等）都可以通过HTTP发送
- 无状态：无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传。

http消息结构

- 一个HTTP客户端是一个应用程序（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP请求的目的。
- 一个HTTP服务器同样也是一个应用程序（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。

客户端http请求消息

客户端发送一个HTTP请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据四个部分组成



http请求方法

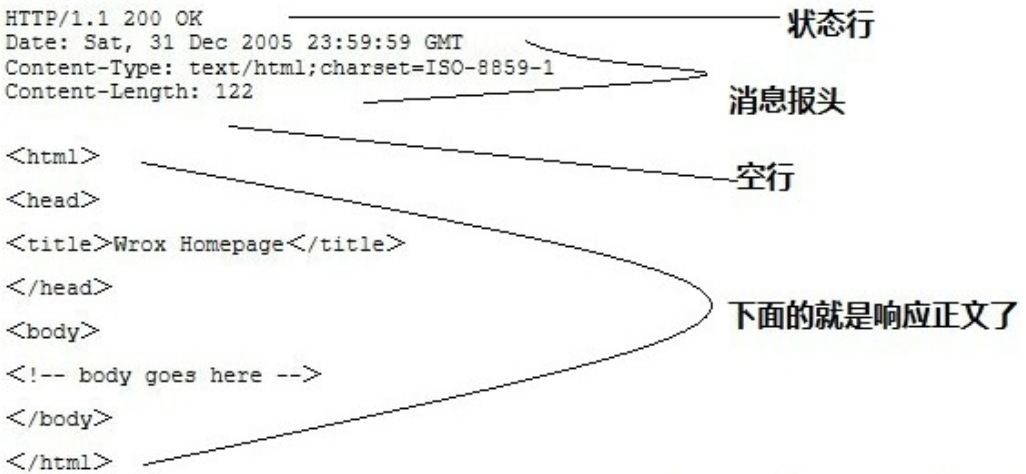
- GET：指定页面信息，并返回实体主体
- POST：向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求数据中。
- PUT：从客户端向服务器传送的数据取代指定的文档的内容。
- DELETE：请求服务器删除指定的页面。
- 其他的请求方法还有：HEAD，CONNECT，OPTIONS，TRACE

http请求头部（部分）

- Accept：可接受的响应内容类型
- Accept-Charset：可接受的字符集
- Accept-Encoding：可接受的响应内容的编码方式。
- Cookie：由之前服务器通过Set-Cookie设置的一个HTTP协议Cookie
- Content-Type：请求体的MIME类型
- Host：表示服务器的域名以及服务器所监听的端口号。如果所请求的端口是对应的服务的标准端口（80），则端口号可以省略。
- User-Agent：浏览器的身份标识字符串

服务器响应消息

- HTTP响应也由四个部分组成，分别是：状态行、消息报头、空行和响应正文。



http响应常用的状态码

- 200 – OK：请求成功
- 301 – Moved：永久移动。资源（网页等）被永久转移到其它URL
- 302 – Found：临时移动。资源（网页等）被临时转移到其它URL
- 404 – Not Found：请求的资源（网页等）不存在
- 500 – Internal Server Error：内部服务器错误

http响应头部信息（部分）

- Content-Encoding：文档的编码（Encode）方法。
- Content-Type：表示后面的文档属于什么MIME类型。Servlet默认为text/plain，但通常需要显式地指定为text/html。
- Expires：应该在什么时候认为文档已经过期
- Last-Modified：文档的最后改动时间。
- Location：表示客户应当从哪里去提取文档，与302状态码配合使用
- Refresh：表示浏览器应该在多少时间之后刷新文档，以秒计
- Set-Cookie：设置和页面关联的Cookie。

7. request对象

- 是一个javax.servlet.http.HttpServletRequest对象。
- 每当客户端请求一个JSP页面时，JSP引擎就会制造一个新的request对象来封装浏览器的请求信息
- 提供了获取cookie、header和session等对象的数据的方法。
- 请求信息的内容包括：请求的头信息（Header）、系统信息（比如：编码方式）、请求的方式（比如：GET或POST）、请求的参数名称和参数值等信息

(1) 获取单个参数值

- 以字符串形式返回给定参数的值，如果参数不存在则返回null。
- 语法：

  String getParameter(String name)
- 例5-5

(2) 获取某个参数的所有值

- 返回指定名称的参数的所有值，若不存在则返回null
- 语法：
  String[] getParameterValues(String name)
- 例5-6

(3) 获取请求客户端的其他信息

方法	返回值	说明
getHeader(String name)	String	返回指定名称的HTTP头信息
getMethod()	String	获取客户端向服务器发送请求的方法
getContextPath()	String	返回请求路径
getProtocol()	String	返回请求使用的协议
getRemoteAddr()	String	返回客户端IP地址
getRemoteHost()	String	返回客户端主机名称
getRemotePort()	int	返回客户端发出请求的端口号
getServletPath()	String	返回接受客户提交信息的页面
getRequestURI()	String	返回部分客户端请求的地址，不包括请求的参数
getRequestURL()	StringBuffer	返回客户端请求地址

例5-7

8. response对象

- response对象作用：JSP会根据客户端的请求建立一个默认的response对象，用来封装JSP处理数据后产生的结果，并将其传回到客户端相应客户端的请求，其对应的接口为javax.servlet.http.HttpServletResponse接口。
- response对象用来提供给客户端浏览器的参考信息，比如响应的标头（head），响应的本体(HTML文本内容)以及服务端的状态码信息。
- 提供了几个用于设置送回浏览器的相应方法

(1) sendRedirect方法

- void sendRedirect(String location)
- 本方法将响应发送到另一个指定的位置进行处理，
- 该方法会同时设置状态码为302
- 例5-8

(2) setHeader方法

- void setHeader(String name, String value)
- 使用指定名称和值设置响应头的名称和内容
- 例5-9

(3) 其他常用方法

- String encodeURL(String url)：将URL编码，回传包含Session ID的URL
- void addCookie(Cookie cookie)：添加指定的cookie至响应中
- void setCharacterEncoding(String charset)：指定响应的编码集（MIME字符集）
- void setContentLength(int len)：指定HTTP servlets中响应的内容的长度，此方法用来设置 HTTP Content-Length 信息头
- void setContentType(String type)：设置响应的内容的类型，如果响应还未被提交的话

9. Cookie管理

- Cookie是Web服务器保存在用户硬盘上的一段文本。
- Windows系统中保存路径C:\Documents and Settings\登录用户名\Cookies
- Cookie中，信息的片断以“名/值”对(name-value pairs)的形式储存。
- 作用：
  - Cookie对象通常用于在浏览器端保存与服务器会话过程中的一些数据。
  - 当浏览器访问Web服务器（某一站点）时，相应的cookie会自动发送到服务器上。

服务器识别客户端Cookie的三个步骤

- 服务器脚本发送一系列cookie至客户端浏览器。
- 浏览器在本地机中立即存储这些信息
- 当下一次浏览器发送任何请求至服务器时，它会同时将发送这些cookie信息到服务器，然后服务器使用这些信息来识别用户或者进行其他的操作。

(1)使用response向客户端写入Cookie

- 对于cookie的写入，要结合response对象的addCookie()方法来实现
- 写入Cookie的主要步骤：
  - 创建Cookie对象
  - 设定Cookie的属性（一般设置Cookie的有效期）
  - 调用response.addCookie(Cookie c)方法将其写入到客户端

```
<%
   Cookie c=new Cookie("season","spring");
   c.setMaxAge(30);    //cookie的有效期为30秒
   response.addCookie(c);
%>
```

Cookie的有效期

- 过期属性是按秒为单位记录的，使用正整数
- 负值表示该cookie的生存期是当前浏览器会话
- 零值表示立即删除该cookie
- 如果不设置cookie的有效期，就不能在硬盘上保存cookie的信息，仅存在浏览器内存中。一旦浏览器关闭，cookie信息就消失。
- Cookie有效期的设置必须在response.addCookie()方法之前。

(2)使用request读取客户端Cookie

- request对象的getCookies()方法可以读取所有客户端发送过来的Cookie对象
- cookie对象的getName()方法可以获取该对象的名称
- cookie对象的getValue()方法可以获取该对象的值
- 例5-10

10. Session对象

- Session对象存储在服务器端，作用是在会话范围内，记录每个客户端的访问状态，以便于跟踪每个客户端的操作状态
- 它所实现的接口为：javax.servlet.http.HttpSession
- 可以在jsp页面中直接使用session对象，也可以通过pageContext.getSession()或 request.getSession ()方法重新获取。

session的工作原理

①客户首次访问服务器的一个页面时，服务器就会为该客户分配一个session对象，同时为该session对象指定一个唯一的ID，并将该ID号发送到客户端并要求客户端写入到Cookie中，使得客户端与服务器对应该客户端的某一个session对象建立一对一的关系。

session的工作原理

②当客户继续访问服务上的其他资源时会在HTTP头中带上session ID，服务器不再为该客户分配新的session ID，直到客户端浏览器关闭、超时或调用session的invalidate()方法时期失效，客户端与服务器的会话结束。

③当客户端重新打开浏览器访问网站时，服务器会重新为客户分配一个Session对象，重新分配Session id。

session的常用方法

- public void setAttribute(String name, Object value)：使用指定的name和value来产生一个对象并绑定到session中
- public Object getAttribute(String name)：返回session对象中与指定name绑定的对象，如果不存在则返回null。
- public void removeAttribute(String name)：删除指定name的session属性
- public void invalidate()：使session无效
- public String getId()：获取当前的会话ID
- 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。
- 例5-11, 5-12

cookie与session的区别和联系

- 不同：
  - 存放地点：cookie存放在客户端的硬盘里，属于离线存放，而session存放在服务器的内存中。
  - 存活时间：cookie可以长期存放在客户端，具体的存活时间由setMaxAge()方法所指定的数值决定，session随用户访问服务器而产生，随客户超时或下线而消失。
  - 安全性：cookie存放在客户端，可能会被别有用心的网站读取，安全性较差，而session存放在服务器的内存中，用户不能修改，且随客户端浏览器的关闭而消失，安全性较好。
- 联系：不论是cookie还是session内置对象都需要浏览器支持cookie并且没有禁用cookie。

11. application对象

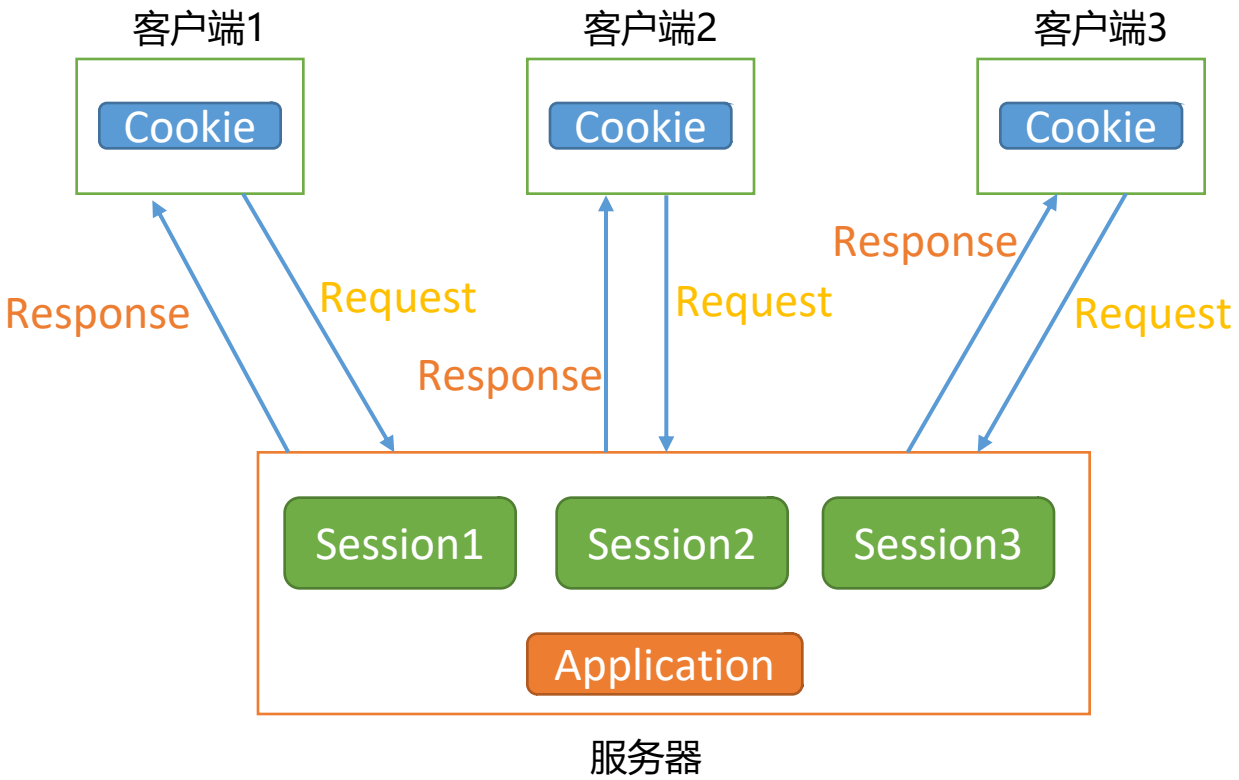
- application对象可将信息保存在服务器中，直到服务器关闭，application对象中保存的信息会在整个应用生存期内都有效。
- 与session对象相比，application对象的生命周期更长，类似于系统的“全局变量”
- 不同于session对象每个客户端均有自己的一份，application对象在网站中仅有一份

application对象的常用方法

方法	返回值	说明
getAttribute(String name)	Object	通过关键字返回保存在application对象中的信息
getAttributeNames()	Enumeration	获取所有application对象使用的属性名
setAttribute(String key,Object obj)	void	通过指定的名称将一个对象保存在application对象中
getMajorVersion()	int	获取服务器支持的Servlet版本号
getServerInfo()	String	返回JSP引擎的相关信息
removeAttribute(String name)	void	删除application对象中指定名称的属性
getRealPath()	String	返回虚拟路径的真实路径
getInitParameter(String name)	String	获取指定name的application对象属性的初始值

例5-13, 5-14

五大对象关系图


客户端1
客户端2
客户端3
Cookie
Cookie
Cookie
Response
Request
Response
Request
Response
Request
Session1
Session2
Session3
Application
服务器

12. out对象

- out对应的接口为javax.servlet.jsp.JspWriter。
- 作用:主要是向浏览器输出数据，也可以通过out对象对缓冲区进行操作。
- 除了直接使用内置对象以外，我们还可以使用pageContext.getOut()方法获取out对象。
- 默认情况下，服务器利用out对象将网页内容输出时，会输出的内容放在一个默认大小为8KB的缓冲区中，等到缓冲区满时再一次将内容送往客户端。

out对象的常用方法

- void print(DataType dt)：输出任意类型的数据到HTML文本流中，等同于<%=...%>
- out.println(DataType dt)　输出任意类型的数据到HTML文本流中换行