**第10课 Java 的网络应用**

|  |  |  |
| --- | --- | --- |
| **课 题** | Java 的网络应用 | |
| **课 时** | 6课时（270 min）。 | |
| **教学目标** | **知识技能目标：**  1．了解常用 JSP 内置对象的特点和使用方法。  2．掌握访问 Bean 属性的两种方法。  **思政育人目标：**  让学生通过学习现代文读写，了解现代文的分类，培养学生对现代文读和写的兴趣和能力。 | |
| **教学重难点** | **教学重点：**记叙文阅读与写作、说明文阅读与写作  **教学难点：**议论文阅读与写作 | |
| **教学方法** | 讲授法、问答法、讨论法 | |
| **教学用具** | 电脑、投影仪、多媒体课件、教材 | |
| **教学设计** | 第1节课：考勤（2min）--知识讲解（40min）--作业布置（3min）  第2节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第3节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第4节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第5节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min）  第6节课：知识讲解（40min）--课堂小结（3min）--作业布置（2min） | |
| **教学过程** | **主 要 教 学 内 容 及 步 骤** | **设计意图** |
| **考勤**  **（2min）** | ■【教师】清点上课人数，记录好考勤  ■【学生】班干部报请假人员及原因 | 培养学生的组织纪律性,掌握学生的出勤情况 |
| **知识讲解**  （40min） | **【教师】**展示一个简单的 JSP 页面（一）  **任务描述**  编写一个 JSP 页面，显示用户的 IP 地址和其他相关信息。  **任务目标**  1. 会 JSP 的基本语法规则。  2. 能够掌握常用 JSP 内置对象的特点和使用方法。  **预备知识**  JSP 页面和 HTML 页面的主要区别是：HTML 页面是静态的，在客户端看到的HTML 页面中的代码和在服务器上存储的 HTML 文件代码是一样的，而在 JSP 页面中，客户端所能看到的代码是由 JSP 服务器在 JSP 页面运行时动态生成的，和服务器端存储的 JSP 文件代码有很大区别。我们可以通过下面的例子来了解 JSP 页面的组成和运行情况。  **一、JSP 简介**  JSP（Java Server Pages）是由 SUN 公司倡导，多家公司参与，于 1999 年推出的一种动态网页技术标准。它是基于 Java Servlet 的 Web 开发技术，利用这一技术可以开发动态的、高性能的 Web 应用程序。在 HTML 文件中加入 Java 程序片段和 JSP 标记，就构成了 JSP 网页。JSP 技术继承了 Java 语言的特点，可以非常容易地跨平台、跨 Web 服务器软件来设计和开发，并保持了 Java 语言的健壮性和安全性。另外，JSP 既可以支持 HTML/DHTML 的传统浏览器文件格式，又可以支持应用于无线通信设备进行网页预览的WML 文件格式，还可以支持其他一些 B2B 电子商务网站应用的 XML 格式。  但是，JSP 的高效运行是通过 .class 文件常驻内存来实现的，这需要占用大量的内存和硬盘空间，对服务器的要求较高。另外，JSP 页面执行时是首先转换为 .java 文件，然后将 .java 文件编译为字节码文件，在 JSP 页面出错时，出错信息实际上指向的是转换后的那个 .java 文件，而不是 JSP 本身，这样会导致 JSP 程序调试困难。在技术方面，JSP 和 ASP 具有相同的程序结构，但二者在采用的编程语言、Web 服务器类型、操作系统平台等方面均有差别，JSP 和 ASP 的比较如表 10−1 所示。    **二、JSP 基本语法**  **（一）Java 程序片**  JSP 页面中使用 Java 程序片实现逻辑计算。Java 程序片有三种形式：实体定义、表达式和 Java 代码块。使用实体前，首先要定义实体。下面是 Java 程序片的定义和使用的语法格式。  1. 实体定义  实体定义是在 <%!...%> 标记中定义的变量、方法和类。这些变量、方法和类可以在包含该实体定义的 JSP 页面中使用。  2. 表达式  可以在 <%= ...%> 标记中插入一个表达式，这个表达式必须能计算出数据值。表达式的值由服务器负责计算，并将计算结果以字符形式发送到客户端。  3. Java 代码块  可以在 <%...%> 标记中包含多个 Java 语句，构成 Java 代码块。一个 JSP 页面可以有许多 Java 代码块，JSP 引擎按顺序执行这些 Java 代码块。Java 代码块中定义的变量相当于局部变量，在本 JSP 页面的所有 Java 代码块中起作用。  下面的 JSP 页面用于统计访问该页面的客户人数。由于多个客户点击同一页面文件，只要在 JSP 页面中定义一个全局变量，作为统计客户的计数器，每当页面被点击执行时，使该变量加 1，然后输出该计数器的值。  例 10.1  count.jsp：  <%@page contentType = "text/html; charset = GB2312"%>  <html>  <head>  <title> 统计客户人数 </title>  </head>  <body>  <%!  int count = 0;  %>  <% count++;  %>  <p> 您是第 <% = count%> 个访问本页面的客户。  </body>  </html>  **（二）JSP 标签**  JSP 标签用于控制页面属性，包括注释、指令标签和动作标签三类。下面是具体的  语法格式。  1. 注释  注释本身不具有语句功能，只用来增强 JSP 文件的可读性，便于维护 JSP 文件。常见注释分 HTML 注释和 JSP 注释两种。  HTML 注释不会被忽略，客户端通过浏览器查看 JSP 源文件时能够看到 HTML 注释。HTML 注释的格式是：  <!—注释内容 -->  JSP 注释在编译时会被 JSP 引擎忽略。JSP 注释的格式是：  <%-- 注释内容 --%>  2. page 指令  page 指令主要用来定义整个 JSP 页面的各种属性。一个 JSP 页面可以包含多个 page指令，指令中除 import 属性外，每个属性只能定义一次。page 指令的格式是：  <%@ page  [language = "java"]  [extends = "classname"]  [import = "packname/classname"]  [session = "true/false"]  [buffer = "none/sizekb"]  [autoFlush = "true/false"]  [isTheadeSafe = "true/false"]  [info = "info\_text"]  [errorPage = "error\_url"]  [isErrorPage = "true/false"]  [contentType = "MIME\_type"]  [pageEncoding = " "]  [isELIgnored = "true/false"]  %>  page 指令由多个属性组成，下面是对各个属性的解释：  language 属性：定义 JSP 页面中所使用的脚本语言。  extends 属性：定义 JSP 页面产生的 Servlet 所继承的父类。  import 属性：定义脚本环境中要使用的类。  session 属性：指定 JSP 页面是否参与一个 HTTP 会话。  buffer 属性：定义输出到客户端的流缓冲模式。  autoFlush 属性：指定客户端缓冲区是否自动刷新。  isTheadeSafe 属性：指定 JSP 页面是否可以供多个线程使用。  info 属性：定义 JSP 页面中待说明的信息。  errorPage 属性：定义当发生异常错误时调用的 JSP 页面。  isErrorPage 属性：指定当前 JSP 页面是否是另外一个 JSP 页面错误显示的目标。  contentType 属性：定义 JSP 页面的字符编码和页面响应的 MIME 类型。  pageEncoding 属性：定义 JSP 页面的字符编码。  isELIgnored 属性：指定是否忽略 EL（Expression Language，表达式语言，已纳入JSP2.0 标准）。  3. include 指令  include 指令的作用是在该标签位置处静态插入一个文件。所谓静态插入指用被插入的文件内容代替 include 指令标签，与当前 JSP 文件合并成新的 JSP 页面后，再由 JSP引擎编译为 Java 文件。  include 指令的格式是：  <%@ include file = "filename" %>  被插入的文件必须与当前 JSP 页面在同一 Web 服务目录下，合并后的 JSP 页面必须符合 JSP 语法规则。  4. taglib 指令  taglib 指令用来扩展标记指令，其格式如下：  <%@ taglib uri = "taglibURI" prefix = "tagPrefix" %>  其中 uri 是一个标记库描述器，用来唯一命名一组定制的标记，并且告诉容器如何处理特殊的标记。prefix 定义了 JSP 页面里要引用 taglib 标签时的前缀，这些前缀不能是 JSP、jspx、java、javax、sun、servlet 和 sunw。  5. include 动作  include 动作标签的格式如下：  <jsp:include page = "filename" />  或：  <jsp:include page = "filename" >  </jsp:include>  include 标签的作用是在当前 JSP 页面中动态包含一个文件，即将当前 JSP 页面和被包含的文件各自独立编译为字节码文件。当前 JSP 页面执行到 include 标签处时，才加载被包含文件的字节码。  6. param 动作  param 动作标签的格式如下：  <jsp:param name = "paramname" value = "paramvalue" />  param 标签用于定义变量名称和变量值，常与 include 动作、forward 动作和 plugin  动作配合使用，将定义的变量值传递给动态加载的文件。  7. forward 动作  forward 动作标签的格式如下：  <jsp:forward page = "filename" />  或：  <jsp:forward page = "filename" >  </jsp:forward>  forward 标签的作用是使当前页面执行到 forward 指令处转向其他 JSP 页面执行。  8. plugin 动作  <jsp:plugin type = "applet" code = "classname" codebase = "classdir"  width = "Width" ...> <jsp:fallback> 提示信息 ...  </jsp:plugin>  plugin 标签指示 JSP 页面加载 Java Plugin 插件，Java Plugin 插件由客户负责下载，并使用该插件运行 Applet 小应用程序。  9. useBean 动作  useBean 标签用于创建并使用一个 JavaBean，将在下个任务中介绍。  **【学生】**思考、讨论。 | **展示一个简单的 JSP 页面（一），让学生更加仔细的阅读，从而激发学生的学习欲望。** |
| **作业布置**（3min） | **【教师】**布置课后作业  论述JSP 标签。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**  （40min） | **【教师】**展示一个简单的 JSP 页面（二）  **三、JSP 内置对象**  1. request 对象  当客户访问服务器页面时，会提交一个 HTTP 请求，request 对象就是对 HTTP 请求包的封装。因此，使用 request 对象的方法，可以获取客户端和服务器端信息，如客户端主机名、IP 地址、传递参数名、参数值、服务器主机名和 IP 地址等。  request 对象的常用方法如下：  getProtocol( ) 方法：用于获取客户向服务器提交信息所使用的通信协议，如HTTP/1.1 等。  getServletPath( ) 方法：用于获取客户请求的 JSP 页面文件的目录。  getContentLength( ) 方法：用于获取客户提交的整个信息的长度。  getMethod( ) 方法：用于获取客户提交信息的方式，如 post 或 get。  getHeader（String s）方法：用户获取 HTTP 头文件中由参数 s 指定的头名字的值。  一般来说 s 参数可指定的头名字有 accept、referer、accept−language、content−type、accept−encoding、user−agent、host、content−length、connectiong、cookie 等。如 s 取值user−agnet 将获取客户的浏览器的版本号等信息。  getHeaderName( ) 方法：用于获取所有头名字的一个枚举。  getHeaders（String s）方法：用于获取头文件中指定头名字的全部值的一个枚举。  getRemoteAddr( ) 方法：用于获取客户的 IP 地址。  getRemoteHost( ) 方法：用于获取客户机的名称，如果获取不到，就获取 IP 地址。  getServerName( ) 方法：用于获取服务器的名称。  getServerPort( ) 方法：用于获取服务器的端口号。  getParameterNames( ) 方法：用于获取客户提交的信息体部分的 name 参数值的一个枚举。  本任务中获取客户 IP 地址和主机名就是 request 对象的一个典型应用。另外，我们还可以如下面的例子一样使用 request 对象来获取用户在表单中提交的内容。question.jsp页面中有两组单选按钮，用户提交的信息交由 answer.jsp 处理并输出结果。  question.jsp: <%@ page contentType = "text/html; charset = GB2312" %>  <html>  <body>  <form action = " answer.jsp" method = "post" name = "form">  <p> 李白是哪个朝代的人？ <br>  <input type = "radio" name = "r" value = "a"> 唐朝 <input type = "radio" name = "r" value = "b"> 宋朝 <input type = "radio" name = "r" value =  "c"> 元朝  <input type = "radio" name = "r" value = "d" checked = "ok"> 明朝  <p>《红楼梦》的作者是谁？ <br>  <input type = "radio" name = "P" value = "a"> 罗贯中 <input type = "radio" name = "P" value = "b"> 曹雪芹 <input type = "radio" name = "P" value =  "c"> 诸葛亮  <input type = "radio" name = "P" value = "d" checked = "ok"> 司马迁  <br>  <input type = "submit" value = " 提交 " name = "submit">  </form> </body> </html> answer.jsp: <%@ page contentType = "text/html;  charset = GB2312" %> <html>  <body> <% int n = 0; String s1 = request.getParameter("r"); String s2  = request.getParameter("P"); if(s1==null) s1 = ""; if(s2==null) s2 = "";  if(s1.equals("a")) n++; if(s2.equals("b")) n++;  %>  <p> 您得了 <% = n%> 分  </body>  </html>  2. response 对象  服务器响应客户，即向客户发送信息时，使用的是 HTTP 响应包，系统将 HTTP 响应包封装为 response 对象。在 JSP 页面中，可以使用 response 对象的方法动态控制响应方式，向客户端发送数据。  response 对象的常用方法如下：  addHeader（String head, String value）方法：用于向响应包中添加一个头名字（head）及对应的值（value）。  sendRedirect（URL url）方法：用于实现客户重定向。  setStatus（int n）方法：用于设置响应包的状态代码。  服务器响应客户时发送到客户端的响应包的首行被称为状态行。状态行由 3 位数的状态代码和描述状态代码的文字组成。  在下面的例子中，使用 response 对象的 setContentType 方法动态设置文件的输出类型，将当前页面保存为 Word 文档。  savefile.jsp：  <%@ page contentType = "text/html; charset = GB2312" %>  <html>  <body>  <p> 将当前页面保存为 Word 文档吗？  <form action = "" method = "get" name = "form1">  <input type = "submit" name = "submit1" value = " 保存 ">  </form>  <% String s = request.getParameter("submit1");  if(s==null) s = ""; byte b[] = s.getBytes("ISO-8859-1"); s =  new String(b); if(s.equals(" 保存 "))  response.setContentType("application/msword; charset = GB2312");  %>  </body>  </html>  3. session 对象  客户与服务器的通信是通过 HTTP 协议完成，但是 HTTP 协议是一种无状态协议，客户向服务器发送请求（request），然后服务器返回响应（response），连接就关闭了。  服务器不保留客户与服务器每一次连接的信息，因此，服务器无法判断上下两次连接是否是同一个客户。要想记住客户的连接信息，必须使用会话对象（session），session 对象记录了每个客户与服务器的连接信息。  从一个客户打开浏览器连接到服务器的某个服务目录，到客户关闭浏览器，这一过程称为一次会话。这时，在服务器端，系统为该客户创建了一个 session 对象，在客户端，系统为该客户创建了一个 Cookie 对象。一个客户对同一个服务目录中不同网页的访问属于同一对话。  当一个客户首次访问服务目录中的一个 JSP 页面时，JSP 引擎为该客户创建了一个session 对象。同一客户访问的服务目录不同，JSP 引擎为该客户创建不同的 session 对象。从时间上看，若客户离开先前的服务目录，连接到其他服务器或者服务目录，当再次回到先前服务目录时，JSP 引擎不再给客户分配新的 session 对象，而是使用先前的session 对象。只有当客户关闭浏览器或这个 session 对象达到了最大生存时间，该客户的 session 对象才被取消，这时服务器与客户的会话关系消失。  session 对象的常用方法如下：  setAttribute（String key，Object obj）方法：对象类似于散列表（hashtable），可以调用该方法将参数 obj 对象添加到 session 对象中，并为添加的对象指定一个索引关键字key。如果添加的两个对象的关键字相同，则先前添加的对象被清除。  getAttribute（String key）方法：获取 session 对象中包含的关键字为 key 的对象。由于任何对象都可以添加到 session 中，因此用该方法取回的对象应强制转换为原来的类型。  getAttributeName( ) 方法：获取 session 中包含的所有对象的一个枚举。  getCreationTime( ) 方法：获取 session 对象创建的时间，即从 1970 年 7 月 1 日午夜起至该对象创建时刻所经过的毫秒数。  getLastAccessedTime( ) 方法：获取当前 session 对象后一次被操作的时间，单位为毫秒。  setMaxInactiveInterval（int interval）方法：设置 session 对象的生存时间，单位为秒。  removeAttribute（String key）方 法： 从 当 前 session 对象中删除关键字为 key 的对象。  getId( ) 方法：获取 session 对象的编号。  invalidate( ) 方法：使 session 对象失效。  isNew( ) 方法：判断是否是一个新的客户。  下面的例子用于统计访问网站的客户数量。与例 10.1 不同的是，本例中需要判断客户是否第一次访问本站点，否则不会给计数器加 1。  NewCount.jsp：  <%@ page contentType = "text/html; charset = GB2312" %>  <html>  <body>  <%!  int number = 0;  %> <%  if(session.isNew()) number++;  %>  <p> 您是第 <% = number%> 个访问本网站的客户。  </body> </html>  4. application 对象  application 对象与 session 对象不同，它在服务器启动后就产生了，随着服务器的关闭而消失。所有客户的 application 对象是同一个，即所有客户共享一个 application 对象。application 对象的常用方法如下：  setAttribute（String key, Object obj）方法：同 session 类似，也可以添加对象到 application 中。  getAttribute（String key）方法：获取 application 中包含的关键字为 key 的对象。  getAttributeName( ) 方法：获取 application 中包含的所有对象的一个枚举。  removeAttribute（String key）方法：从当前 application 对象中删除关键字为 key 的对象。  getServletInfo( ) 方法：获取 Servlet 编译器的当前版本信息。  注意：有些服务器不直接支持使用 application 对象，必须先用 ServletContext 类声明，再使用 getServletContext( ) 方法初始化 application 对象。  getInfo.jsp：  <%@ page contentType = "text/html; charset = GB2312" %>  <html>  <body>  <h1> 读取系统信息 </h1>  <% String path = "/getInfo.jsp";  out.println("context 数据的内容：");  out.println(application.getContext(path)+"<p>");  out.println(" 文件的格式：");  out.println(application.getMimeType(path)+"<p>");  out.println(" 本页面实际路径：");  out.println(application.getRealPath(path)+"<p>");  out.println("JSP 引擎：");  out.println(getServletInfo()+"<p>");  out.println("application 对象 ID：");  out.println(getServletContext()+"<p>"); %>  </body>  </html>  5. page 对象  page 对象代表 JSP 页面本身，更确切地说，它代表 JSP 被转译后的 Servlet，其作用相当于 Java 中的 this。  下面的例子使用 page 对象的 hashCode( ) 方法和 toString( ) 方法，分别获取 page 对象的哈希代码和 ID 号。  pageobj.jsp：  <%@ page contentType = "text/html; charset = GB2312" %> <html>  <body>  <h1> 读取页面信息 </h1>  <%  int hc = page.hashCode(); String s = page.toString();  out.println("page 对象的 ID 值："+s+"<br>");  out.println("page 对象的 hash 代码："+hc+"<br>"); %>  </body>  </html>  6. out 对象  out 对象是一个输出流，用来向客户端输出数据。下面的例子用于向客户端输出一个表格，大家可以看到 out 对象的具体使用。  outStream.jsp：  <%@ page contentType = "text/html; charset = GB2312" %> <html>  <body>  <h1> 输出表格 </h1>  <%  out.println("<font face = ' 隶书 ' size = '2'>");  out.println("<table border>");  out.println("<tr>");  out.println("<th width = 80> 姓名 </th>");  out.println("<th width = 60> 性别 </th>");  out.println("<th width = 200> 出生日期 </th>");  out.println("</tr>");  out.println("<tr>");  out.println("<td> 张三 </td>");  out.println("<td> 男 </td>");  out.println("<td> 1979.8 </td>");  out.println("<td width = 100> 这是表格 </td>");  out.println("</tr>");  out.println("</table>");  out.println("</font>"); %>  </body>  </html>  **【学生】**思考、讨论。 | **通过教师讲解，了解一个简单的 JSP 页面（二）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了一个简单的 JSP 页面（二），让学生知道使用 request 对象的方法，可以获取客户端和服务器端信息。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | **【教师】**布置课后作业  论述JSP 内置对象。 | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示JSP 页面（三）  **一、操作步骤**  JSP 页面需要在支持 JSP 的 Web 服务器上才能正常运行，通常使用 Tomcat 服务器，另外也可以选择 WebLogic 或 WebSphere。  **（一）下载 Tomcat**  （1）现在使用的 Tomcat 版本号为 Tomcat 10.0.0−M1，安装文件可以在其官方网站https://tomcat.apache.org/ 去下载，如图 10−1 所示。    （2）在左侧 Download 区域，选择下载的版本是“Tomcat 10”，单击后。  **（二）环境变量配置**  （1）鼠标右键点击“此电脑”→选择“属性”→“高级系统设置”→“环境变量”，如图 10−3 所示。    （2）然 后， 点 击“系 统 变 量 ” 框 下 的“新 建 ” 按 钮， 新 建 两 个 变 量 分 别 为CATALINA——BASE 和 CATALINA\_HOME，变量值均为“E:\software\apache−tomcat−10.0.0−M1”（Tomcat 安装位置，即刚刚解压文件夹下 bin 文件所在位置）。  （3）选择“用户变量”框下的 Path 变量，点击“编辑”，在变量值中添加“;%CATALINA\_HOME%\bin”。这样 Tomcat 就配置好了。打开 cmd 命令提示符，输入 startup 后回车，就可以看到Tomcat 成功启动了。下面，将编写一个名为 Ex1001.jsp 的页面并将其放在“E:\software\apache−tomcat−10.0.0−M1\webapps\ROOT” 目 录 下， 就 可 以 在 浏 览 器 中 通 过“http://localhost:8080/ Ex1001.jsp”访问这个页面了。  **二、程序清单**  Ex1001.jsp：  <%@page contentType = "text/html; charset = GB2312"%>  <html>  <head>  <title>JSP 测试页面 </title>  </head>  <body>  <h1>JSP Page</h1> <h2> 欢迎来自  <%  String clientip = request.getRemoteAddr();  out.println(clientip);  %>  的朋友！ </h2>  <h2> 您所使用的计算机名称是  <%  String clientname = request.getRemoteHost();  out.println(clientname);  %></h2>  </body>  </html>  在浏览器中选择查看页面源代码时，看到的是这样的 HTML 代码：  <html>  <head>  <title>JSP 测试页面 </title>  </head>  <body>  <h1>JSP Page</h1>  <h2> 欢迎来自 202.168.56.23 的朋友！ </h2>  <h2> 您所使用的计算机名称是 TeServer203  </h2>  </body>  </html>  可以看到，JSP 页面中除 <%...%> 标记中的代码外，都原样出现在了客户端的浏览器里，而 <%...%> 标记中的 Java 代码则转换为了相应的文字，成了 HTML 代码的一部分。  **【学生】**思考、讨论。 | **教师通过JSP 页面（三）展示，让学生了解JSP 页面（三）的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了JSP 页面（三），让学生了解具体的任务步骤。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述程序清单。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示简单 Bean 及其使用  **任务描述**  编写一个 Bean，并在 JSP 页面中使用。  **任务目标**  1. 能够掌握 Bean 的编写特点。  2. 会访问 Bean 属性的两种方法。  3. 能够掌握 Bean 的作用域。  **预备知识**  在一个 JSP 页面中编写的代码不能在多个页面中重复使用，但可以使用 Bean 来完成这种功能。  **一、Bean 简介**  Bean 是一种软件组件，在 JSP 页面中常用 Bean 来封装事务逻辑、数据库操作等。这样，可以实现前台展示、业务逻辑、数据库操作三者的分析，使程序变得清晰，使系统变得健壮和灵活。  Bean 也是一种 Java 类，与普通 Java 类不同的是，在编写 Bean 的时候需要遵循如下的规范：  （1）Bean 必须是 public 类；  （2）Bean 必须有一个共有的无参构造函数；  （3）Bean 必须实现 Serializable 或 Externalizable 接口，以确保组件的持续性。  **二、访问 Bean 的属性**  访问 Bean 属性值的方法有两种：一种是通过 Bean 的方法访问 Bean 的属性值；一种是使用 JSP 动作标签访问 Bean 的属性值。下面介绍第二种访问 Bean 属性值的语法格式。  **（一）getProperty 标签**  getProperty 标签的作用是获得 Bean 的属性值，并将这个值以字符串方式在客户端显示。使用该标签之前，必须使用 useBean 标签创建一个 Bean 对象。getProperty 标签的语法格式如下：  <jsp:getProperty name = "Bean 对象名 " property = " 属性名 "/> 其中的 name 值必须是 useBean 标签中 id 属性的值，property 值是 Bean 的属性名，即 Bean 类定义时的成员变量。  **（二）setProperty 标签**  setProperty 标签的作用是设置 Bean 的属性值。使用该标签之前，必须使用 useBean标签创建一个 Bean 对象。setProperty 标签的语法格式如下：  <jsp:setProperty name = "Bean 对象名 " property = " 属性名 " value = " 属性值 "/>  其中的 name 值和 property 值同 getProperty 标签的一样，value 是要设置的某个属性的具体值。  使用 setProperty 标签给 Bean 属性赋值有以下三种方式：  （1）使用字符串或表达式直接给 Bean 属性赋值。这种赋值方式要求表达式的值类型与 Bean 属性的值类型相同，其语句格式如下：  <jsp:setProperty name = "Bean 对 象 名 " property = " 属 性 名 " value =  "<%expression%>"/>  （2）使用表单参数给 Bean 属性赋值。这种赋值方式要求表单中提供参数的数据组件名字与 Bean 的属性名相同，其语句格式如下：  <jsp:setProperty name = "Bean 对象名 " property = "\*"/>  该标签不用具体指定每个 Bean 的属性名，系统会自动根据表单中数据组件名与Bean 的属性名一一对应赋值。在这种方式中没有 value 属性。  （3）使用表单的参数值给 Bean 属性赋值。这种赋值方式要求表单中提供参数的数据组件名与 setProperty 标签中的 param 属性值名字相同，其语法格式如下：  <jsp:setProperty name = "Bean 对象名 " property = " 属性名 " param = " 参数名 "/>  **三、Bean 的作用域**  在 JSP 页面中使用 useBean 标签创建 Bean 对象时，标签中有一个 scope 属性，该属性指定了创建的 Bean 对象的作用范围。scope 有四种取值，分别是 request、page、session 和 application。  当 scope 取值为 request 时，对于同一客户每次不同的请求，JSP 引擎分配给客户的Bean 对象也不同。JSP 引擎对请求做出响应之后，取消分配给这个客户的这个 Bean 对象。Bean 对象的生存周期是在客户请求开始到对请求做出响应的这段时间。  当 scope 取值为 page 时，对于同一客户访问不同页面，JSP 引擎分配给客户的 Bean对象也不同。客户进入页面时 JSP 引擎给客户分配一个 Bean 对象，客户离开该页面时，JSP 引擎取消分配给客户的 Bean 对象。Bean 对象的生存周期是客户进入页面到客户离开页面的这段时间。  当 scope 取值为 session 时，对于同一客户访问不同 Web 目录下的页面，JSP 引擎为其分配不同的 Bean 对象。客户访问某个 Web 目录时 JSP 引擎给客户分配一个 Bean 对象，客户离开该 Web 目录时 JSP 引擎取消分配给客户的 Bean 对象。Bean 对象的生存周期是客户访问 Web 目录到客户离开这个 Web 目录的这段时间。  当 scope 取值为 application 时，JSP 引擎为每个客户分配同一个 Bean 对象，也就是说，所有客户共享同一个 Bean 对象。Bean 对象的生存周期是从 Bean 对象分配给客户到服务器关闭的这段时间。  **任务实施**  /\* 源程序文件名：Ex1002.java\*/  /\* 将编译后的 Bean 字节码文件 , 连同包结构 ( 目录结构 ) 复制到 Tomcat 安装目录下的  \webapps\ROOT\WEB-INF\classes 目录中。\*/  package MyPackage.Test10;  public class Ex1002  {  double radius = 0;  double circleArea = 0;  double circleLength = 0;  public Ex1002()  {  radius = 1;  }  public double getRadius()  {  return radius;  }  public void setRadius(double newRadius)  {  radius = newRadius;  }  public double getCircleArea()  {  circleArea = Math.PI\*radius\*radius;  return circleArea;  }  public double getCircleLength()  {  circleLength = 2.0\*Math.PI\*radius;  return circleLength;  }  }  BeanTest.jsp：  <%@ page contentType = "text/html; charset = GB2312" %>  <%@ page import = "MyPackage.Test10.\*" %>  <html>  <body>  <jsp:useBean id = "circle" scope = "page" class = "MyPackage.Test10.  Ex1002" />  <p> 创建了一个名为 circle 的 Bean, 其作用域为 page。  <% circle.setRadius(20);  %>  <p> 圆的半径是：  <% = circle.getRadius()%>  <p> 圆的周长是：  <% = circle.getCircleLength()%>  <p> 圆的面积是：  <% = circle.getCircleArea()%>  </body>  </html>  在JSP页面中，要正确使用Bean，必须完成以下三步骤。  1. 部署 Bean 字节码  Bean 字节码有两种部署方式：一是将 Bean 字节码部署在 Web 服务器的公有目录下，即 \Tomcat 10.0\common\classes 目录下，该目录下的 Bean 可供本服务器中所有 Web 服务目录下的 JSP 页面访问；另一种是部署在 Web 服务器的私有目录中，即 Web 服务目录的 \WEB−INF\classes 下，这里的 Bean 只能被该 Web 服务目录下的 JSP 页面访问。本任务中使用到的 Bean 就放在 Tomcat 默认 Web 服务目录下。  2. 导入 Bean  要在 JSP 页面中访问某个 Bean，则必须在 JSP 页面中导入相应的 Bean 类。如本任务中的语句：  <%@ page import = "MyPackage.Test10.\*" %>  3. 获取 Bean 对象  获取 Bean 对象在 JSP 页面中导入 Bean 类后，还要使用 useBean 标签创建一个 Bean 对象，以供 JSP 页面使用。如本任务中的语句：  <jsp:useBean id = "circle" scope = "page" class = "MyPackage.Test10.  Ex1002" />  这里使用 useBean 标签创建了一个基于类 Ex1002 的 Bean 对象 circle，scope 属性指定了 Bean 对象的作用域为 page。  下面的例子是将 Ex1002.java 改写成用 JSP 标签完成。  BeanTest.jsp <%@ page contentType = "text/html; charset = GB2312" %>  <%@ page import = "MyPackage.Test10.\*" %>  <html>  <body>  <jsp:useBean id = "circle" scope = "page" class = "MyPackage.Test10.  Ex1002" />  <p> 创建了一个名为 circle 的 Bean, 其作用域为 page。  <%circle.setRadius(10);  %>  <p> 圆的半径是：  <jsp:getProperty name = "circle" property = "radius"/>  <p> 圆的周长是：  <jsp:getProperty name = "circle" property = "circleLength"/>  <p> 圆的面积是：  <jsp:getProperty name = "circle" property = "circleArea"/>  </body>  </html>  **【学生】**思考、讨论。 | **教师通过简单 Bean 及其使用展示，让学生了解简单 Bean 及其使用的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了简单 Bean 及其使用，能够掌握 Bean 的作用域。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述Bean 的编写特点。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示基本 Servlet 程序  **任务描述**  编写 Servlet，在客户端浏览器中显示一串字符。  **任务目标**  1. 会编写 Servlet。  2. 能够了解 Servlet 类的基本结构。  3. 能够了解 Servlet 的生命周期。  **预备知识**  除了 JSP 外，还可以使用 Servlet 对客户请求做出响应。  **一、Servlet 简介**  Servlet 是使用 ServletAPI 编写的 Java 程序，是运行在 Web 服务器上的独立模块，其结构基于请求 / 响应模式。在实际应用中，可以灵活地加载和卸载 Servlet 模块，以此提高 Web 服务器的功能。  整个处理流程如下：  （1）HTTP 请求：客户将客户请求发送给 Servlet 引擎。  （2）请求：Servlet 引擎将请求发送给处理请求的 Servlet 模块。  （3）响应：Servlet 模块接受请求后，调用响应的服务（service( ) 方法）对请求进行处理，然后将结果返回给 Servlet 引擎。  （4）HTTP 响应：Servlet 引擎将结果发送给客户端。  Servlet 是 JSP 的早期版本，JSP 是 Servlet 的简易表达形式。JSP 经过转译后的结果就是 Servlet。JSP 适合于展示用户界面，Servlet 适合于企业逻辑处理。因此，常使用JSP 编写用户界面，而使用 Servlet 实现业务逻辑。  运行一个 Servlet，首先要将 Servlet 源文件编译为字节码文件，然后将字节码文件保存到相应的 Web 目录中，最后设置 Servlet 的调用路径，即配置 Web.xml 文件。  **二、Servlet 的基本结构**  Servlet 模块是用 ServletAPI 编写的。ServletAPI 包含两个包：javax.servlet 和 javax.servlet.http。javax.servlet 包中的类与 http 协议无关；javax.servlet.http 包中的类与 http 协议相关，该包中的部分类继承了 javax.servlet 中的部分类和接口。下面介绍编写 Servelt时要继承的父类即 GenericServlet 和 HttpServlet。  **（一）GenericServlet**  GenericServlet 类在 javax.servlet 包中，它提供了 Servlet 接口的基本实现，该类包含三个重要的方法，它们是 init( ) 方法、destroy( ) 方法和 service( ) 方法。service( ) 方法是  抽象方法，所有子类都应当实现这个方法。  **（二）HttpServlet**  HttpServlet 类在 javax.servlet.http 包中，它扩展了 GenericServlet 类。HttpServlet 类定义了 2 个 service( ) 方法和 6 个 doXxx( ) 方法。  第一个 service( ) 方法：  public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException{}  这是一个公有方法，该方法接收客户端请求包后，创建 request 对象和 response 对象，并分别转换为 HttpServletRequest 和 HttpServletResponse 对象，然后调用第二个service( ) 方法。  第二个 service( ) 方法：  protected void service(HttpServletRequest request, HttpServletResponse  response) throws ServletException, IOException{}  该方法接收 HttpServletRequest/HttpServletResponse 类型的对象后，根据 http 请求方法的类型，调用下面 doXxx( ) 方法之一，进行逻辑处理并响应客户端。  doXxx( ) 方法有 6 个：doGet( ) 方 法、doPost( ) 方 法、doDelete( ) 方 法、doPut( ) 方法、doTrace( ) 方法和 doOption( ) 方法。它们分别对应表单中的 6 种 method 属性。  doGet( ) 方法：  protected void doPost(HttpServletRequest request, HttpServletResponse  response) throws ServletException, IOException{}  doPost( ) 方法：  protected void doPost(HttpServletRequest request, HttpServletResponse  response) throws ServletException, IOException{}  程序员在编写自定义的 Servlet 类时，都必须扩展 HttpServlet 类。在扩展类中，覆盖 service( ) 方法、doPost( ) 方法或 doGet( ) 方法中的任意一个，Servlet 执行 service( ) 方法时，会自动调用 doPost( ) 方法或 doGet( ) 方法实现逻辑处理。  **三、Servlet 生命周期**  当服务器调用 Servlet 类时，Servlet 对象被创建。从服务器创建 Servlet 对象到该对象被撤销这段时间称为 Servlet 生命周期。  init( ) 方法：当 Servlet 第一次被加载时，服务器调用本方法创建一个 Servlet 对象，完成必要的初始化工作。在 Servlet 对象生命周期内，本方法只调用一次。  service( ) 方法：每当客户请求到来时，Servlet 引擎将请求对象传递给本方法，同时创建一个响应对象，service( ) 方法获得请求 / 响应对象后，进行请求处理（调用被覆盖的 doXxx( ) 方法），将处理的结果以响应对象的方式返回给客户端。在 Servlet 对象生命周期内，本方法可能被多次请求，多次调用。destroy( ) 方法：Servlet 引擎终止服务时（如关闭服务器），调用本方法撤销 Servlet 对象。在 Servlet 对象生命周期内，本方法只调用一次。  **任务实施**  **一、操作步骤**  在编译 Servlet 源文件的时候，需要使用到 ServletAPI 基本包，这些包在 servlet−api.jar 文件中，我们可以在 \Tomcat 10.0\common\lib 文件夹下找到该文件。在使用Eclipse 作为 IDE 时，需要在项目中进行一些配置，具体步骤如下。  （1）在 Eclipse 的“Package Explorer”视图中，右键单击项目名称，在弹出的快捷菜单中选择“Properties”选项，如图 10−8 所示。  （2）在如图 10−9 的对话框中，先选择左边列表中的“JavaBuildPath”，在右边的选项卡中选择“Libraries”，再单击“AddExternalJARs...”按钮，在弹出的选择文件对话框中选择 \Tomcat 10.0\common\lib 文件夹下的 servlet−api.jar 文件，将该文件添加到列表中。  编译好 Servlet 字节码文件后，将其复制到 \Tomcat 10.0\webapps \ROOT\WEB−INF\classes 目录下。用记事本或其他文本编辑软件打开 \Tomcat 10.0\webapps\ROOT \WEB−INF 下的Web.xml 文件，在 <!− − JSPC servlet mappings start − −> 和 <!− − JSPC servlet mappings end − −> 标记之间增加以下代码：  <servlet>  <servlet-name>Servlet.Test</servlet-name>  <servlet-class>Ex1003</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>Servlet.Test</servlet-name>  <url-pattern>/SerTest</url-pattern>  </servlet-mapping>  这里的 Servlet.Test 是为 Servlet 类起的名字，Ex1003 是 Servlet 的字节码文件名，/SerTest 是运行 Servlet 类的路径名。  保存 Web.xml 文件后，重新启动 Tomcat 服务器，然后在浏览器地址栏中输入http://localhost:8080/SerTest 就可以访问该 Servlet 了。  **二、程序清单**  /\* 源程序文件名：Ex1003.java\*/  import java.io.\*;  import javax.servlet.\*;  import javax.servlet.http.\*;  public class Ex1003 extends HttpServlet  {  public void init(ServletConfig config) throws ServletException  {  super.init(config);  }  public void service(HttpServletRequest request, HttpServletResponse  response) throws IOException  {  response.setContentType("text/html; charset = GB2312");  PrintWriter out = response.getWriter();  out.print("<HTML><BODY>");  out.print("<font size = '6' color = 'red'> 将本字符串用红色字体输出到客户  端 </font>");  out.print("</BODY></HTML>");  }  }  可以看到，Servlet 程序的作用就是生成 HTML 页面。在本任务中，最终用户所看  到的 HTML 页面中的所有代码都可以在 Servlet 程序的相关输出代码中找到。  response.setContentType("text/html; charset = GB2312");  该语句同本单元任务 1 中 savefile.jsp 中的类似语句一样，用于设置 response 对象  的 ContentType 属 性， 相 当 于 jsp 页面中 <%@page contentType=”text/html; charset=  GB2312”%> 语句的作用。  out.print("<HTML><BODY>");  out.print("<font size = '6' color = 'red'>将本字符串用红色字体输出到客户端</  font>");  out.print("</BODY></HTML>");  这三条语句用于输出 HTML 代码，输出内容和我们在客户端浏览器中查看该页面  源代码时所看到的一样。  **【学生】**思考、讨论。 | **教师通过基本 Servlet 程序展示，让学生了解基本 Servlet 程序的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了基本 Servlet 程序，能够了解 Servlet 的生命周期。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述Servlet 类的基本结构。** | 通过课后练习，使学生巩固所学新知识 |
| **知识讲解**（45min） | **【教师】**展示思考与练习  **一、简答题**  （1）JSP 程序片包括哪几类？各自的特点是什么？  （2）JSP 的内置对象中，request 对象、response 对象、session 对象和 application对象各有什么区别和共同点？  （3）Bean 是什么？在 JSP 编程中通常起什么作用？  （4）访问 Bean 属性有哪几种方式？举例说明。  （5）Servlet 和 JSP 有何区别？尝试将一个 JSP 页面改成用 Servlet 实现。  （6）Servlet 生命周期在实际应用中有什么作用？  **二、选择题**  （1）JSP 标签用于控制页面属性，下列选项不属于 JSP 标签的是（　　）。  A. 注释 B. 指令标签 C. 动作标签 D. 标题标签  （2）JSP 高效运行是通过（　　）文件常驻内存来实现的。  A. .class B. .java C. .html D. .jsp  **实训任务**  （1）编写 JSP 页面，定义一个员工类，在页面中可以输入新员工的信息，并显示所有员工的信息。  （2）编写 JSP 页面和定义员工类的 Bean，实现和题（1）类似的功能。  （3）使用 Servlet 和 Bean 实现题（1）的功能。  **【学生】**思考、讨论。 | **教师通过思考与练习展示，让学生了解思考与练习的基本理论知识。** |
| **课堂小结**  （3min） | 【**教师**】**回顾和总结本节课的知识点。**  **这节课我们一起学习了思考与练习，能够让学生巩固学习知识。** | 通过对所学知识的回顾，培养学生的归纳总结能力 |
| **作业布置**（2min） | 【**教师**】**布置课后作业**  **简述 Tomcat 服务器的安装和配置过程。** | 通过课后练习，使学生巩固所学新知识 |
| **教学反思** | 课堂教学实践中，教师要时刻关注学生的学习过程，关注所使用的方法和手段以及达到的效果，捕捉教学中的灵感，及时调整设计思路和方法，使课堂教学效果达到最佳。 | |