Servlet：

内容讲解：

·Servlet简介

·Servlet程序开发与部署

·Servlet操作request.response.session.application

·Servlet的跳转

具体内容：

什么是Servlet：

之前我们在JESE学习中，介绍过applet，这种程序是最早形态的程序，称为java小程序

Servlet：服务端小程序

Servlet是最早使用java 实现CGI（通用网关接口）程序，但是与传统的CGI不同的是，Servlet采用的是多线程的处理方式，而传统的CGI采用的是多进程的处理方式，Servlet处理速度要比传统的CGI快很多。

Servlet 是sun公司最早推出的实现ＷＥＢ的程序，但是其本身的代码开发很困难，代码量非常大，而且不易显示，因为所有的ＨＴＭＬ都是通过Ｓｅｒｖｌｅｔ中使用out.print一行一行输出的。之后SUN公司为了解决这样的问题，吸收了ASP的技术特点，在HTML中使用大量的java代码，以scriptlet形式出现，这才有JSP

开发Servlet程序：

开发要求：

·所有的Servlet程序必须放在ＷＥＢ目录中执行，依靠ＷＥＢ容器执行

·Servlet本身是一个类，本身不需要由主方法调用执行

·Servlet与javaBean一样，编译后好的calss保存在WEB-INF\classes目录中

·所有的Servlet类必须继承HttpServlet,这个类才称为是一个Servlet程序

·Servlet必须放在一个包中

第一个Servlet程序：

执行页面显示“HELLO WORLD” ，

程序中必须覆写HttpServlet中doGet方法，方法定义如下：

protected void **doGet**([HttpServletRequest](mk:@MSITStore:F:\Workspaces\教育\JT39\课堂笔记\J2EE\J2EE_5.0.CHM::/javax/servlet/http/HttpServletRequest.html) req,

[HttpServletResponse](mk:@MSITStore:F:\Workspaces\教育\JT39\课堂笔记\J2EE\J2EE_5.0.CHM::/javax/servlet/http/HttpServletResponse.html) resp)

throws [ServletException](mk:@MSITStore:F:\Workspaces\教育\JT39\课堂笔记\J2EE\J2EE_5.0.CHM::/javax/servlet/ServletException.html),

[IOException](http://java.sun.com/j2se/1.5/docs/api/java/io/IOException.html)

|  |
| --- |
| **package** com.jjm.servlet;  **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** ServletDemo01 **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest req, HttpServletResponse resp)  **throws** ServletException, IOException {  PrintWriter out = **null**;  out = resp.getWriter();  out.print("<html>");  out.print("<body>");  out.print("<h1>");  out.print("Hello World");  out.print("</h1>");  out.print("</body>");  out.print("</html>");  }  } |

现在Servlet程序已经写好了，怎么用网页的形式打开呢？需要配置web.xml中的映射

|  |
| --- |
| <servlet>  <servlet-name>hello</servlet-name>  <servlet-class>com.jjm.servlet.ServletDemo01</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>hello</servlet-name>  <url-pattern>/servletdemo01</url-pattern>  </servlet-mapping> |

程序已经正常输出Hello World，如果一个Servlet想要输出一个完整的页面，则代码量肯定非常的多，而且非常麻烦。

一个servlet程序可以配置多个映射地址

Servlet接收表单参数：

MyEclipse可以自动配置web.xml

使用request对象接收表单传递的参数

|  |
| --- |
| <%@page contentType=*"text/html; charset=GBK"*%>  <form action=*"../servlet/demo02"* method=*"post"*>  <table>  <tr>  <td>  姓名：<input type=*"text"* name=*"name"*/>  </td>  </tr>  <tr>  <td>  <input type=*"submit"* value=*"提交"*/>  </td>  </tr>  </table>  </form> |

|  |
| --- |
| **package** com.jjm.servlet;  **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** ServletDemo02 **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest req, HttpServletResponse resp)  **throws** ServletException, IOException {  System.*out*.println("这是doGet方法");  System.*out*.println(**new** String(req.getParameter("name").getBytes("ISO-8859-1")));  }  **public** **void** doPost(HttpServletRequest req, HttpServletResponse resp)  **throws** ServletException, IOException {  System.*out*.println("这是doPost方法");  **this**.doGet(req, resp);  }  } |

关于路径配置导致404错误的问题，

最好在映射目录中不要配置多层目录：

|  |
| --- |
| <servlet-mapping>  <servlet-name>ServletDemo02</servlet-name>  <url-pattern>/demo02</url-pattern>  </servlet-mapping> |

../表示的是上一层目录，如果是../../表示的是上两次的目录

Servlet生命周期：

Web容器也是有生命周期的，此生命周期在Servlet中是比较明显的，如果想要了解Servlet的声明周期，则可以通过Servlet本身提供的方法

Servlet生命周期的流程：

创建——》初始化——》服务（多次）——》销毁

一个妇女孕育生命——出生（一次）——成长、生活（多次）——》死亡（一次）

如果一个人长时间消失（黑户），可以认为是死亡了，但是若干年之后在人间复活（从新上户籍）

Servlet与人的生命是一样的，Servlet也只关心初始化服务和销毁，对应的有三个操作方法：

初始化：正常情况下一个Servlet只有一次初始化，但是一个Servlet长时间不使用之后再次使用，则会重新初始化，

public void init() throws ServletException：如果不取得初始化参数的话使用此方法进行初始化

public void init(ServletConfig config) throws ServletException：可以取得初始化参数

服务：如果一个Servlet长时间没有服务，则认为没用了，等待销毁

public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException

但是一般情况下不回去覆写service方法，而是使用：

**public** **void** doGet(HttpServletRequest req, HttpServletResponse resp)

**throws** ServletException, IOException {}

**public** **void** doPost(HttpServletRequest req, HttpServletResponse resp)

**throws** ServletException, IOException {}

销毁：正常情况下销毁只有一次，容器关闭时销毁，当然一个Servlet长时间不用，也会销毁

public void destroy()

下面我们通过程序来说明生命周期：

|  |
| --- |
| package com.jjm.servlet;  import java.io.IOException;  import java.io.PrintWriter;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  public class ServletDemo03 extends HttpServlet {  public void destroy() {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*销毁\*\*\*\*\*\*\*\*\*\*\*\*");  try {  Thread.sleep(3000);  } catch (InterruptedException e) {  e.printStackTrace();  }  }  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*服务\*\*\*\*\*\*\*\*\*\*\*\*");  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  this.doGet(request, response);  }  public void init() throws ServletException {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*初始化\*\*\*\*\*\*\*\*\*\*\*\*");  }  } |

在Servlet中取得初始化参数：

在web.xml中配置参数

|  |
| --- |
| <servlet>  <description>This is the description of my J2EE component</description>  <display-name>This is the display name of my J2EE component</display-name>  <servlet-name>ServletDemo03</servlet-name>  <servlet-class>com.jjm.servlet.ServletDemo03</servlet-class>  <init-param>  <param-name>name</param-name>  <param-value>www.wanczy.com</param-value>  </init-param>  </servlet> |

|  |
| --- |
| package com.jjm.servlet;  import java.io.IOException;  import javax.servlet.ServletConfig;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  public class ServletDemo03 extends HttpServlet {  public void destroy() {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*销毁\*\*\*\*\*\*\*\*\*\*\*\*");  try {  Thread.sleep(3000);  } catch (InterruptedException e) {  e.printStackTrace();  }  }  // public void doGet(HttpServletRequest request, HttpServletResponse response)  // throws ServletException, IOException {  // System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*服务\*\*\*\*\*\*\*\*\*\*\*\*");  // }  // public void doPost(HttpServletRequest request, HttpServletResponse response)  // throws ServletException, IOException {  // this.doGet(request, response);  // }  public void init() throws ServletException {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*初始化\*\*\*\*\*\*\*\*\*\*\*\*");  }  public void init(ServletConfig config) throws ServletException{  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*初始化取得参数\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(config.getInitParameter("name"));    }  public void service(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException{  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*服务\*\*\*\*\*\*\*\*\*\*\*\*");    }  } |

如果有参数，写了两个初始化方法，则会执行能够接收参数的初始化方法

如果我现在在Servlet程序中存在doGet()和doPost()方法，也存在service方法，则肯定不会去执行doGet()和doPost()，原因是因为doGet()和doPost()方法都是在父类中使用service方法进行区分的，相当于以下形式

容器——service——各个子类的doXxx（）方法，但是在一般情况下不会去覆写service方法而是使用doXxx方法。

一般情况下，Servlet初始化是在Servlet程序第一次运行的时候初始化，当然也可以进行配置，让Servlet在服务器启动的时候就进行初始化操作，修改web.xml文件

|  |
| --- |
| <servlet>  <description>This is the description of my J2EE component</description>  <display-name>This is the display name of my J2EE component</display-name>  <servlet-name>ServletDemo03</servlet-name>  <servlet-class>com.jjm.servlet.ServletDemo03</servlet-class>  <init-param>  <param-name>name</param-name>  <param-value>www.wanczy.com</param-value>  </init-param>  **<load-on-startup>1</load-on-startup>**  </servlet> |

使用Servlet获取其他常用的对象：

request、response、session、application、config

从Servlet本身的定义的方法里面可以发现，doGet或者doPost直接存在request和response对象，那么我们现在关心的只有session和application

取得session对象：

HttpSession是其类型，如果在一个Servlet中想要取得session对象，则必须依靠HttpServletRequest对象完成，观察HttpServletRequest：

·HttpSession getSession()：表示直接返回当前的session

·HttpSession getSession(boolean create)：取得当前的session，如果session不存在，则创建（create=true）

|  |
| --- |
| package com.jjm.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  public class ServletDemo02 extends HttpServlet {  public void doGet(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doGet方法");  System.out.println(new String(req.getParameter("name").getBytes("ISO-8859-1")));  //取得session  HttpSession session = req.getSession(true);  System.out.println(session.getId());    }  public void doPost(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doPost方法");  this.doGet(req, resp);  }  } |

取得application对象：

application使用于取得虚拟目录的真实路径，getServletContext()方法取代application，现在在Servlet中取得application对象依然使用此操作

|  |
| --- |
| package com.jjm.servlet;  import java.io.IOException;  import javax.servlet.ServletContext;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  public class ServletDemo02 extends HttpServlet {  public void doGet(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doGet方法");  System.out.println(new String(req.getParameter("name").getBytes("ISO-8859-1")));  // //取得session  // HttpSession session = req.getSession(true);  // System.out.println(session.getId());  //取得application  ServletContext app = this.getServletContext();  System.out.println(app.getRealPath("/"));    }  public void doPost(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doPost方法");  this.doGet(req, resp);  }  } |

Servlet跳转：

我们说一个jsp页面可以轻松的跳转到Servlet，但是现在要使用Servlet跳转到其他地方呢？Servlet 有两种跳转方式：

·客户端跳转

在Servlet中的doget或者dopost方法之中存在了response对象，则直接可以使用此对象的sendRedirect(“跳转的页面”);

|  |
| --- |
| package com.jjm.servlet;  import java.io.IOException;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  public class ServletDemo02 extends HttpServlet {  public void doGet(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doGet方法");  System.out.println(new String(req.getParameter("name").getBytes("ISO-8859-1")));  //取得session  HttpSession session = req.getSession(true);  // System.out.println(session.getId());  //取得application  // ServletContext app = this.getServletContext();  // System.out.println(app.getRealPath("/"));    //表示取得地址栏路径  String path = req.getContextPath();  String basePath = req.getScheme()+"://"+req.getServerName()+":"+req.getServerPort()+path+"/";  **session.setAttribute("name", new String(req.getParameter("name").getBytes("ISO-8859-1")));//设置属性**    **resp.sendRedirect(basePath+"CRUSD/Demo09.jsp");//跳转成功**    }  public void doPost(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doPost方法");  this.doGet(req, resp);  }  } |

以上已经完成了客户端跳转功能，并且传递了属性，通过session在Servlet设置属性，跳转后页面通过session取得属性。

但是存在一个问题，属性只接受一次的话，使用session是不是太浪费了。

既然是跳转而且属性只接受一次，是不是使用request传递会更适合，但是使用resquest接受属性必须要使用服务端跳转

服务端跳转：

在Servlet中，ResquestDispatcher（）主要的功能就是完成服务端跳转，实际上就是在Servlet中完成<jsp:forward page=”页面”> 功能。

|  |
| --- |
| package com.jjm.servlet;  import java.io.IOException;  import javax.servlet.RequestDispatcher;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  public class ServletDemo02 extends HttpServlet {  public void doGet(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doGet方法");  System.out.println(new String(req.getParameter("name").getBytes("ISO-8859-1")));  //取得session  HttpSession session = req.getSession(true);  // System.out.println(session.getId());  //取得application  // ServletContext app = this.getServletContext();  // System.out.println(app.getRealPath("/"));    //表示取得地址栏路径  String path = req.getContextPath();  String basePath = req.getScheme()+"://"+req.getServerName()+":"+req.getServerPort()+path+"/";  //session.setAttribute("name", new String(req.getParameter("name").getBytes("ISO-8859-1")));  //resp.sendRedirect(basePath+"CRUSD/Demo09.jsp");//跳转成功  req.setAttribute("name", new String(req.getParameter("name").getBytes("ISO-8859-1")));  //服务端跳转 start  RequestDispatcher rd = req.getRequestDispatcher("/CRUSD/Demo09.jsp");  rd.forward(req, resp);  // req.getRequestDispatcher("/CRUSD/Demo09.jsp").forward(req, resp);  //end    }  public void doPost(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {  System.out.println("这是doPost方法");  this.doGet(req, resp);  }  } |

在一般的Servlet开发中，使用RequestDispatcher接口跳转是最多的，而且一般向页面传递参数都是使用request范围属性。

总结：

·Servlet开发形式

·Servlet生命周期

·弄清楚service和doget、dopost之间的关系

·Servlet可以取得session和application对象

·RequestDispatcher接口实现服务端跳转并传递request范围属性