Java\_Web\_开发

第一章 Web应用程序----概述

在这一章里面，我们要学习以下几点：

1:Web应用程序与桌面应用程序的区别

2:Web服务器与客户端

3:Http协议

4:Web应用程序体系结构

5:Web应用程序的目录结构

6:Get与Post提交方式

7:用Jbulider开发第一个Web程序

8:Web应用程序开发过程

9:如何在TomCat5.0中配置Servlet

Web应用程序与桌面应用程序的区别

桌面应用程序：

在台式电脑上运行的应用程序称为桌面应用程序。比如：记事本，Word,Excel。

这种应用程序的特点是

1)缺少共享 2)直接双击就可以运行，不需要IE 3)一般以window的窗口形式存在

Web应用程序：

1)需要IE来执行 2)一般以网页的形式的存在 3)可以有很多的客户端来共享。

比如： www.sina.com

Web应用程序的优点：

1)能够实现最大的资源共享，

比如：开发一个网站，放在Internet上，就可以让全世界的人都来共享

2)维护与部署起来成本低

Web应用程序只需要在服务器上安装一次，就可以通过网络给所有客户端共享，以后要修改

Web应用程序，只需要修改服务器上Web应用程序，则所有的客户端自动进行更新。而桌面

应用程序必须在每台计算机上都安装一次，以后要修改时，每台计算机都要修改

Web应用程序的缺点：

响应速度比桌面应用程序要低很多，因为要通过Internet去访问远程网页.

Web服务器与客户端

Web服务器：

存放网页(\*.jsp)的那一端被称为Web服务器

Web客户端：

浏览网页的那一端被称为客户端

工作原理：

客户端利用IE,通过Internet向Web服务器发出请求(Request),Web服务器在接到请求之后，进行处理

然后再把处理之后的结果通过Internet传给客户端，客户端再通过IE显示出来。我们把服务器接受

客户端的请求并进行处理的这个过程称为响应(Response)。

注意：请求是发给服务器的，而响应是发给客户端的

Http协议

Http协议：

HyperText Transport Protocol 超文本传输协议。是客户端与服务器进行交互所共同遵守的一个规则。

可以把购房者看成是客户端把房产开发商看成是服务器端，购房者要能从开发商那里顺利的买到房子，

同时开发商要能够顺利地把房子卖给购房者，购房者与开发商必须要遵守购房合同。这个合同就是所

谓的协议。

Http协议端口：

客户端要把请求发给服务器，服务器要把处理数据发给客户端，Http协议本身是办不到的。Http协议只是

规定了客户端与服务器发送数据的规则，格式。为了能够实现数据的发送与接受，就要在客户端与服务器 之间建立一个通道，让客户端与服务器通过这个通道来传输数据。这个通道在Web应用程序中被称为端口号

Web应用程序的端口号默认为80。当客户端通过IE去游览网页时，要按照以下格式输入：

http://网站名:端口号--------->比如：http://www.sina.com:80。只不过由于80端口号是默认的所

以可以省略这个端口号

Http协议是无状态的：

客户端使用Http协议通过端口号80向Web服务器发出请求，Web服务器接受请求，并与客户端创建一个连接

Web服务器进行数据处理，并把处理的结果传给客户端。一旦应答了客户的请求，服务器则关闭，这样以前 与客户端的连接的信息就全部都丢失了。所以Htpp被称为没有状态。相当于你到菜场去买菜，你是客户端 菜贩是服务器，一旦你与菜贩完成一次买菜交易，菜贩是不会把你买菜的过程记录下来的。这样即使你事 发现找钱错了，也无法证明

Http协议的状态码：

200:代表处理成功

404:指定客户输入的网页不存在

Web应用程序体系结构

表示层：用户界面

业务层：包含系统业务的功能，就是用一些类去封装

数据层: 具体与数据进行交互的类

表示层不能直接向数据层发请求，表示层应把处理请求发给数据层，数据层再把请求发给数据层

数据层进行处理之后，把结果传给业务层，业务层把数据再传给表示层。此种模式也被称为MVC模式

优点：

各层之间相互独立，一个层出现错误不会影响其它层

Web应用程序的目录结构

用Java技术去开发Web应用程序只有两种类型应用程序。一个是servlet一个是Jsp。最早期都是

servlet， Jsp是在servlet的基础上发展起来的。servlet就是一个java类，它接受客户的请求并进行 处理，然后把处理的结果返回给客户端,这个类与普通的java类的区别是,普通的java类可以运行在单机 上而servlet只能够运行在Internet上。

一个Web应用程序的目录结构：

顶层目录：假设为 Book

/\*.html

/\*.jsp

/images/\*.jpg

/WEB-INF/WEB.XML

/WEB-INF/classes/\*.class

Get与Post提交方式

客户端向服务器发出请求(Request)有两种方式。一种是Get方式与post方式。它们的区别是

Get提交:

A:把要传递的数据，放在查询字符串(URL)的后面，传递的数据是可见的。

比如：网页名?变量名=值1&变量名=值2

B:传递的数据大小有限制

C:由于数据是可见的，所以安全性比较低

D:Get提交方式仅适合于传递数据量少，且数据安全性要求不高的数据

Post提交:

A:把要传递的数据放在表单的元素中，通过表单传给服务器,传递的数据是不可见的

B:对传递的数据大小没有限制

C:适合于传递安全性比较高的数据。比如：传递密码

如何设置一个页面的传递方式：

A:设置为post方式:

<form action="目的页面" method="post">

B:设置为Get方式: 通常用超链接

<a herf="目的页面?变量名=值">

用Jbulider开发第一个Web程序

步骤：

1)先创建一个工程

2)再创建一个Web模块

3)创建一个标准的Servlet

9)运行

Web应用程序开发过程

1)设计目录结构：

2)编写Web应用程序。包括Servlet类、Jsp页面、Html页面

3)编写部署描述文件：Web.xml

4)打包Web应用程序。将所有与Web应用程序相关的文件打包为一个\*.war文件

5)部署Web应用程序。将步骤4中打包的\*.war文件安装到目的计算机中

如何在TomCat5.0中配置Servlet

1)什么是TomCat

TomCat就是一个Web应用程序的容器。Web应用程序必须要在Web应用程序的容器的支持下才能够正确运行。目录有很多的Web容器。Tomat,Welogic,resin,websphere等。最流行是

容是TomCat

2)在安装了Jublider 之后，系统在Borland\JBuilder2006\thirdparty下面会有两种版本的 TomCat。Tomcat 4.0 与Tomcat 5.0。

3)在安装Tomcat的目录下面D:\Borland\JBuilder2006\thirdparty\jakarta-tomcat-5.5.9\bin找到"startup.bat"启动文件

加上下面代码

rem ---------------------------------------------------------------------------

set JAVA\_HOME=D:\Borland\JBuilder2006\jdk1.5 \*\*\*\*\*\*\*\*\*\*\*

rem Guess CATALINA\_HOME if not defined

4)用Jbudlider创建一个标准的Servlet，假设为"MyServlet",模块文件为"WebMod"

4)找到在创建Servlet时指定的Web模块文件夹"C:\Demo\WebModel\WEB-INF\classes"

5)在TomCat的\conf\Catalina\localhost文件夹下面创建一个与Web模块名相同的

Xml文件，在这里面应该是WebModel.xml。在这个文件里面添加如下内容：

<Context path="/WebModel" docBase="C:\Demo\WebModel" debug="0"

虚拟目录 实际的物理路径

reloadable="true" privileged="true" />

6)在WEB-INF下面创建一个web.xml(部署描述文件)，输入以下内容：

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd" version="2.4">

<display-name>WebModel</display-name>

<servlet>

<servlet-name>MyServlet</servlet-name>

<servlet-class>demo.MyServlet</servlet-class>

其中demo是创建MyServlet时指定的包名，如果没有包名

则直接写上"MyServlet"

</servlet>

<servlet-mapping>

<servlet-name>MyServlet</servlet-name>

<url-pattern>/MyServlet</url-pattern>

</servlet-mapping>

</web-app>

不过这个文件Jbulider已经绑我们做好了，我们不需要修改

7)启动TomCat， 在IE中输入：

http://localhost:8080/WebModel/MyServlet来访问

8)上面的配置只适合于开发阶段调试时使用，一旦Web应用程序开发完毕之后，要进行移植时，

就要把Web应用程序打包成War文件(Web 归档文件)。

具体步骤：

A)进入开发Web应用程序的模块文件下面。比如为：WebModel下面

B)执行jar cvf WebModel.war \*.\* (注意要先设置Jdk环境变量，jar是jdk的一个工具)

C)在模块文件下面就会产生一个WebModel.war文件

D)把WebModel.war文件复制到Tomcat的webapps下面。

E)启动Tomcat，TomCat会自动把WebModel.war解压成为一个WebModel文件夹，

\*\*\*\*\*注意要删除在TomCat的\conf\Catalina\localhost文件夹下面创建一个与Web模块名相同的Xml文件，否则不会解压

F)在IE里面执行：

http://localhost:8080/WebMod[模块名]/MyServlet[Servlet类名]

注意Servlet会区分大小写

http://localhost:8080/webmod[模块名]/myservlet[Servlet类名] 会出现错误

第二章 Servlet使用入门

在这一章里面我们将要学习以下几点内容：

1:CGI与Servlet的区别

2:Servlet体系结构

3:Servlet的基础知识

4:Servlet的输出

5:Servlet的生命周期

6:HttpServletResponse与HttpServletRequest几个重要方法

CGI与Servlet的区别

1)在最早期的Web应用程序主要是CGI(Common GateWay Interface 通用网关接口)。

2)这类Web应用程序是使用如Perl这样的语言来编写的。

3)CGI有个致命的缺点，就是对每一个客户端，必须创建一个新的实例，这将占用大量的内存

4)为了解决这个问题，Servlet出现了。

5)Servlet是一个用Java编写的类，在服务器上运行，处理客户端的请求，并把结果放给客户端。

6)对于每一个客户端的请求，只需要创建一个Servlet的实例。从而大大的节省了内存

Servlet体系结构

客户端：

客户端----------->Web服务器----->Web容器---->Servlet

http请求

服务器端：

Servlet------->Web容器----->Web服务器---->客户端

Servlet的基础知识

1)要编写一个Servlet，一般继承自HttpServlet抽像类。所以一般要导入包javax.servlet.\*与javax.servlet.http.\*;

2)自定义的Servlet一般要重写doGet()方法和doPost()方法

3)doGet方法用来处理客户端通过Get方法发出的请求

public void doGet(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException

4)doPost()方法用来处理客户端通过Post方法发出的请求

public void doPost(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException

5)request参数用来表示客户端的请求，response参数用来服务器对客户端的响应

6)为了让一个Serlvet能够处理Get请求与Post请求。一般会在doPost的方法里面写上

doGet(request, response)表示把所有的请求转化成为Get请求

Servlet的输出

1)Servlet可以通过两种方法来输出运算的结果

A:通过System.out.Print方法。此方法会把结果输出到Cmd中，也就是TomCat中。

B:通过response参数的getWriter()方法得到一个PrintWriter对像进行输出，

此输出会把计算结果输出到IE上，一般用此方法

PrintWriter out = response.getWriter();

out.println

Servlet的生命周期

1)Servlet的生命周期是指从一个Servle的创建开始到消亡的全过程

2)Servlet的生命周期各阶段如下：

A:实例化:也就是创建一个Servlet类的实例

B:初始化:向window申请资源。此时会自动去执行Init方法

C:服务:去不断地接受客户端的请求，并做出响应

D:释放:把以前分配的资源进行归还给操作系统，此时会自动执行destroy方法

举例：

做一个servlet计数器，用来输出servlet被调用了多少次

步骤：

1)定义一个全局变量

2)在Init中赋初值为0

3)在doGet把变量加1, 并输出

提问:

1)同时打开另外一个IE浏览器，做一个新的客户端，讨论计数器会不会清零

结论：不管有多少个客户端，都只有一份实例。

2)讨论什么时候计数器才会清零,关闭TomCat容器

结论：说明Web容器一旦关闭，则容器里面的servlet也就会消亡

HttpServletResponse与HttpServletRequest几个重要方法

1:HttpServletResponse

A:response.setContentType("编码")

设置向客户端输出的文本的编码，一般设为"text/html; charset=GBK"

response.setContentType("text/html; charset=GBK");

如果不设置，可能会出现乱码

B:PrintWriter out = response.getWriter();

产生一个PrintWriter对像，此对像能够把信息输入到网页中去

此对像不仅可以输出普通字符，还可以输出html标记

在不使用此对像时，应该关闭该对像。out.close();

C:response.sendRedirect("Index.html");

使页面进行跳转

2:HttpServletRequest

A:request.getParameter("变量名")

B:如果接受中文再输出会是乱码，为了解决这个问题，可以在接受之间加上

request.setCharacterEncoding("GBK");

request.getParameter("变量名")

可以得到客户端以Post或Get方法提交过来的变量

举例：

1:创建一个登录的Html页面，通过Post提交到Servlet中，通过Servlet来判断数据库中用户名是否正确,

如果正确则跳到主页面，否则打印错误，并提供"返回"超链接

2:编写一个用于注册会员的html页面，让用户填写会员的资料后，能够把这些资料保存到数据库中去

第三章 会话跟踪简介

在这一章里面我们将要学习以下几点内容：

1:Servlet中页面跳转的方法

2:Sevlet中session对像

3:Serlvet中Cookie对像

4:Serlvet上下文

Servlet中页面跳转的方法

1:request.getRequestDispatcher("目标").forward(request,response);

2:response.sendRedirect("目标");

3:forward与sendRedirect的区别：

A:forward跳转之后在地址栏中显示依然是旧网页的地址,sendRedirect显示的是新网页的地址

B:forward只能跳转到当前Web程序中的页面，而sendRedirect可以跳到任何目录。

比如：

request.getRequestDispatcher("http://www.sina.com").forward(request,response);错误

response.sendRedirect("http://www.sina.com"); 正确

什么是Session对像

1:首先你应该把session想像成为一个大盒子,里面用来存放客户的数据

2:当你打开一个网站的首页时,服务器就会为你创建一个大盒子,并且为这个大盒子编一个号,

于是这个大盒子就可以让你在这个网站的不同页面之间共享，直到你离开这个网站的时候,这服务器又会 收回这个大盒子

3:可见session对像的其中的一个功能就是为了实现让同一个客户端在不同页面之间实现数据共享.

但是一旦用户离开这个网站,那么session也就会自动消失

4:一个网站可以被很多人同时访问,访问的每一个人在打开主页时,都会有一个大盒子,即session.

如果另一个用户也同时访问网站，他也拥有自己的Session，但两个用户之间无法通过Session共享信息

5:以游泳分配一个柜子为例来说明sesson的用法

6:session对像的常用手法：

A:在一个页面里面创建一个session变量.

B:在另外一个页面里面去使用该session变量.

Session对像的方法

1)在Servlet中要使用Session，必须要使用HttpSession接口

2)Session对像是一个数据集合，也就是一个Session对像里面可以放很多session变量，

每一个Session变量都可以存放数据

3)if(request.getSession().getAttribute("IsLogin")!=null)

判断一个Session变量IsLogin是否存在，只有存在才可以继续读取里面的数据

3)得到一个IsLogin变量的值

if(request.getSession().getAttribute("IsLogin")!=null) 只有判断存在了，才可以读数据

request.getSession().getAttribute("IsLogin"):

4)request.getSession().setAttribute("IsLogin","Yes");设置session中IsLogin里面的值

什么是Cookies对像

cookies对像可以客户端长期保存信息。在客户端的计算机里面会有一个cookies文件夹，那里就是存放客户端的信息的。当客户端第一次去访问某个网站时，该网站就会在客户端的计算机里面偷偷地写入一个Cookies

下一次再访问该网站时，它就会读取你的计算机硬盘上cookies，并将新的信息保存在你的计算机上。

跟session不同的，cookies是把信息存储在客户端，不是在服务器里面。

操作cookies

1:得到所有的Cookies

Cookie[] cookies=request.getCookies();

2:得到指定Cookies的名字：

Cookie.getName()

3:得到指定Cookies的值

Cookie.getValue();

4:设置指定Cookies的值

cookie.setValue(值)

5:添加一个Cookies

cookie= new Cookie("Visted","1");

cookie.setMaxAge(10000);设置cookie的过期时间以钞为单位

response.addCookie(cookie);

4:修改指定cookie的值：注意修改之后也要重新添加，否则添加之后无效

cookie.setValue(值);

cookie.setMaxAge(1000);

response.addCookie(cookie);

应用举例：

1:证明正常情况下，一个网页或Servlet的数据是不能给另外一个网页或servelt共享

做两个servlet，一个servlet里面给一个session变量赋值，给一个私有变量赋值

在另外一个servlet里面去使用session变量与私有变量的值，看一下结果

2:有一个登录界面(login.html),一个主页面(index)，在主页面里面有二个超链接分别为

无限下载,新闻快递。用户必须要登录之后进入主页面，然后才能使用这二个超链接功 能，如果直接使用Index.html则会提示没有登录,然后转到登录界面 [注意使用session对像]

登录页面-------->转到servlet1,用来处理用户名是否正确，如果正确，转到主页面，否则转到 错误页面

主页面---------->转到servlet2,用来判断用户是否登录过，如果登录过则转到无限下载或新闻 快递页面,否则转到登录页面

3:统计某个客户端曾经访问某个网站的次数，和网站的所有访问次数

Cookie[] cookies=request.getCookies();

if (cookies!=null) //如果cookies不为空

{

for (i = 0; i < cookies.length; i++)

{

if (cookies[i].getName().equals("Visted"))

{

break;

}

}

if (i<cookies.length) //找到

{

cookie=cookies[i];

int n=Integer.parseInt(cookie.getValue());

cookie.setValue(Integer.toString(n+1));

cookie.setMaxAge(1000);

response.addCookie(cookie);

}

else

{

cookie= new Cookie("Visted","1");

cookie.setMaxAge(10000);

response.addCookie(cookie);

}

}

else //为空，一定要创建

{

cookie= new Cookie("Visted","1");

cookie.setMaxAge(10000);

response.addCookie(cookie);

}

out.println("你共访问网"+cookie.getValue());

4:在登录时，选择自动登录，下次在登录时直接进入主页面

创建一个Servlet，在里面判断是否有Cookies，如果有则直接进入主页面，

如果没有则首先进入登录页面

Serlvet上下文

1:Servlet中的Session可以让同一个客户端在不同网页之间进行共享

2:Servlet中的上下文可以让不同的客户端来共享一些数据，这些数据要存放在上下文中，

可以把上下文看成一个容器，里面存放的数据可以被所有客户端来共享

3:如何得到上下文的一个实例：

ServletContext context=this.this.getServletContext();

4:如何得到一个上下文指定键的值：

if (context.getAttribute("Pwd")!=null)

{

Pwd=context.getAttribute("Pwd").toString();

}

5:如何设置一个上下文的值：

if (context.getAttribute("Pwd")==null)

{

context.setAttribute("Pwd","\*\*\*");

}

举例：

1：通过示例，说明Servlet中的Session与上下文的不同

步骤:

1)做一个appser1，在它里面创建一个Session和一个上下文，

创建在appser2中输出Session和上下文

2)打开IE,先在同一个浏览器中先执行appser1，再执行appser2

3)打开另外一个IE直接执行appser2,看数据共享情况

2:通过上下文做一个聊天室

A:一个登录页面，提交到Dowithser中

B:一个Dowith的Serlvet用来把Dowithser中的用户名放入session中

C:一个框架集，含有上下两部分，下一部分包含一个talk.html页面，

上一个是一个topser的Servlet，用来显示所有发言

第四章 JavaMail和Servlet

发送邮件的工作原理：

1:客户端把邮件发送给-------->发信服务器---->发送到目的地

2:发信服务器是各大电子邮箱的服务器。一般在前面加上smtp。比如：

smtp.sina.com,smtp.163.com,smtp.126.com等

3:只要去申请一个免费的电子邮箱，就可以获得一个发信服务器

发送邮件必须需要两个包：

mail，activation包。这两个包可以在

jakarta-tomcat-4.1.31\common\lib下面找到。一定要复制到

web模块下面WebMod\WEB-INF\lib下面

发送邮件需要导入的包：

import javax.mail.\*;

import javax.mail.internet.\*;

import javax.activation.\*;

发送邮件的涉及到的类

A:Properties类：

用来指定发信服务器，和使用发信服务器时是否需要身份验证：

Properties props = new Properties();//创建一个Properties的实例

props.put("mail.smtp.host", "smtp.sina.com");//设置发信服务器

props.put("mail.smtp.auth", "true");//要经过身份验证

B:Session类，会话类。要发邮件必须要创建一个Session类

Session session = Session.getInstance(props);

C:MimeMessage类.用来指定发送邮件的标题，文本，等

MimeMessage message = new MimeMessage(session);//得到MimeMessage的实例

message.setFrom(new InternetAddress("chengyuqing110@sina.com"));//设置发件人

message.addRecipients(Message.RecipientType.TO,"chengyu-qing@163.com,chengyu-qing@tom.com"); //设置收件人，可以同时指定多个收件人

message.setSubject("主题");//设置邮件的主题

message.setText("内容文本");//设置邮件的内容文本

D:Transport类。用来把邮件进行实际的发送

Transport transport = session.getTransport("smtp");//得到Transport类的一个实例

transport.connect(host, username, password);//连接发信服务器

transport.sendMessage(message, message.getAllRecipients());//发送邮件

transport.close();//发送之后关闭

说明：

1:选择发信人邮箱一定要是发信服务器里面申请的邮箱

比如：

如果发信服务器-------->smtp.sina.com

则发信人邮箱一定要是sina的邮箱，否则会出现错误

发送带有附件的邮件

具备知识：

要发送附件，必须有Multipart类。这个类用来负责管理附件，

一个Multipart类同多个MimeBodyPart构成。每一个MimeBodyPart就代表一个附件

如果要发送附件，则"文本"也代表一个附件。文本是附件的第一个MimeBodyPart

String host = "smtp.sina.com";

String from = "chengyuqing110@sina.com";

String to = "chengyu-qing@163.com";

String username = "chengyuqing110";

String password = "12345678";

Properties props = new Properties();

props.put("mail.smtp.host", host);

props.put("mail.smtp.auth", "true");

Session session = Session.getInstance(props);

session.setDebug(true);

MimeMessage message = new MimeMessage(session);

message.setFrom(new InternetAddress(from));

message.addRecipients(Message.RecipientType.TO,to);

message.setSubject("这是试验");

//===设置附件的正文=======================================

MimeBodyPart bp1=new MimeBodyPart();

bp1.setText("正文");

//===============设置附件==================================

MimeBodyPart bp2=new MimeBodyPart();

bp2.setDataHandler(new DataHandler(new FileDataSource("c:\\过滤器.ppt")));

bp2.setFileName(MimeUtility.encodeText("c:\\过滤器.ppt")); //处理中文

//================================加入附件里面=======================

Multipart mp=new MimeMultipart();

mp.addBodyPart(bp1);

mp.addBodyPart(bp2);

message.setContent(mp);

//==================================================================

Transport transport = session.getTransport("smtp");

transport.connect(host, username, password);

transport.sendMessage(message, message.getAllRecipients());

transport.close();

//====================================================================

收邮件

String Uid =request.getParameter("txtUid");//邮件服务器的用户名

String Pwd =request.getParameter("txtPwd");//邮件服务器的密码

String Host =request.getParameter("txtHost");//邮件服务器的主机名，一般是pop.sina.com或pop.163.com

Properties props = new Properties();

Session Getsession=Session.getDefaultInstance(props);//建立session

Store store = Getsession.getStore("pop3");

store.connect(Host,Uid,Pwd);

Folder folder=store.getFolder("INBOX");

folder.open(Folder.READ\_ONLY);

Message[] message=folder.getMessages();

message[i].getSubject()//得到邮件的主题

message[i].getSentDate().toLocaleString()//得到发信日期

Multipart p=(Multipart)message[i].getContent(); //得到一个Multipart，去读出附件

p.getBodyPart(0).getContent().toString();得到邮件的文本

//===========================================================================

第五章 Jsp简介

Jsp概念:

1:什么是Jsp

Jsp(Java Server Page)是一种Java服务器端技术。

一个Jsp页面包括三个方面的内容

Html,JavaScript,Jsp程序片断

2:如何写Jsp脚本

把Jsp脚本写在<% %>里面

把Js脚本写在<script>代码</script>里面

3:可见Jsp实际上就是在Html语言中嵌入Jsp脚本

4:编写第一个Jsp应用程序

Jsp的执行过程：(\*\*\*\*)

浏览器--->请求-------->Web服务器---->找到Jsp页面---->编译

|

|

浏览器<-------响应<------Serverlet

从该执行过程不难看出Jsp在最后实际上要变成Serverlet之后才能发给客户端

Jsp与Servelet的区别:

1)Jsp是在Serverlet的基础上发展起来的

2)Serverlet一般用于处理没有界面的业务逻辑,如果处理有界面的程序，就显示比较笨拙

Serverlet用out.println一句一句的输出Html标记(界面),当整个网页非常多且复杂的时候

就会有很多的out.println来输出html标记

3)Jsp很好的解决了这个问题，对于Html标记可以直接书写，只有需要写程序代码时才以

<% %>的方式加入

4)Jsp把界面部分与代码部分分离，这样可以在开发网站时实现分工。美工只管界面部分，而

程序员只管代码部分

Jsp的语法：

A)Jsp的输出：out.println用来输出

1)输出字符串：out.println("字符串")

2)输出Java脚本：out.println("<script>alert('我是真的爱你');</script>");

3)输出Html标记：out.println("<a href=http://www.sina.com>新浪网</a>");

链接到QQ对话:out.println("<a target=blank href=http://wpa.qq.com/msgrd?v=1&uin=16663755&site=walmart&menu=yes>留言</a>");

B)表达式。<%=Java表达式%>。表达式后面不要加分号

例如：<%=Math.max(100,90)%>,<%=3+4\*5%>等

C)Scriptlet:

Scriptlet就是在<%.....%>里面嵌入Java代码。它在形式上与表达式的区别就是在

<%...%>里面没有"="，而且中间是一段Java代码，不是表达式，每条语句要以分号

结尾。举例说明

D)声明：<% ! Java变量,方法 %>

1):声明变量： <%! int a=3,b=4 %>

2)声明类：

<%!

public class stud

{

public String GetId()

{

return "W1";

}

}

%>

3)声明函数：

<%!

public int sum(int a,int b)

{

return a+b;

}

%>

说明：

1)声明变量即可以在<%! %>只能够声明，不能实现

比如：<%! int a=3,b=4,c; c=a+b;%>错误<%! int a=3,b=4,int c=a+b;%>正确

2)声明变量既可以<%! %>中声明，也可以在<%中声明%>。

<%! %>中声明的变量：servlet(Jsp最终要转换成Serverlet)的实例变量。

根据Servlet的原理，所以的客户端只共享一个Serlvet的实例，所以在此处定义的

变量可以其它的页面来共享，也就是具备有继承性

<% %>中声明的变量：是局部变量，只能够给本页面使用，不具备继承性

<%!

int i=0; //实例变量，有继承性

%>

<%

i++;

out.println(i);

%>

与的区别

<%

int i=0; //局部变量，无继承性

i++;

out.println(i);

%>

3)但是声明函数与类只能够在<%! %>中声明，而不能在<% %>

4)一般声明变量都在<% %>中声明，而声明函数与类都在<%!中声明%>

5)为了代码的可读性一般都把类用一个单独的Java文件来保存，而不会在一个

Jsp页面中去定义

E)Jsp指令:

1)Jsp指令控制对整个页面的处理。

2)Page指令：

1:导入Java包

格式：<%@ page import="aspproject.Student,java.util.\*"%>

说明：自定义的类一定要放在WEB-INF下面的classes下面,不过只要引用了一个类，

Jbulider会自动的创建一个classes文件夹，并放入相应的类

2:当本页面发生错误时，跳转到错误页面进行处理

格式：

源页面:<%@page errorPage="Error.jsp" isErrorPage="false" %>

错误页面:<%@ page isErrorPage="true" %>

说明：错误是当出现异常，但是没有捕获时,如果捕获了，则不会发生页面跳转

3)include指令：

1:该指令用于把一个Jsp页面包含到另外一个Jsp页面中

2:格式：<%@ include file="HeadPic.jsp" %>

说明：嵌入一个页面是为了重用这个页面，而不必在每一个页面上写一些相同的代码。

例如:在很多的Jsp页面中都要在顶部显示同样的图片，这时就不必在每个页面

都去包含这些图片，而是单独写一个文件来完成此任务，然后在其它的Jsp页面中

利用include包含即可

作业：

1)编写一个Java类,用来描述学生。包含学生的学号，姓名，性别。声明三个实例，并加到

一个ArraryList,并把结果显示在表格中

2)编写一个页面，让用户输入两个数和一个操作符，在另外一个页面显示计算的结果

3)做一个民意调查页面，来调查"在校大学生谈恋爱",好，不好，不好说三个选项

一个页面：显示界面用来让用户进行投票，另外一个页面显示投票的结果

jtds连接Sql server

4)使用Jsp实现用户登录

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class.forName("net.sourceforge.jtds.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:jtds:sqlserver://chen:1433;DatabaseName=jsp", "sa", "");

java.sql.PreparedStatement ps = conn.prepareStatement(Sql);

java.sql.ResultSet rs = ps.executeQuery();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

第六章 Jsp隐式对像

Jsp的隐式对像有以下几个：

A:out对像：主要用来进行输出

1)输出字符串：out.println("字符串")

2)输出Java脚本：out.println("<script>alert('我是真的爱你');</script>");

3)输出Html标记：out.println("<a href=http://www.sina.com>新浪网</a>");

B:request对像：用来获取客户端的请求

1)request.getParameter("名字"):获取用户提交过来的请求

2)request.getParameterValues("名字"):获取单选框的值

3)request.getRemoteAddr():得到发送请求的主机名

4)request.getRequestURI():得到发送请求的Url

C:Response对像：用来向客户端输出响应结果

1)response.setContentType("编码")

设置向客户端输出的文本的编码，一般设为"text/html; charset=GBK"

response.setContentType("text/html; charset=GBK");

如果不设置，可能会出现乱码

2)response.setHeader("refresh","1");使页面进行刷新

扩展：response.setHeader("refresh","5;URL=新页面");

过5钞钟跳转到新的页面

3):response.sendRedirect("Index.html");

使页面进行跳转

D:Session对像

E:application对像。相当于Serlvet上下文

F:PageContext对像。只能够在同一个页面里面使用。一般很少用

其中PageContext,request,session,application四种对像都可以通过

setAttribute()方法存储值，通过getAttribute()得到值

但是四种对像的范围不一样

PageContext(也被称为Page对像):一个页面中

Request:一次请求中

session:一次会话中

application:一个应用程序中

范围大小是：application>session>request>pageContext

举例：

1:举例显示会话计数、应用程序计数、页面计数的区别

2:做一个在线购物的例子

第七章 JavaBean和Jsp的标准动作

什么是JavaBean:

1)是一个可重复使用的软件组件

2)JavaBean实际上就是一个Java类,这个Java类可以重复使用

3)JavaBean在网站开发中一般用来封装业务层和数据库操作

4)虽然我们可以像开发一个普通Java类一样去开发一个JavaBean,但是JavaBean有它自己的规则

5)JavaBean的规则：

A:是一个公有类

B:具有不带任何参数的构造函数

C:具有getXxx(),setXxx()，用来读取或写入Bean的属性(注意get与set一定要小写\*\*\*\*\*\*\*)

6)JavaBean与Java类之间的区别在于JavaBean有特定的规则，并且不需要继承自特定的类。而Java类一定要继承自特定的类。默认为Object类

如何来编写JavaBean [选择File-->new-->general-->JavaBean]

package demo;

public class BeanSum

{

private int num1;

private int num2;

public BeanSum()

{

}

public void setnum1(int num1)

{

this.num1=num1;

}

public void setnum2(int num2)

{

this.num2=num2;

}

public int getnum1()

{

return this.num1;

}

public int getnum2()

{

return this.num2;

}

public int sum()

{

return this.num1+this.num2;

}

}

说明：

1:在JavaBean中除了有set与get方法之外，还可以其它的方法

2:在JavaBean的方法的变量大小写要符合Java的命名规则：

如果只有一个单词，则全部小写,如果一个变量名或方法由多个单词构成

则除第一个单词小写外，其它的单词第一个字母一律大写

比如：shopName 不能写成 ShopName 或shopname

setShopName 不能写成 SetShopName或setShopName

如何使用JavaBean

A:嵌入 java 代码方式

<%@ page import="demo.BeanSum" %>

下边就可以像在 java 语言中那样用了：

<%

BeanSum demo=new TestBean();

demo.num1=10; //取属性是一定要取小写的num1,尽管定义时写的是大写]

demo.num2=20;

out.println(demo.sum());

%>

这一种方式也就完全把JavaBean当成了普通对像

B:使用 jsp 标记符方式

<jsp:useBean id="demo" class="demo.BeanSum" scope="session" /> 相当于 BeanSum demo=new TestBean();

<jsp:setProperty name="besum" property="num1" value="10" />相当于demo.num1=10;

<jsp:setProperty name="besum" property="num2" value="20"/> 相当于demo.num2=20;

<%out.println(demo.sum())%> 注意这一个代码要分开，因为这段代码既不是实例化JavaBean又不是属性。

C:感觉用"jsp 标记符方式"比起"嵌入 java 代码方式"困难，那是因为没有把JavaBean与表单元素结合起来

D:通过<jsp:setProperty name="demo" property = " \* " /> 能够自动地把上一个页面提交过来的同名的表单元素的值赋给所有的属性

E:举例：

源页面：<input name="num1" type="text" id="num1" size="12"> 名字为num1

<input name="num2" type="text" id="num2" size="12"> 名字为num2

目的页面：

<jsp:useBean id="demo" class="demo.BeanSum" />

<jsp:setProperty name="demo" property = " \* " /> 会自动赋值

<%out.println(demo.sum())%>

F:在通过jsp标记来使用JavaBean时，有一个关键字scope用来指定这个JavaBean的活动范围

Page:只能在其页面时使用，当页面刷新时，就会将其销毁，这是默认的情况

session:一直存在于会话中，也就是说可以被同个客户端的不同页面之间共享

application:能够给所有的客户端来共享

G:也就是说不同的页面可以通过 Bean 交互, 而你在一个页面中的普通类,

就没有这个功能。

Jsp的标准动作：

1:<jsp:setProperty>用来给JavaBean设置值

2:<jsp:getProperty>用来得到JavaBean属性的值

3:<jsp:forwad page="url">用来把页面跳转到page所指的页面。

与response.sendRedirect()的区别

1)response.sendRedirect()是在客户端进行跳转，地址栏中会出现新的页面的地址<jsp:forwad page="url">是在服务器端跳转，地址栏中依然是以前旧的地址。被称为跳转

2)<jsp:forwad page="url">是传递请求，所以这个请求到下一个页面依然可以使用，

而response.sendRedirect()是在跳转，请求在下一个页面不在可以使用

可以使用request.setAttribute来设置值。被称为"转发"

举例：让用户输入用户名和密码如果用户名正确则转到首页，如果错误则转到错误页面

4:<jsp:include page="url" flush="true"/>:用于把其它的页面合并到本页面

与<%@ include %>的区别：

A:jsp:include是动作，而<%@ include %>是指令

B:jsp:include是在运行时进行合并，而<%@ include %>在编译时合并

Jsp:inclulde----->1.jsp--->编译-->servlet |

|--->合并

2.jsp--->编译-->servlet |

<%@include %>---->1.jsp-----|

|--->合并--->编译

2.jsp-----|

举例：做一个JavaBean让用户输入商品名称，商品数量，折扣。

然后根据商品名称求出单价,根据数量与折扣求出应付款

第七章 Jsp表达式语言

什么是jsp表达式语言:

Jsp用于在网页上显示动态的内容。通常需要在Jsp页面中嵌入Java脚本以

完成复杂的功能。但是大量的Java脚本使得Jsp页面难以维护。Jsp表达式语言(Expression language)

可用于在Jsp页面上生成动态的内容，并代替jsp脚本元素，以一种更简洁方式来

显示内容。Jsp表达式语言是从Jsp2.0规范开始支持的

什么要使用Jsp表达式语言：可以更加简洁的实现下面的三个功能

1)简洁访问存储对像(PageContext,Request,Session,Application)里面的值

2)利用EL简洁访问从表单元素取过来的值取替request.getParameter方法

3)JavaBean属性的简略记法

Jsp表达式语言(EL)的格式：

${表达式}

如何利用EL简洁访问存储对像的值

A:先对比。

页面1.jsp中

<%

session.setAttribute("Uid","chen");

%>

<jsp:forward page="jsp2.jsp"/>

页面2.jsp接受来自于1.jsp中的值

<%

String Uid="";

if (session.getAttribute("Uid")!=null)

{

Uid=session.getAttribute("Uid").toString();

}

%>

<h1>接受的值为<%=Uid%>

利用EL进行改进2.jsp页面

<h1>接受的值为${Uid}</h1>

结论：

1:通过EL访问非常简洁

2:如果值不存在，EL返回值不是NUll而是空字符串

3:如果四种对像都有相同名字的属性，

则在能够访问的范围(注意)内查找的顺序为PageContext,Request,Session,Application

找到第一个为止。也就是说如果此时能够访问PageContext,Request,Session,Application。则EL的值为PageContext的值。

如果能够访问的范围为Request,Session,Application则EL的值为Request等等

4:可以通过${session|Request|Application|Scope.Uid}的形式来决定要取到什么范围的值

举例：登录的例子，如果登录正确，则显示欢迎信息

如何利用EL简洁访问从表单元素取过来的值用以取替request.getParameter方法

格式：${param.表单元素值} 注意表单元素值必须要有name属性

1.jsp1页面：输入学号，语文，数学，化学

2.jsp2页面：显示该学生的总分与平均分

改进：把显示的结果也放入jsp1页面中,能够达到保留表单元素值的目的

学号:<input type="text" name="txtId" value="${param.txtId}"/><br>

如何利用EL对JavaBean属性进行简略记法

格式1：${BeanName.BeanProperty}

格式2：${BeanName["BeanProperty"]

jsp1页面：输入商品名,数量，单价，实付款

jsp2页面：通过一个ShopBean求出实付款和找零

不使用EL表达式:

商品名：<jsp:getProperty name="ShopBean" property="shopname"/><br>

数量： <jsp:getProperty name="ShopBean" property="amount"/><br>

使用EL表达式：

商品名：${ShopBean.shopname}<br>

数量： ${ShopBean.amount}<br>

第八章 标签开发

为什么要自定义标签：

虽然Jsp里面付带有很多的标签，这是这些标签的功能和个数毕竟有限，为了得到功能更强大

并满足用户需要的标签，就要自己定义标签。自定义标签有两大优点：

A:实现代码重用 B:可以把数据的表示与实现分离

如何来自定义标签：

A:编写标签处理程序，处理程序就是一普通的Java类，这个类必须要继承自BodyTagSupport,

并实现其中的重要的两个方法

public class TagTest extends BodyTagSupport

{

public int doStartTag() throws JspException

{

}

public int doEndTag() throws JspException

{

}

}

B:编写标签库的描述文件(\*.tld)注意一定要放在WEB-INF文件夹下面

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN" "http://java.sun.com/j2ee/dtds/web-jsptaglibrary\_1\_2.dtd">

<taglib>

<tlib-version>1.0</tlib-version> //库的版本,随便写

<jsp-version>1.0</jsp-version> //jsp的版本,随便写

<short-name>tagtest</short-name> //标签的缩写名

<tag>

<name>tagtest</name> //标签唯一名称,该名字决定了使用标签的名字

<tag-class>tagapp.TagTest</tag-class> //标签处理器类名

<body-content>jsp</body-content> //内容类型。一般为Jsp,表示可以是jsp中的任何内容

</tag>

</taglib>

C:使用标签：

1:选导入标签库：<%@ taglib uri="WEB-INF/TagTest.tld" prefix="chen" %>

2:通过<前缀:标签名>标签体</前缀:标签名>

编写标签库要注意的几项：

1:javax.servlet.jsp.tagext.\*。提供创建标签库所需要的接口与类

2:在doStartTag()里面不能访问BodyContent主体内容，因为此时主体内容还没有产生

3:如果在doStartTag()里面返回值为SKIP\_BODY，则表示跳过了主体内容，则在doEndTag

里面也不能访问BodyContent主体内容,否则会出现异常

4:所以如果要访问主体内容的操作，必须省略doStartTag()或在doStartTag()方法中

返回EVAL\_BODY\_BUFFERED表示不跳过主体内容，然后在doEndTag()方法中去处理

5:doEndTag()方法可以返回EVAL\_Page(表示执行标签后面的页面代码)或

SKIP\_PAGE(表示执行完标签之后就停止页面内容执行)一般返回EVAL\_Page

6:要在标签中进行输出，可以通过 pageContext类的一个getOut()方法：

JspWriter out = pageContext.getOut();

out.println("字符串")进行输出

7:要在标签中访问Jsp中的隐式对像：

HttpServletResponse response=(HttpServletResponse)pageContext.getResponse();

HttpServletRequest request=(HttpServletRequest)pageContext.getRequest();

pageContext.getSession()

8:要在标签中访问Jsp标签中的标签体的内容(一般在doEndTag中)：

this.bodyContent.getString();

9:可以被Script使用的标签：

定义了id和type属性的标签可以被标签后面的Scriptlet使用。

<mytag：connection id = “oraDB” type = “DataSource” name = “Oracle”>

<%oraDB.getConnection(); %>

标签的分类：

A:简单标签：

格式：<前缀:标签名/>

特点：没有属性也没有标签体

定义方法：只要实现doStartTag()方法，并返回SKIP\_BODY

举例：

1)自定义一个标签，能够自动产生网页的Log画面

2)自定义一个标签，能够自动产生一个登录窗体

3)自定义一个标签，能够产生一个关闭超连接

B:有标签体的标签

格式：<前缀:标签名>内容</前缀:标签名>

特点：没有属性，但是有标签体

定义方法：

一般有标签体的标签，都是希望标签处理器能够对标签体进行操作，如上所述由于

doStartTag本身不能对标签体进行引用,所以要对标签体进行操作只需要实现doEndTag()，并返回EVAL\_Page

举例：

1)自定义一个标签,用来把标签的体中的英文字母由小写转换成为大写

2)自定义一个标签,以标签体作为消息提示来弹出一个对话框

C:有属性的标签(\*\*\*\*)

格式：<前缀:标签名 属性名1=值1 属性名2=值2/>

特别：有属性，但是没有标签体

定义方法：

1)首先定义属性，通过set与get定义属性

2)由于没有标签体，所以只要实现doStartTag()方法，并返回SKIP\_BODY

3)书写Tld标签描述文件。

<taglib>

<tlib-version>1.0</tlib-version> //库的版本,随便写

<jsp-version>1.0</jsp-version> //jsp的版本,随便写

<short-name>tagtest</short-name> //标签的缩写名

<tag>

<attribute>

<name>num</name> //属性名

<required>true</required>//是否必须

<rtexprvalue>true</rtexprvalue>//是否支持jsp表达式来为属性赋值，一般为true

</attribute>

<attribute>

<name>num</name>

<required>true</required>

<rtexprvalue>true</rtexprvalue>

</attribute>

..................

<name>tagtest</name> //标签唯一名称,该名字决定了使用标签的名字

<tag-class>tagapp.TagTest</tag-class> //标签处理器类名

<body-content>jsp</body-content> //内容类型。一般为Jsp,表示可以是jsp中的任何内容

</tag>

</taglib>

举例:

1)编写一个标签，有两个属性num1,与num2求出其中的最大值

2)编写一个属性，传入一条不返回结果的sql语句,这个标签会自动执行对应的sql语句

3)编写一个标签，可以实现数据信息少量显示。主要以下属性：

title,moreUrl,detailUrl,keyFiled,textFiled,sql;

4)编写一个标签，传入商品类别，标签会查询出所有该类别的商品信息.

解决乱码：

String Name=new String(request.getParameter("txtName").getBytes("iso-8859-1"));

out.print(Name);

修改web应用程序的默认页

把TomCat下面的\conf\web.xml下面最后面的如下部分，复制到Web应用程序

web.xml下面即可

<welcome-file-list>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

在web.xml中设置参数，在jsp页面读出参数

步骤

1：在web.xml中加入以下代码

<context-param>

<param-name>uid</param-name>

<param-value>chen</param-value>

</context-param>

<context-param>

<param-name>pwd</param-name>

<param-value>123</param-value>

</context-param>

2:

//jsp

application.getInitParameter("ParameterName");

//servlet

request.getSession().getServletContext().getInitParameter("ParameterName");