[概述 1](#_Toc9381)

[服务器 1](#_Toc22682)

[实现步骤（myeclipse） 1](#_Toc4188)

[请求方式 2](#_Toc7521)

[Servlet跳转 3](#_Toc3043)

[内部跳转 3](#_Toc10327)

[重定向 3](#_Toc24834)

[Response应答内容 3](#_Toc31517)

[HttpSession会话 4](#_Toc21985)

[Cookie 5](#_Toc10763)

[JSP 5](#_Toc8482)

[JSTL 6](#_Toc15946)

[EL表达式 7](#_Toc2036)

[宿管小项目 7](#_Toc3650)

概述

JAVA EE: 企业级开发Web项目 B/S架构,HttpServletRequest请求HttpServletResponse响应

浏览器

Tomcat(S)

URL(统一资源定位符) URI(统一资源标识符) URN

http://服务器地址(域名或ID):端口/应用名称/资源路径

eg <http://localhost:8080>(服务器) <http://localhost:8080/app>(应用端)

请求

响应

客户端 服务器Tomcat、glasslish

服务器

Tomcat服务器下载：<http://tomcat.apache.org/>

加载服务器：window—preference—myeclipse--servers—tomcat—选择版本加载路径即可

运行服务器：servers窗口—选中服务器-右击添加项目add Deployment—运行:run server

开发工具：一Eclipse+Tomcat 二myeclipse 三netbeans+glasslish 四intelliji+tomcat

新建完servlet若提示servlet相关的包无法引入，则Tomcat目录\apache-tomcat-…\lib\找servlet-api.jar复制到项目的web-INF/lib文件内即可

中文乱码

输入中文乱码

①检查下 form 的action 改成post

②servlet中 加 request.setCharacterEncoding("utf-8");

响应中文乱码 response.setCharacterEncoding("utf-8");或浏览器字符编码问题设置为utf-8

实现步骤（myeclipse）

创建Servlet：编写类继承javax. servlet.http.Http Servlet类，覆盖其中的service方法

public class HServlet extends HttpServlet {

protected void service(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

System.out.println("访问成功"); }}

@WebServlet("/Hello.do")

public class HelloServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException { }

protected void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

req.setCharacterEncoding("utf-8"); String user = req.getParameter("user");

resp.setContentType("text/html");// 设置应答返回内容的类型

resp.setCharacterEncoding("utf-8"); PrintWriter pp = resp.getWriter();

pp.write(user + "你好呀！"); pp.append("<p style='color:blue;'>登录成功</p>");}}

注册Servlet：在web.xml编写如下内容进行注册Servlet

<servlet> <servlet-name>Servlet名称(任意)</servlet-name>

<servlet-class>编写Servlet的全限定名称</servlet-class></servlet>

<servlet-mapping> <servlet-name>Servlet名称(任意)</servlet-name>

<url-pattern>编写Servlet可处理的URL模式</url-pattern></servlet-mapping>

url-pattern(注:要么/开头,要么\*.开头,两者互斥)

以/开头则后面不能出现\*. /a即等于应用路径加/a /a.do即等于应用路径加/a.do

以\*.开头后面不能出现/ eg \*.do

请求方式

对应触发servlet的doget或dopost方法

Get方式发送的参数跟在URL后面，有可能乱码，没有请求体

1 浏览器输入地址回车后是get方式提交

2 所有的超链接都是get方式提交

3 form的method=”get”或不写 也是get方式提交

Post 方式发送的参数放入请求体中

<body>

<form action="Hello.do" method="get">// method="get"

账号： <input type="text" name="user"> <button type="submit">登录</button>

</form>

<a href="Hello.do">调用servlet</a>

</body>

public class HelloServlet extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException { resp.setCharacterEncoding("utf-8");

resp.getWriter().append("调用了get方法 ");//直接运行servlet}

protected void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException { resp.setCharacterEncoding("utf-8");

resp.getWriter().append("调用了post方法 ");}}

Servlet跳转

跳转到html(无法接收数据)、jsp和servlet

内部跳转

(转发请求，可传递数据，其中地址必须以/开头)

req.getRequestDispatcher("/ok.html").forward(req, resp);

req.getRequestDispatcher("/req.jsp").forward(req, resp);

req.getRequestDispatcher("/b.do").forward(req, resp);

<body>

<form action="Hello.do" method="post">

账号： <input type="text" name="user">

<button type="submit">登录</button> </form>

</body>

public class HelloServlet extends HttpServlet {@WebServlet("/Hello.do")

protected void doPost(HttpServletRequest req, HttpServletResponse resp)throws ServletException, IOException {

req.setCharacterEncoding("utf-8");

req.setAttribute("number", 6666);//手动添加数据

req.getRequestDispatcher("/Hello.jsp").forward(req, resp);

}}

<body> 欢迎光临！<br>

<%String name = request.getParameter("user");

Integer number= (Integer) request.getAttribute("number");%>

界面输入的用户名是：<%=name%>

后台手动设置的数据是：<%=number%></body>

重定向

1 它不能带数据跳转，重定向到内部地址须写当前应用名称+跳转地址

resp.sendRedirect(req.getContextPath()+"/fail.html");

2 重定向到外部地址须写绝对路径

resp.sendRedirect("https://www.baidu.com");

选择标准: 是否需要改变地址栏上面的地址

Response应答内容

HTML、图片、画布生成验证码

response.setContentType("image/jpeg");

String imgPath = "E:\\文档\\2Java基础\\Java EE\\1.jpg";

FileInputStream fis = new FileInputStream(imgPath);

int size = fis.available(); byte[] buff = new byte[size];

fis.read(buff);fis.close();// 图片存入buff

OutputStream os = response.getOutputStream();

os.write(buff);os.flush();os.close();// 图片写入页面

<input id="veryCode" name="veryCode" type="text"><br>

<img src="Img" height="60px" width="220px"><br>

public class Img extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

Random r = new Random();// 随机数

int height = 30;int width = 120;// 验证码区域宽和高

String base = "0123456789abcdefghijklmnopqrstuvwxyz";

BufferedImage img = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);//图片类

Graphics2D g = (Graphics2D) img.getGraphics();

g.setColor(Color.LIGHT\_GRAY);// 设置画笔颜色

g.fillRect(0, 0, width - 1, height - 1);// 绘制背景矩形

g.setColor(Color.BLUE);

g.drawRect(0, 0, width - 1, height - 1);// 边框

// 随机产生160条干扰线，使图象中的认证码不易被其它程序探测到。

g.setColor(Color.BLACK);

for (int i = 0; i < 10; i++) {

int x = r.nextInt(width); int y = r.nextInt(height);

int xl = r.nextInt(12); int yl = r.nextInt(12);

g.drawLine(x, y, x + xl, y + yl); }

for (int i = 0; i < 4; i++) {// 0-34到 随机产生验证码

char c = base.charAt(r.nextInt(base.length() - 1));

// 用随机产生的颜色将验证码绘制到图像中。

g.setColor(new Color(r.nextInt(255), r.nextInt(255), r.nextInt(255)));

g.drawString(c + "", i \* 30 + 5, 25);}

ImageIO.write(img, "jpg", response.getOutputStream());}}

<%String veryCode = request.getParameter("veryCode");%>

界面输入的验证码是：<%=veryCode%><br>

HttpSession会话

1创建session，session可传值、跨页面、会过期

<body>//login.html

<form action="Login" method="post">

账号： <input type="text" name="user"><br>

<button type="submit">登录</button>

</form>

</body>//Login.servlet

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

String name = request.getParameter("user");// 1接收界面上输入的数据

//创建会话,若某时段内存在session，就获取session，否则就创建session

HttpSession s1=request.getSession();

s1.setAttribute("user", name);//将用户名存入session

response.sendRedirect("loginResult.jsp");// 重定向 }

<body>//loginResult.jsp

<% HttpSession ss = request.getSession();%>//获取session

用户名：<%=ss.getAttribute("user")%>

</body>

2使用session，购物车

Cookie

概念：服务器端发送到客户端的一些小型文件，时间可设置，用途：记住密码，保留重要信息在客户端

<body>//HTML

<form action="Cookies">

user:<input type="text" name="user"><br>

pwd:<input type="password" name="pwd"><br>

<input type="checkbox" name="check" value="1">记住密码<br>

<input type="submit" value="提交">

</form>

</body>setcookie

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

String name = request.getParameter("user");String pwd = request.getParameter("pwd");

String check = request.getParameter("check"); String msg = "";

response.setContentType("text/html"); response.setCharacterEncoding("utf-8");

if (check != null) {// 勾选了记住密码

Cookie c1 = new Cookie("user", name);//cookie记录用户名、密码

c1.setMaxAge(60 \* 60 \* 24 \* 7);// 设置cookie最大寿命，单位是秒

//c1.setValue("修改之后的值是");

Cookie c2 = new Cookie("pwd", pwd);// 将cookie响应到客户端

response.addCookie(c1);response.addCookie(c2); msg += "cookie保存成功";} response.getWriter().print(msg + "<br>login success"); }

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException, IOException {

Cookie[] cc = request.getCookies(); String msg = "";//getcookie

for (int i = 0; i < cc.length; i++) {

if ("user".equals(cc[i].getName())) {

String user = cc[i].getValue();msg += "cookie名：user，cookie值" + user + "<br>";}

if ("pwd".equals(cc[i].getName())) {

String pwd = cc[i].getValue();msg += "cookie名：pwd，cookie值" + pwd; }

} response.setCharacterEncoding("utf-8"); response.getWriter().append(msg);}

JSP

Java服务器页面，原理：运行时会先编译成一个Servlet，再执行，

JSP的组成

1 指令 <%@ xxx %> 每个指令 都有不同的属性来指定

①page指令

language="java" 说明jsp页面上脚本的语言

contentType="text/html; charset=UTF-8" 界面的文档类型 以及字符集

pageEncoding="UTF-8" 界面文件本身的字符集

import 引入java类的全路径

②taglib 指令 引入其他标签库

<%@ taglib uri="别的标签库的路径" prefix="前缀"%>

③include指令 引入其他页面

<%@ include file="second.jsp" %>

2 脚本 jsp中脚本和html 分别处理的

①<% %> 这里的代码生成到Service方法中

②<%= %> 相当于response.getWriter.print(等号后面的内容); 用于输出一些数据

③<%! %> 这里的代码生成到类成员 全局变量

3 隐式对象 jsp中可以直接使用 而不需要实例化一些对象

request HttpServletRequest

response HttpServletResponse

session HttpSession

out PrintWriter 打印数据到界面

application Servlet上下文 只要服务器不重启就不会消失 （计数器）

config

page 相当于本页面this

exception

PageContext 获取到页面的上下文

<%

String user = request.getParameter("user");

session.setAttribute("user", user);

if (application.getAttribute("time") == null) {// 变量time若不存在即第一访问时

application.setAttribute("time", 1);

} else {

Integer time = (Integer) application.getAttribute("time");

time = time + 1;

application.setAttribute("time", time);

}

pageContext.getRequest(); pageContext.getSession();

pageContext.getOut(); pageContext.getPage();

out.write(pageContext.getServletContext().getContextPath());

%>

<p> 用户名是：<%=user%></p>

<p> 恭喜你成为本网站全球第<%=application.getAttribute("time")%>位访问者</p>

JSTL

JSP标准标签库，目的是减少JSP中脚本的使用，包含核心库、格式库和XML库

使用JSTL

1使用taglib命令引入标签库

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

2 常用jstl标签 权限（page、request、session等）

<c:set var="名" value="值" scope="权限"></c:set>设置变量

<c:out value="${i}"></c:out>用于打印数据

<c:if test="条件"></c:if>分支语句

<c:choose>多分支

<c:when test="条件1"></c:when>

<c:when test="条件2"></c:when>

<c:otherwise></c:otherwise>//否则....

</c:choose>

<a href="<c:url value="www.baidu.com"/>"></a><!-- 链接 -->

<c:redirect url="http://www.baidu.com"></c:redirect>执行这就跳

<c:forEach var="名" begin="始值" end="终值" step="增量">${i}

</c:forEach>可遍历集合等list、map

<% String s[]={"aly","bob","catc"};

session.setAttribute("array",s); %>

<c:forEach items="${array}" var="a" > ${a} </c:forEach>

EL表达式

${} 用来显示或输出某变量或表达式的值

1 ${a}表或a的值

2 运算：加减乘除、求模、与或非、比较运算等

<c:set var="a" value="12"></c:set>

a=${a}<br> a+100=${a+100}<br> 与：${a>3&&a<90}

3 EL表达式的内置对象，它可从JSP的内置对象中获取变量

requestScope ${requestScope.xxxx} 获取request. setAttribute()方法设置的值

responseScope ${ responseScope.xxxx} 获取response. setAttribute()方法设置的值

sessionScope ${sessionScope.xxxx} 获取session. setAttribute()方法设置的值

applicationScope ${applicationScope.xxxxx}获取application. setAttribute()方法设置的值

Param 用于获取form中提交的数据以及超链接中传递的数据

Cookie ${cookie.USER.value}

Header 获取头部的信息

<%session.setAttribute("abc", 100); %>

<form action="EL.jsp">

<input type="text" name="user"> <input type="submit" value="提交">

</form>

abc=${sessionScope.abc} 用户名=${param.user}<br> ${header["user-agent"]}

宿管小项目

1 数据库设计（两张表students和room）

2 搭建环境，添加架包

搭建代码架构（代码分层）

Dao层 数据库操作，专门执行sql语句

logic层 调用Dao层

3 功能实现(对宿舍、用户等对象的增删改)

登录

List.servlet--->list界面

删除

①list界面上 删除超链接的href 写删除servlet的路径，并传递删除的条件

②dao 添加方法 执行delete语句

③servlet

a.接收超链接传过来的删除条件

b.调用dao执行delete语句

c.根据删除结果跳转到list.servlet重新查询数据，

并显示list.jsp

修改更新信息

①list界面---传一个id--->toUpdateServlet

②toUpdateServlet

a. 获取id

b. 根据id去数据查询需要被修改的这条数据

c. 跳转到 更新界面 让用户修改数据

③update.jsp 用户输入数据，点提交 --->updateServlet

④jsp--->dao-->servlet

a.获取界面上输入的数据

b.调用dao执行update语句

c.跳转回 list.servlet

新增

①add.jsp 用户输入数据 点击 提交 addServlet

② 接收数据，执行insert 语句 跳转回list.servlet

实现分页 select \*from 表名 limit a,b

原理:取结果集从索引a开始共b条数据 a=(currentPage-1)\*b,b为pagesize

当前页:currentPage

每页要显示的数据:pagesize

总的数据行数:totalcount,需执行select count查询而得

总的页数:totalpage=totalcount/pagesize,若除尽则为商，若除不尽则为商数加一

Select \* from room limit (currentPage-1)\*5,5

假设21条数据，每页显示五条

页码 索引

1 0-4

2 5-9

3 10-14

4 15-19

5 20

新增类处理分页PageUtil类

修改RoomDao的listRoom()方法 改为listRoomByPage(PageUtil pu)

修改页面显示根据totalpage即总页数去显示页数，前提需知道totalcount，所以先查询数据条数即在dao中添加方法查询总数据的条数

单击页码，实现翻页

　javax.servlet包加载不了怎么办？

　　右键点击你的工程，弹出属性设置列表：Build Path->Add Libraries...，弹出添加类库对话框，选择Server Runtime，点击Next按钮，选择web服务器(前提是配置了Web服务器，例如Tomcat)，点击Finish即可。