今日内容

# HttpServletResponse对象

作用：封装数据响应到浏览器 数据 字节流 字符路 转发页面

1 向页面响应字符流

|  |
| --- |
| response.getWriter().write(" <form action=''>") |

2向页面响应字节流

|  |
| --- |
| //响应字节流  //response.getOutputStream().write(b); 输出流 输入流  //获取输入流  InputStream in = this.getServletContext().getResourceAsStream("b.jpg");  ServletOutputStream out = response.getOutputStream();  //响应之前 设置响应的头信息 告诉浏览街 文件解析的格式  response.setContentType("image/jpg");  //定义一个字节数组  byte[] bs = new byte[1024];  //定义一个长度  int len = 0 ;  //读取输入流  while((len=in.read(bs))!=-1){  //向页面响应  out.write(bs, 0, len);  }  in.close();  } |

重定向

|  |
| --- |
| /\*//设置状态码  response.setStatus(302);  //再设置一个响应头 location  response.setHeader("location", "/WEB10/send.html");\*/  //向request域中存数据  request.setAttribute("name", "美美");  //重定向 实现页面的跳转  response.sendRedirect("/WEB10/Response4");//地址栏发生改变request域中的数据不会传递 |

# 完成用户注册案例（三层架构设计）

1 注册页面 表单和表对象

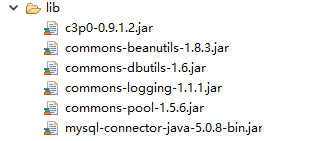
2 注册的servlet 接收参数 调用dao（操作数据库的代码）传参数

3 dao 接收servlet的参数 执行 insert 方法 返回值

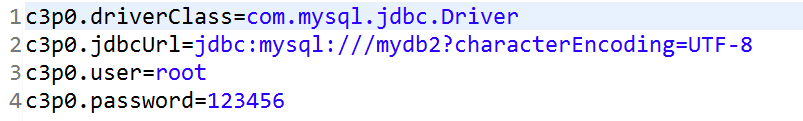
4 servlet 拿到返回值 进行判断 如果注册成功 跳转到登录页面 如果失败跳转到注册页面

【代码实现】

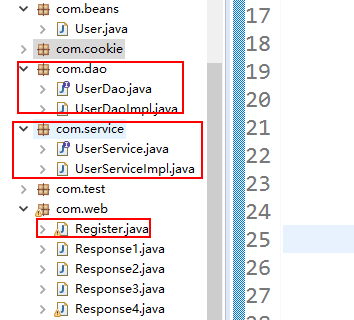
1 导入数据库操作的所有jar包 和 工具包



2 编写c3p0 的配置文件



3 代码分层



Dao层

|  |
| --- |
| public class UserDaoImpl implements UserDao{  @Override  public int regiter(User user) throws SQLException {  //执行sql语句  ComboPooledDataSource dataSource = new ComboPooledDataSource();  QueryRunner runner = new QueryRunner(dataSource);  String sql = "insert into user1 values(null,?,?,?,?,?,?,null)";  int i = runner.update(sql, user.getUsername(),user.getPassword(),user.getGender(),user.getEmail(),user.getTelephone());  return i;  }  } |

Service层

|  |
| --- |
| public class UserServiceImpl implements UserService{  UserDao dao = new UserDaoImpl();  @Override  public int regiter(User user) throws SQLException {  // TODO Auto-generated method stub  return dao.regiter(user);  }  } |

Web层

|  |
| --- |
| public class Register extends HttpServlet {  private static final long *serialVersionUID* = 1L;  UserService userService = new UserServiceImpl();  protected void service(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("UTF-8");  response.setCharacterEncoding("UTF-8");  response.setContentType("text/html; charset=UTF-8");  try {  User user = new User();  //接收页面的参数  Map<String, String[]> map = request.getParameterMap();  //使用工具类封装数据  BeanUtils.*populate*(user, map);  //将user传递 ----> service  //调用service 的方法  int i = userService.regiter(user);  if(i==1){//注册成功  response.sendRedirect("/WEB10/login.html");  }else{  response.sendRedirect("/WEB10/regiter.html");  }  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  } |

# 会话技术的概述



为了存储用户浏览时产生的一些必要的数据信息。

一个存储在浏览器端

一个存储在服务器端

# Cookie简介

|  |
| --- |
| Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management.  A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number. Some Web browsers have bugs in how they handle the optional attributes, so use them sparingly to improve the interoperability of your servlets. |

【创建cookie】

|  |
| --- |
| //创建cookie  Cookie cookie = new Cookie("name","lisi");  //返回给浏览器  response.addCookie(cookie); |

【获取cookies】

|  |
| --- |
| // 获取cookie的数据  Cookie[] cookies = request.getCookies();  //如果cookie部位空遍历cookie  if (null != cookies) {  for (Cookie cookie : cookies) {  String name = cookie.getName();  if ("name".equals(name)) {  //获取指定名称的value  String value = cookie.getValue();  System.*out*.println(value);  }  } |

【关于cookie的path和中文】

|  |
| --- |
| String str = "李四";//将汉字编码  String encode = URLEncoder.*encode*(str, "UTF-8");  //创建cookie 不支持中文  Cookie cookie = new Cookie("name",encode);  //http://localhost:8080/WEB10/TestCookie1  cookie.setPath("/");//权限  cookie.setMaxAge(60\*60);//一个小时  //取中文数据  String value = cookie.getValue();  //解码  String string = URLDecoder.decode(value, "UTF-8"); |

# 完成上次访问时间案例

1 取本地的cookie ？（名字） time 如果有取出值，展示

2 如果没有 向cookie中放置当前时间 time 返回你是第一次访问

|  |
| --- |
| //取数据  Cookie[] cookies = request.getCookies();  if(null!=cookies){  for (Cookie cookie2 : cookies) {  if("time".equals(cookie2.getName())){  String value = cookie2.getValue();  long long1 = Long.*parseLong*(value);  Date date = new Date(long1);  response.getWriter().write("你上次访问的时间是："+date.~~toLocaleString~~());  }  }  }else{  //向往cookie中放置当前时间  //1 当前时间  long millis = System.*currentTimeMillis*();  //2 放值  Cookie cookie = new Cookie("time",millis+"");  //返回给浏览器  response.addCookie(cookie);  response.getWriter().write("你是第一访问");  }  } |

# Session 简介