**DBUtils--数据库工具**

1. **概述**

三个核心功能：

1. QueryRunner 提供对Sql语句的操作的API;
2. ResultSetHandler 接口怎么封装结果集；
3. DBUtils类 一个工具类，定义了关闭资源与事务处理的方法；
4. **QueryRunner核心类**
5. QueryRunner（DataSource）提供数据源（连接池）底层自动维护connection；
6. Update(String sql,Object...params) 增删改（需要参数）的操作；
7. Query(String sql,ResultSetHandler<T> rsh,Object...params)
8. **ResultSetHandler结果集处理类**
9. 处理单条结果集数据的实现类有：

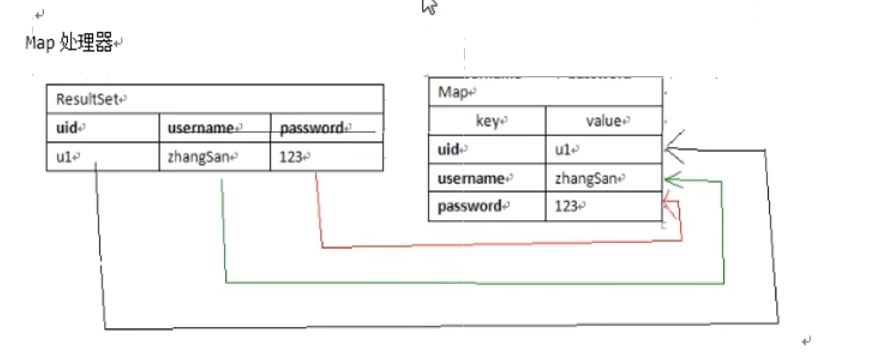
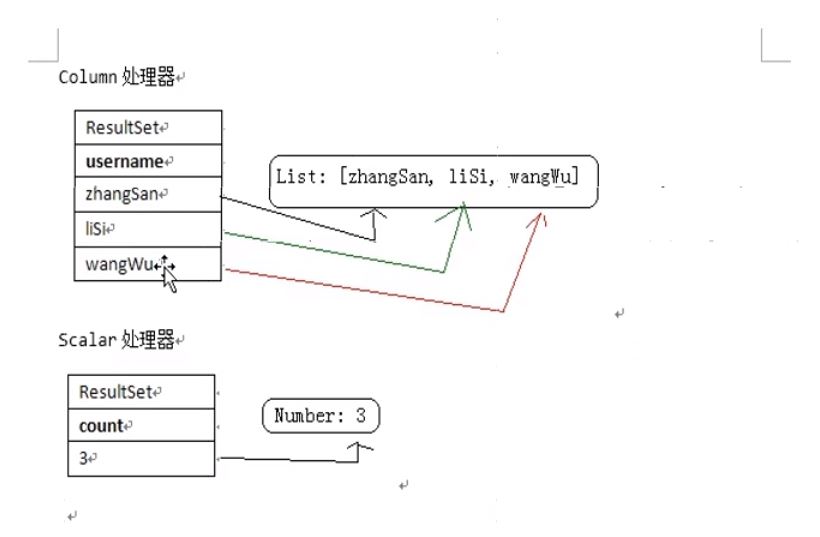
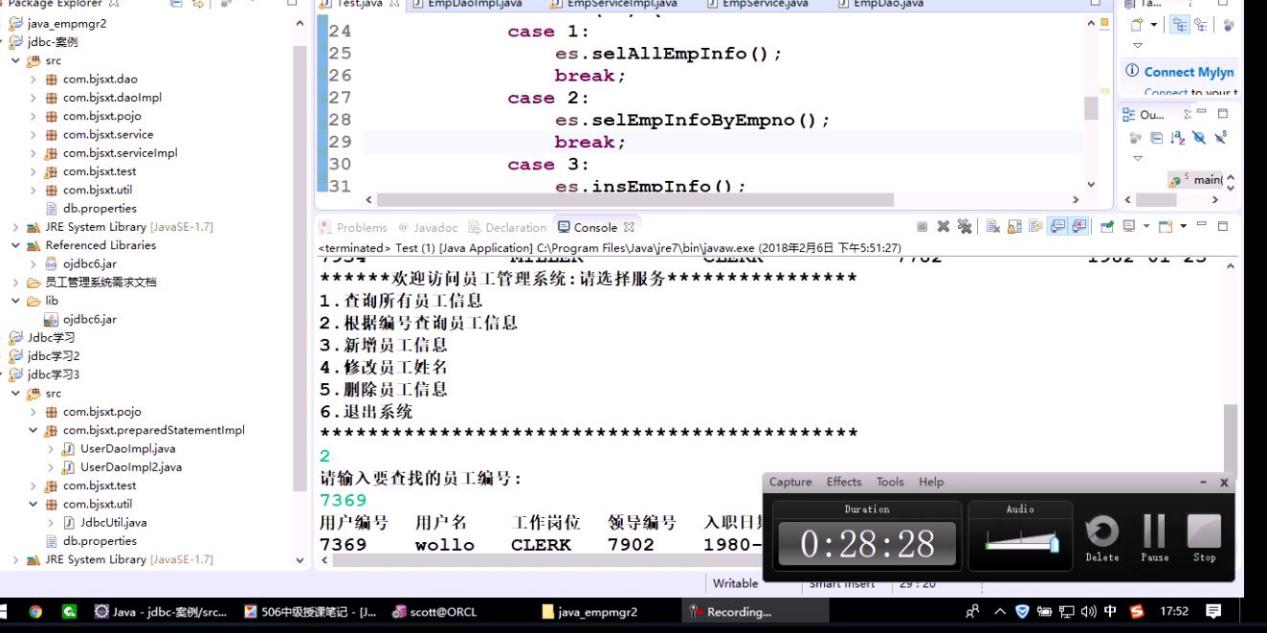
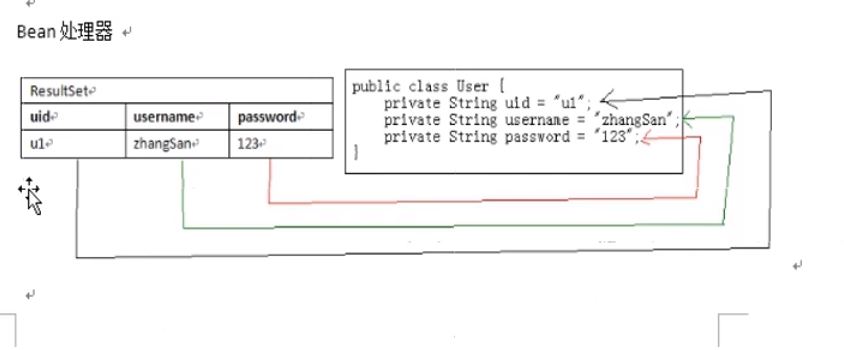
ScalarHandler、ArrayHandler、MapHandler、BeanHandler

1. 处理多条结果集数据的实现类有：

BeanListHandler、AbstractListHandler（ArrayListHandler MapListHandler、ColumnListHandler）、AbstractKeyedHandler（KeyedHandler BeanMapHandler）

重点：ScalarHandler BeanHandler BeanListHandler

1. **使用DBUtils实现数据库增删改查**
2. 使用prepareStatement对象进行对数据库的增删改查
3. JDBC工具类抽取--进行封装
4. Dbcp和C3p0数据库连接池--对C3p0进行封装为C3p0Utils类
5. DBUtils工具进行对数据库增删改查操作--封装



JavaWeb核心知识——Servlet

1. **Servlet概念**

运行在服务器段的Java小程序，servlet的实质就是Java代码，sun公司的一套规范，用来处理客户端请求，响应给浏览器的动态资源。

Servlet规范：（包含三个技术点）

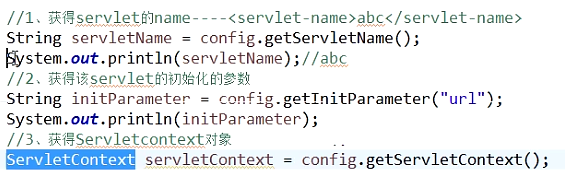
1. servle技术
2. Filter技术
3. Listener技术
4. **Servlet的API与生命周期**

**1.Servlet接口，提供了许多的抽象方法**

1. void init(ServletConfig config)的抽象方法

Servlet对象创建时执行

ServletConfig：servlet对象的配置信息



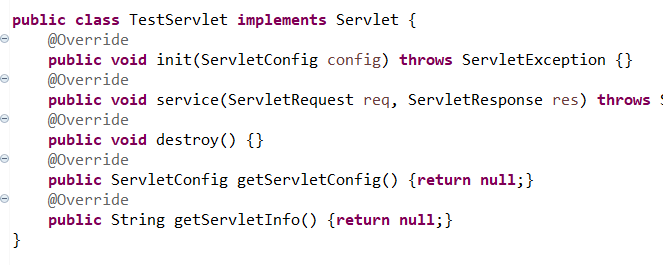
1. void service(req,res)的抽象方法

每一次请求都会执行

1. void destroy()抽象方法

Servlet对象销毁时执行

1. **Servle生命周期（面试）**
2. 何时创建：默认第一次访问servlet时创建
3. 何时销毁：服务器关闭servlet对象就销毁
4. 每次访问必然执行的方法：service方法



1. **Servle接口的默认接口实现类**
2. GenericServlet
3. HttpServlet（GenericServlet子类）

doGet()和doPost()方法

1. **ServletContext接口和ServletConfig接口**
2. ServletContext对象

概念：代表一个Wed应用的环境（上下文）对象，内部封装的是该Web应用的信息。

何时创建：服务器启动（web应用创建）

何时销毁：服务器关闭（web应用卸除）

怎样获取ServletContext对象：(this)config.getServletContect

ServletContext的作用：

1. 获取web应用的初始化参数；

<context-param>

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

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

</context-param>

2)获取Web应用下的资源文件

getResourceAsStream(String path)

GetRealPath(相对web应用的相对地址);

类加载器获取classes文件夹下的资源;

XXX.**class**.getClassLoader().getResource("相对classes文件夹的相对地址");

3）ServletContext域对象，实现多个Servlet对象共享数据

域对象.getAttribute(name);

域对象.setAttribute(name, object);

域对象.removeAttribute(name);

Enumeration 域对象.getAttributeNames();获得Context域中所有域属性名

1. ServletConfig对象

1)获得ServletCoext对象 getServletContext();

2)获得Servlet的名字 getServletName();

3)获得初始化参数值 getInitParameter(String name);

4)获得所有初始化参数名 Enumeration getInitParameterNames();

**四大域对象：**

1. application(ServletContext)对象; 作用范围：当前web应用
2. Session对象； 作用范围：一次会话（浏览器打开到浏览器关闭） 通过request对象获得
3. Request对象； 作用范围：一次请求
4. PageContext对象； 作用范围：当前jsp页面

**请求重定向：**

response.sendRedirect(URL地址) 两次两次响应 服务器外部的请求 浏览器地址栏会发生改变

重定向是取不到request中的数据的。只能用session。

**请求转发：**（多建议使用这个）  
 request.getRequestDispatcher(URL地址).forward(request, response) 浏览器地址栏没有发生改变

属于服务器内部的转发 属于一次请求 转发[中数据](https://www.baidu.com/s?wd=%E4%B8%AD%E6%95%B0%E6%8D%AE&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/qq_34111779/article/details/_blank)的存取可以用request作用域

实践：

1. 解决中文输出乱码的问题：response.setContentType(“text/html;charset=utf-8”);
2. 实现网页定时刷新并跳转: response.setHeader(“Refresh”,”2,http:/www.baidu.com”)
3. 编写一个Servlet实现统计网站被访问次数的功能:

Init(){int count; this this.getServletContext().setAttribute("count", count);}

doGet(req;res){getAttribute(“count”)if(count=0){cpunt=1;}else{count=count+1;setAttribute()}}

1. 解决请求参数的中文乱码:

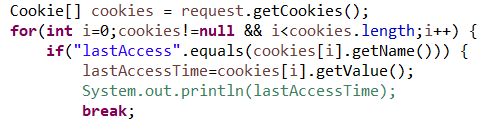
Post:req.setCharacterEncoding(“utf-8”);

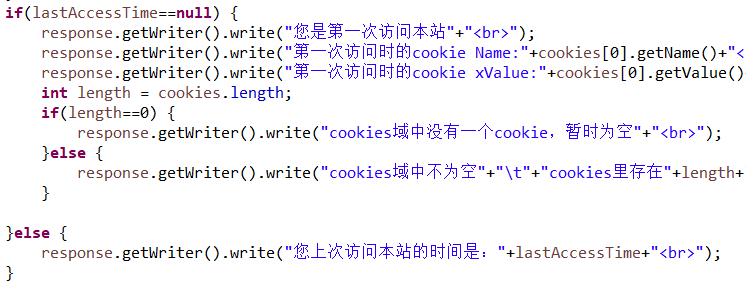
Get:name=new String(name.getBytes(“iso8859-1”),”utf-8”);根据参数而定

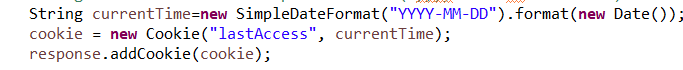
1. 编写一个类，实现资源防盗链的功能



1. 显示用户上次访问时间







1. 实现购物车的功能
2. 创建一个Book类封装图书的信息：id name
3. 创建一个BookDB类模拟数据库存放相关的书籍
4. 创建一个ListBookServlet类来显示数据库中的所有书籍
5. 创建一个PurchaseServlet类将用户购买图书信息保存至一个session域中
6. 创建一个CartServlet显示用户购买的图书
7. 实现首页与注册页面
8. 实现用户自动登录功能
9. 实现全站统一编码功能
10. 使用JDBC完成增删改查
11. 使用DBUtils完成增删改查
12. 实现文件的上传与下载
13. 解决下载中文文件乱码问题
14. 使用MVC设计模式中Modle2思想完成用户注册功能

# **[response.getWriter().write()与out.print()的区别](https://www.cnblogs.com/zhwl/p/3623688.html)**

1、首先介绍write()和print()方法的区别：  
  (1)、write()：仅支持输出字符类型数据，字符、字符数组、字符串等  
  (2)、print()：可以将各种类型（包括Object）的数据通过默认编码转换成bytes字节形式，这些字节都通过write(int c)方法被输出  
 2、介绍response.getWriter()和out的区别：  
  (1)、out和response.getWriter的类不一样，一个是JspWriter，另一个是java.io.PrintWriter。  
  (2)、执行原理不同:JspWriter相当于一个带 缓存功能的printWriter，它不是直接将数据输出到页面，而是将数据刷新到response的缓冲区后再输出，  
      response.getWriter直接输出数据（response.print()），所以（out.print）只能在其后输出。   
  (3)、out为jsp的内置对象，刷新jsp页面，自动初始化获得out对象，所以使用out对象是需要刷新页面的，  
       而response.getWriter()响应信息通过out对象输出到网页上，当响应结束时它自动被关闭，与jsp页面无关，无需刷新页面  
       形象的比喻：当我们调用response.getWriter()这个对象同时获得了网页的画笔，这时你就可以通过这个画笔在网页上画任何你想要显示的东西。  
  (4)、out的print()方法和println()方法在缓冲区溢出并且没有自动刷新时候会产生ioexception，  
       而response.getWrite()方法的print和println中都是抑制ioexception异常的，不会有ioexception