

CSDN

首页 博客 学院 下载 GitChat TinyMind 论坛 问答 商城 ...

搜博文文章

乐于学习、乐于分享、乐于总结

编程是一门技术，更是一门艺术

注册

5订阅

本田新款suv

原

从头认识SpringBatch批处理框架--实例场景一信用卡消费对账

2016年02月26日 11:04:08

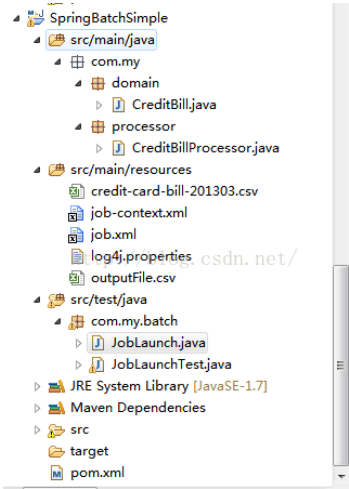
场景说明

个人使用信用卡消费,银行定期发送银行卡消费账单,本例将模拟银行处理个人信用卡消费对账单对账,银行需要定期地把个人消费的记录导出成csv文件

主要流程:

(从credit-card-bill-201303.csv)读取数据---->处理数据----->写数据到 outputFile文件

项目结构



项目结构说明:

- CreditBill:信用卡消费记录领域对象
- CreditBillProcessor: 信用卡消费记录处理类
- jobLaunch:调用批处理作业类
- jobLaunchTest:JUnit单元测试,使用Spring提供的测试框架
- credit-card-bill-201303.csv:信用卡消费账单文件
- job-context.xml:定义作业基础信息
- job.xml:定义作业文件
- outputFile.xml:输出处理过后的信用卡消费账单文件

项目实现步骤详解:

1.准备credit-card-bill-201303.csv对账文件

这里我们使用csv格式的文件,该文件的每一行表示信用卡消费记录,中间用逗号分隔,分别表示:

银行卡账户、账户名、消费金额、消费日期、消费场所如下图所示:

4.04739E+15	,tom,	100,	2013-2-2 12:00,	Lu Jia Zui road
4.04739E+15	,tom,	320,	2013-2-3 10:35,	Lu Jia Zui road
4.04739E+15	,tom,	674.7,	2013-2-6 16:26,	South Linyi road
4.04739E+15	,tom,	793.2,	2013-2-9 15:15,	Longyang road
4.04739E+15	,tom,	360.7,	2013-2-11 11:12,	Longyang road
4.04739E+15	,tom,	893,	2013-2-28 20:34,	Hunan road

2.定义领域对象:

为了与账单文件形成映射,需要新建信用卡消费记录对象 -CreditBill,主要属性:银行卡账户、账户名、消费金额、消费日期、消费场所

具体代码如下:

```
1 package com.my.domain;
2 /**
3  * 信用卡消费记录领域对象
4  * 该类主要用于在ItemReader 读取文件数据之后转换成领域对象CreditBill,
5  * 以便于ItemProcessoe和ItemWriter 操作使用
```

```
8  */ 9 | public class CreditBill {
9  /**
10  * 银行账户
11  */
12  private String accountID;
13  /**
14  * 账户名
15  */
16  private String name;
17  /**
18  * 消费金额
19  */
20  private double amount;
21  /**
22  * 消费日期
23  */
24  private String date;
25  /**
26  * 消费场所
27  */
28  private String address;
29  /**
30  * @return the 银行账户
31  */
32  public String getAccountID() {
33      return accountID;
34  }
35  /**
36  * @param 银行账户 the accountID to set
37  */
38  public void setAccountID(String accountID) {
39      this.accountID = accountID;
40  }
41  /**
42  * @return the 账户名
43  */
44  public String getName() {
45      return name;
46  }
47  /**
48  * @param 账户名 the name to set
49  */
50  public void setName(String name) {
51      this.name = name;
52  }
53  /**
54  * @return the 消费金额
55  */
56  public double getAmount() {
57      return amount;
58  }
59  /**
60  * @param 消费金额 the amount to set
61  */
62  public void setAmount(double amount) {
63      this.amount = amount;
64  }
65  /**
66  * @return the 消费日期
67  */
68  public String getDate() {
69      return date;
70  }
71  /**
72  * @param 消费日期 the date to set
73  */
74  public void setDate(String date) {
75      this.date = date;
76  }
77  /**
78  * @return the 消费场所
79  */
80  public String getAddress() {
81      return address;
82  }
83  /**
84  * @param 消费场所 the address to set
85  */
86  public void setAddress(String address) {
87      this.address = address;
88  }
89  }
90
91 }
92 }
```



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 QQ客服 🗣 客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

该配置文件主要是配置作业仓库、作业调度器、事务管理器,具体代码如下:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:p="http://www.springframework.org/schema/p"
5       xmlns:tx="http://www.springframework.org/schema/tx"
6       xmlns:aop="http://www.springframework.org/schema/aop"
7       xmlns:context="http://www.springframework.org/schema/context"
8       xsi:schemaLocation="http://www.springframework.org/schema/beans
9       http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
10      http://www.springframework.org/schema/tx
11      http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
12      http://www.springframework.org/schema/aop
13      http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
14      http://www.springframework.org/schema/context
15      http://www.springframework.org/schema/context/spring-context-2.5.xsd"
16      default-autowire="byName">
17   <!-- 定义作业仓库SpringBatch提供了两种作业仓库来记录job执行期产生的信息: 一种是内存, 另一种是数据库 -->
18   <bean id="jobRepository"
19         class="org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean">
20   </bean>
21   <!-- 定义作业调度器, 用来启动Job -->
22   <bean id="jobLauncher"
23         class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
24     <property name="jobRepository" ref="jobRepository"/>
25   </bean>
26   <!-- 事务管理器, 用于springbatch框架在对数据操作过程中体统事务能力 -->
27   <bean id="transactionManager"
28         class="org.springframework.batch.support.transaction.ResourcelessTransactionManager"/>
29 </beans>
```



本田新款suv

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 QQ客服 🗣 客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

4.定义job.xml文件

job.xml文件主要配置批处理作业Job、Step、ItemReader(读数据)、ItemProcessoe(处理数据)、ItemWriter(写数据) 具体的配置如下图:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bean:beans xmlns="http://www.springframework.org/schema/batch"
3       xmlns:bean="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:tx="http://www.springframework.org/schema/tx"
7       xmlns:aop="http://www.springframework.org/schema/aop"
8       xmlns:context="http://www.springframework.org/schema/context"
9       xsi:schemaLocation="http://www.springframework.org/schema/beans
10      http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
11      http://www.springframework.org/schema/tx
12      http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
13      http://www.springframework.org/schema/aop
14      http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
15      http://www.springframework.org/schema/context
16      http://www.springframework.org/schema/context/spring-context-2.5.xsd
17      http://www.springframework.org/schema/batch
18      http://www.springframework.org/schema/batch/spring-batch-2.2.xsd">
19   <!-- 引入job-context.xml配置文件 -->
20   <bean:import resource="classpath:job-context.xml"/>
21   <!-- 定义billJob billStep 包含读数据 处理数据 写数据 -->
22   <job id="billJob">
23     <step id="billStep">
24       <tasklet transaction-manager="transactionManager">
25         <!-- commit-interval="2" 表示任务提交间隔的大小 此处表示每处理2条数据 进行一次写入操作 -->
26         <chunk reader="csvItemReader" writer="csvItemWriter"
27               processor="creditBillProcessor" commit-interval="2">
28         </chunk>
29       </tasklet>
30     </step>
31   </job>
32   <!-- 读取信用卡账单文件, CSV格式 -->
33   <bean:bean id="csvItemReader"
34         class="org.springframework.batch.item.file.FlatFileItemReader"
35         scope="step">
36     <!-- 设置读取的文件资源 -->
37     <bean:property name="resource"
38           value="classpath:credit-card-bill-201303.csv"/>
39     <!-- 将文本中的每行记录转换为领域对象 -->
40     <bean:property name="lineMapper">
41       <bean:bean class="org.springframework.batch.item.file.mapping.DefaultLineMapper">
42         <!-- 引用lineTokenizer -->
43         <bean:property name="lineTokenizer" ref="lineTokenizer"/>
44         <!-- fieldSetMapper 根据lineTokenizer中定义的names属性映射到领域对象中去 -->
45         <bean:property name="fieldSetMapper">
46           <bean:bean class="org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper">
47             <bean:property name="prototypeBeanName" value="creditBill">
48             </bean:property>
49           </bean:bean>
50         </bean:property>
51       </bean:bean>
52     </bean:property>
53   </bean:bean>
```

```
52         </bean:property>
53     </bean:bean>
54     <!-- lineTokenizer 定义文本中每行的分隔符号 以及每行映射成FieldSet对象后的name列表 -->
55     <bean:bean id="lineTokenizer"
56         class="org.springframework.batch.item.file.transform.DelimitedLineTokenizer">
57         <bean:property name="delimiter" value=","/>
58         <bean:property name="names">
59             <bean:list>
60                 <bean:value>accountID</bean:value>
61                 <bean:value>name</bean:value>
62                 <bean:value>amount</bean:value>
63                 <bean:value>date</bean:value>
64                 <bean:value>address</bean:value>
65             </bean:list>
66         </bean:property>
67     </bean:bean>
68
69     <!-- 写信用卡账单文件, CSV格式 -->
70     <bean:bean id="csvItemWriter"
71         class="org.springframework.batch.item.file.FlatFileItemWriter"
72         scope="step">
73         <bean:property name="resource" value="file:outputFile.csv"/>
74         <bean:property name="lineAggregator">
75             <bean:bean
76                 class="org.springframework.batch.item.file.transform.DelimitedLineAggregator">
77                 <bean:property name="delimiter" value=","/></bean:property>
78                 <bean:property name="fieldExtractor">
79                     <bean:bean
80                         class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor"
81                         <bean:property name="names" value="accountID,name,amount,date,address">
82                             </bean:property>
83                     </bean:bean>
84                 </bean:property>
85             </bean:bean>
86         </bean:property>
87     </bean:bean>
88     <!-- 领域对象 并标注为原型-->
89     <bean:bean id="creditBill" scope="prototype"
90         class="com.my.domain.CreditBill">
91     </bean:bean>
92     <!-- 负责业务数据的处理-->
93     <bean:bean id="creditBillProcessor" scope="step"
94         class="com.my.processor.CreditBillProcessor">
95     </bean:bean>
96 </bean:beans>
```



联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 · 招聘 · 广告服务 · 网站地图
©2018 CSDN版权所有 京ICP证09002463号
百度提供支持

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

5.创建ItemProcessor

该类主要是用于处理ItemReader读取的数据,通常包括数据过滤、数据转换等操作，此处我们只是简单的打印账单信息。具体代码如下:

```
1  /**
2   *
3   */
4  package com.my.processor;
5
6  import org.springframework.batch.item.ItemProcessor;
7
8  import com.my.domain.CreditBill;
9
10 /**
11  * 处理数据器
12  * @author wbw
13  */
14
15 public class CreditBillProcessor implements ItemProcessor<CreditBill, CreditBill> {
16
17     public CreditBill process(CreditBill bill) throws Exception {
18         System.out.println("信用卡消费领域对象:"+bill.getAccountID()+"-"+bill.getName()+"-"+bill.getAmount()+"-"+bill.getAddress());
19         return bill;
20     }
21 }
```

6.启动Job

我们这里介绍两种启动运行方式,

第一种: main方法

```
1  /**
2   *
3   */
4  package com.my.batch;
5
6  import org.springframework.batch.core.Job;
7  import org.springframework.batch.core.JobExecution;
8  import org.springframework.batch.core.JobParameters;
9  import org.springframework.batch.core.launch.JobLauncher;
```

```
12 13 | /**
14  * 测试
15  * @author wbw
16  *
17  */
18 public class JobLaunch {
19
20     /**
21      * @param args
22      */
23     @SuppressWarnings("resource")
24     public static void main(String[] args) {
25         //初始化应用上下文
26         ApplicationContext context = new ClassPathXmlApplicationContext("job.xml");
27         //获取作业调度, 根据Bean的名称从Spring的上下文获取
28         JobLauncher launcher = (JobLauncher) context.getBean("jobLauncher");
29         //获取任务对象
30         Job job = (Job) context.getBean("billJob");
31         try {
32             JobExecution result = launcher.run(job, new JobParameters());
33             System.out.println(result.toString());
34         } catch (Exception e) {
35             e.printStackTrace();
36         }
37     }
38 }
```



联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

第二种:JUnit单元测试

```
1 package com.my.batch;
2
3
4 import org.junit.After;
5 import org.junit.Before;
6 import org.junit.Test;
7 import org.junit.runner.RunWith;
8 import org.springframework.batch.core.Job;
9 import org.springframework.batch.core.JobExecution;
10 import org.springframework.batch.core.JobParameters;
11 import org.springframework.batch.core.launch.JobLauncher;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.beans.factory.annotation.Qualifier;
14
15 @RunWith(SpringJUnit4ClassRunner.class)
16 @ContextConfiguration(locations={"job.xml"})
17 public class JobLaunchTest {
18     @Autowired
19     private JobLauncher jobLauncher;
20
21     @Autowired@Qualifier("billJob")
22     private Job job;
23
24     @Before
25     public void setUp() throws Exception {
26     }
27
28     @After
29     public void tearDown() throws Exception {
30     }
31
32     @Test
33     public void billJob() throws Exception {
34         JobExecution result = jobLauncher.run(job, new JobParameters());
35         System.out.println(result.toString());
36     }
37 }
```

执行Job后 outputFile文件已录入数据

4.04739E+15	,tom,	100,	2013-2-2 12:00,	Lu Jia Zui road
4.04739E+15	,tom,	320,	2013-2-3 10:35,	Lu Jia Zui road
4.04739E+15	,tom,	674.7,	2013-2-6 16:26,	South Linyi road
4.04739E+15	,tom,	793.2,	2013-2-9 15:15,	Longyang road
4.04739E+15	,tom,	360,	2013-2-11 11:12,	Longyang road
4.04739E+15	,tom,	893,	2013-2-28 20:34,	Hunan road

控制台信息：

10:00:21.265 [main] DEBUG o.s.b.s.t.ResourcelessTransactionManager - Committing resourceless transaction

10:00:21.265 [main] DEBUG o.s.b.repeat.support.RepeatTemplate - Repeat operation about to start

10:00:21.265 [main] DEBUG o.s.b.c.s.c.StepContextRepeatCallback - Preparing chunk execution

10:00:21.265 [main] DEBUG o.s.b.s.t.ResourcelessTransactionManager - Creating new transaction

10:00:21.265 [main] DEBUG o.s.b.repeat.support.RepeatTemplate - Starting repeat context.

10:00:21.265 [main] DEBUG o.s.b.repeat.support.RepeatTemplate - Repeat operation about to start

10:00:21.265 [main] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating instance of bean

10:00:21.265 [main] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Finished creating instance

10:00:21.265 [main] DEBUG o.s.b.repeat.support.RepeatTemplate - Repeat operation about to start

10:00:21.265 [main] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Creating instance of bean

10:00:21.265 [main] DEBUG o.s.b.f.s.DefaultListableBeanFactory - Finished creating instance

10:00:21.265 [main] DEBUG o.s.b.repeat.support.RepeatTemplate - Repeat is complete according to

信用卡消费领域对:4047390012345678-tom-360.0-Longyang road

信用卡消费领域对:4047390012345678-tom-893.0-Hunan road

10:00:21.265 [main] DEBUG o.s.b.item.file.FlatFileItemWriter - Writing to flat file with 2 items

10:00:21.265 [main] DEBUG o.s.b.c.s.item.ChunkOrientedTasklet - Inputs not busy, ended: false

10:00:21.265 [main] DEBUG o.s.b.core.step.tasklet.TaskletStep - Applying contribution: [StepExecution]

10:00:21.265 [main] DEBUG o.s.b.s.t.ResourcelessTransactionManager - Participating in existing transaction

10:00:21.265 [main] DEBUG o.s.b.core.step.tasklet.TaskletStep - Saving step execution before

10:00:21.265 [main] DEBUG o.s.b.s.t.ResourcelessTransactionManager - Participating in existing transaction

10:00:21.265 [main] DEBUG o.s.b.s.t.ResourcelessTransactionManager - Initiating transaction

10:00:21.265 [main] DEBUG o.s.b.s.t.ResourcelessTransactionManager - Committing resourceless transaction

10:00:21.265 [main] DEBUG o.s.b.repeat.support.RepeatTemplate - Repeat operation about to start

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

这里只显示了两条信用卡消费账单信息,因为我们在job.xml文件中设置任务提交间隔的大小 每处理2条数据 进行一次写入操作。如下图

```
<bean:import resource="classpath:job-context.xml" />
<!--定义billJob billStep 包含读数据 处理数据 写数据-->
<job id="billJob">
    <step id="billStep">
        <tasklet transaction-manager="transactionManager">
            <!--commit-interval="2" 表示任务提交间隔的大小 此处表示每处理2条数据 进行一次写入操作-->
            <chunk reader="csvItemReader" writer="csvItemWriter"
                processor="creditBillProcessor" commit-interval="2"
            />
        />tasklet
    />step
</job>
```

总结:

从这个实例场景中,我们学会了使用SpringBatch已经提供好的功能组件和基础设施,快速搭建批处理应用。了解了SpringBatch的一些基本概念:

- JobRepository:作业仓库,负责Job、Step执行过程中的状态保存;
- JobLauncher:作业调度器,提供执行Job的入口;
- Job:作业,由一个或多个的Step组成;
- Step:作业步,Job的一个执行环节,由一个或多个的Step组成Job
- Tasklet:Step中具体执行逻辑的操作,可以重复执行,可以设置具体的同步、异步操作
- Chunk:给定数量的Item集合
- Item:一条数据记录
- ItemReader:从数据源(文件、数据库或消息队列等)中获取Item
- ItemProcessor:在Item写数据之前,对数据进行数据过滤、数据清洗、数据转换、数据校验等操作
- ItemWriter:将Item数据记录批量写入数据源(文件、数据库或消息队列等)

未完待续-----

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/u011659172/article/details/50747688>

个人分类：[从头认识SpringBatch批处理框架](#)

相关热词：[从头jsp](#) [从头编译源码](#) [php从头](#) [kmp从头](#) [从头算](#)

上一篇

什么是批处理

下一篇

从头认识SpringBatch批处理框架---JobRepository数据库存储元数据并分析存储元数据表

招聘海归硕士

想问下国内有海归招聘的平台吗

广告

想对作者说点什么? [我来说一句](#)

大向CZ 2018-03-19 14:50:54 #2楼

没有下文了吗?

免费云主机试用一年

招聘海归硕士

运维是干什么的

东风启辰suv

python培训机构

登录

注册

×

<https://blog.csdn.net/u011659172/article/details/50747688>

Spring Batch 在大型企业中的最佳实践

在大型企业中，由于业务复杂、数据量大、数据格式不同、数据交互格式繁杂，并非所有的操作都能通过交互界面进行处理。而有一...

springbatch学习(一) 大数据批处理框架 Spring Batch全面解析

如今微服务架构讨论的如火如荼。但在企业架构里除了大量的OLTP交易外，还存在海量的批处理交易。在诸如银行的金融机构中，...

Spring Batch批量处理数据实战教程

-

Spring Batch批处理框架


Spring Batch是一个轻量级的，完全面向Spring的批处理框架，可以应用于企业级大量的数据处理系统。Spring Batch以POJO和大...

spring batch 学习多种场景练习demo项目源码

2017年11月09日 275KB 下载

程序猿不会英语怎么行？英语文档都看不懂！

不背单词和语法，一个公式教你读懂天下英文→



一个很好的银行对账工具软件

一个很好的银行对账工具软件: [软件下载]

百万数据的对账优化

写在前面最近线上的对账程序一直不稳定，经常出现对账时间超长，影响结算跑批任务，导致后续业务受影响。原来的流程系统订单...

一个很好的2个银行进行对账单对账程序

2010年05月12日 511KB 下载

记一次支付系统的设计体验

投稿作者：文刀（个人微信公众号：jishuhui_2015），Java Web全栈工程师，高级架构师，技术布道者。曾任两家上市公司的技术...

平台与上游对账逻辑

平台数据：平台交易表trans_info 上游数据：上游对账文件 对账表： check_account_batch // 对账批次表（对账批次号、收单机构...

新哈弗H6来袭,颜值完虐老款,配置全面升级,

哈弗H6



R12现金管理系统：银行对账单（Bank Statement）的标准处理流程

-银行对账单概念银行对账单，是由银行发出的一种单据，详细记录着某个银行账户的资金进出情况。它是银行和企业核对账务的联...

渠道对账及差错处理系统设计

为什么要对账？对账其实是对一定周期内的交易进行双方确认的过程，一般都是在第二天第三方支付公司对前一日交易进行清分，...

大数据批处理框架 Spring Batch全面解析

如今微服务架构讨论的如火如荼。但在企业架构里除了大量的OLTP交易外，还存在海量的批处理交易。在诸如银行的金融机构中，...

大数据批处理框架Spring Batch+spring boot+quartz

大数据时代，数据的收集、处理、存储、分析、挖掘、检索、展示，环环相扣。其中数据处理环节是一个典型的批处理场景——定期...

Spring Batch 批处理框架介绍

前言 在http://www.importnew.com/26177.html 大型的企业应用中，或多或少都会存在大量的任务需要处理，如邮件批量通知所有将...

50万码农评论：英语对于程序员有多重要！

不背单词和语法，老司机教你一个数学公式秒懂天下英语



Spring Batch批处理框架初探

关于spring batch使用整理的一系列教程。



本田新款suv

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 · 招聘 · 广告服务 · 网站地图
©2018 CSDN版权所有 京ICP证09002463号
百度提供支持

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

439

从头认识SpringBatch批处理框架---JobRepository数据库存储元数据并分析存储元数据表

JobRepository存储执行期的元数据，提供两种默认实现,一种是存放在内存中,默认实现类为:MapJobRepositoryFactoryBean。在xml...

Spring Boot 集成 批处理框架Spring batch

Spring Batch是一个轻量级的框架,完全面向Spring的批处理框架,用于企业级大量的数据读写处理系统。以POJO和Spring 框架为基...

Spring batch批处理框架

spring batch框架的简介批处理任务是大多数IT项目的一个重要组成部分，批处理在业务系统中负责处理海量的数据，无须人工干预...

Spring Batch批处理框架 (刘相著) pdf扫描版

2018年06月08日 64B

下载

在职研究生有用吗

在职研究生



spring batch 学习笔记

Spring Batch 是一个轻量级的、完善的批处理框架,旨在帮助企业建立健壮、高效的批处理应用。Spring Batch是Spring的一个子项目...

Spring Batch(7): 并行与扩展

1. 概述Spring Batch提供了多种方式用于处理并行，提高性能。主要分为2大类： - 单个进程，多线程 - 多个进程因此，可以细分为...

Spring.Batch批处理框架

2016年10月23日 170.78MB

下载

PDF

spring batch教程 之 配置并运行Job

Spring batch整体的架构设计使用如下关系图来进行表示： 虽然Job对象看上去像是对于多个Step的一个简单容器，但是开发者必须...

Spring Batch 批处理框架

一、 介绍Spring Batch能够支持简单的、复杂的和大数据量的批处理作业，是一个批处理应用框架，不是调度框架，但需要和调度...

对于程序员来说，英语到底多重要？

不背单词和语法，一个公式秒懂英语！



多重重要？！



Spring Batch批处理框架_PDF电子书下载 带书签目录 高清完整版 .pdf

2018年01月15日 87.89MB

下载

PDF

走进企业级批处理框架--Springbatch

Springbatch是一个轻量级的，完全面向Spring的批处理框架，可以应用于企业级大量的数据处理系统。Spring Batch可以提供大量的...

从头认识SpringBatch批处理框架---Chunk拦截器

Chunk操作中提供了丰富的拦截器机制,拦截器可以实现额外的控制能力,例如日志记录、任务跟踪、状态报告、数据传递等能力, 在S...

搭建批处理框架问题总结Spring Batch + Spring Batch Admin + Quartz


Transaction Manager 事务管理器问题： 问题现象：在Tasklet中调用Dao update更新数据，发现数据库没有变。 问题分析：猜测是...

什么是批处理

现代互联网企业、金融行业、电信行业甚至传动行业通过OLTP(联机事务处理)的业务系统积累了海量的企业数据,需要企业应用能够...

自考大专哪个科目好考呢

自考大专



通用对账系统介绍与设计(上)

http://mp.weixin.qq.com/s/y2l_EYSLlq_239vZXr0OZQ 2017-09-13 王兴建 OmniStack 本文首先介绍了对账的概念、基本内容， ...

java中对于大量数据采用批量处理来提高效率

设计的话，是在dao层写批量新增的方法，以及实现类dao的实现类。在service调用这个dao就可以了！不过最终写的还是单个只...



本田新款suv



联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 · 招聘 · 广告服务 · 网站地图

©2018 CSDN版权所有 京ICP证09002463号

百度提供支持

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

7977

大型的支付系统如何**对账**、风控

本文源自我在知乎上的一个回答，最近在微博上被分享了好多次，就贴在这儿分享给大家。为了更好地解释支付结算系统对账...

支付系统**对账**算法优化方案


一、目前对账的算法：1、从上游渠道（银行、银联等金融机构）获取对账文件，程序逐行解析入库 2、在存储过程中，以上游对账...

支付宝等大型支付系统后台系统是如何**对账**和风控的

支付宝等大型支付系统交易额巨大，后台系统是如何对账和风控的呢？

程序猿不会英语怎么办？英语文档都看不懂！

不背单词和语法，一个公式教你读懂天下英文→



对于一张表的数据很大时查询数据的优化

根据条件查询一张很大的数据：比如，根据 对账日期， 渠道编号和全部的交易类型查询数据查询数据l_cbs_recon_bank_order_c...

java学习7：银行存取款的例子，主要涉及对象同步问题，类似于数据库存取款，数据一...

银行存取款的例子，主要涉及对象同步问题，类似于数据库存取款，数据一致性问题。--存取款都是用线程来控制 注意：这里程序...

java海量数据导出xls分页解决报内存溢出问题

```
/**      * 写XLS文件      * @param fileName 文件名（全路径）      * @param colTitleList 标题      * @param co...
```

java后台实现支付宝**对账**功能

完成支付宝支付、查询的接口之后，我们应该还需要定时与支付宝进行对账，以确保商户系统的订单信息是正确的，想知道支付宝支...

支付系统的**对账**处理

可以说，对账是支付系统最头疼的事情。每一笔交易，都要做到各参与者的记录能够吻合，没有偏差。对账系统的工作，是发现有差...

项目管理 软件

软件项目管理流程总结



微信支付后 **对账**管理

现在微信支付也有一定的用户量所以现在为大家提供一个微信支付的**对账**文档和方法 对账逻辑这个因公司的需求不一样而不同，所...

POI生成excel数据量大时内存溢出解决

通过SXSSFWorkbook替换XSSFWorkbook File file = new File(Constants.WEB_ROOT ...

没有更多推荐了，[返回首页](#)



本田新款suv

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 · 招聘 · 广告服务 · 网站地图
©2018 CSDN版权所有 京ICP证09002463号
百度提供支持

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

1393

1342

1478

1280

104

746

1425

4428

5941

793

个人资料



咩咩文

关注

原创204

粉丝54

喜欢460

评论42

等级: 博客 6

访问: 61万+

积分: 6918

排名: 4517

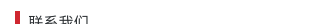
勋章: 恒



北大资源



本田新款suv



javasprint 24篇

联系我们 74篇



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 QQ客服 🗣 客服论坛

关于 · 招聘 · 广告服务 · 网站地图
©2018 CSDN版权所有 京ICP证09002463号
🔍 百度提供支持

经营性网站备案信息 2篇

网络110报警服务 6篇

中国互联网举报中心 1篇

北京互联网违法和不良信息举报中心 1篇

2017年8月 2篇

展开

热门文章

HTML5数据存储---使用clear()方法清除loca
lStorage保存对象的全部数据
阅读量：65524

Spring Boot application.properties或applic
ation.yml相关配置
阅读量：40330

Spring 使用@ComponentScan扫描注解包
阅读量：34979

Spring技术内幕之Spring Data JPA-自定义
Repository实现
阅读量：23394

Spring Boot 集成 批处理框架Spring batch
阅读量：19338

最新评论

Spring技术内幕之Spring...
qq_16430661： 你好作者，这个配置如果写注解的
话是怎么写 package com.data.jpa.conf...

Spring -websocket...
sinat_29440863： 求源码 可以发一下吗 谢谢 1647
123441@qq.com

Spring Boot appl...
qq_26820793： 没有注释,毫无意义...

Spring Boot+Sprin...
wangzheshengcun： [reply]lijigheart[reply] 是这个
注解整合了

个推推送
weixin_40334044： 我想知道你PushMessage消息
对象类文件的内容，这样更全