


(/apps/redirect?  
utm\_source=side-  
banner-click)

# Spring Batch批量处理支付宝账单实践-进阶篇



monkey01 (/u/808054b533e9) + 关注

2017.12.01 14:24 字数 2787 阅读 597 评论 0 喜欢 9

/u/808054b533e9

## 前言

之前一篇写了SpringBatch批量加载支付宝账单的基础篇<http://www.jianshu.com/p/6f038c1f6037> (<https://www.jianshu.com/p/6f038c1f6037>)，实现了将支付宝账单通过springbatch加载、逻辑加工后、输出到自己定义的账单格式文件，上篇也说了只是介绍了基本使用，本篇是上一篇的进阶，还是会继续基于Springbatch全程使用javaconfig的方式实现，数据加载入库、异常数据处理、并行、定时任务等，在写这篇文章前，发现全网写的关于Springbatch的文章绝大部分都是基于XML配置的，随着springboot的逐渐普及，大家也都习惯了抛弃xml，使用javaconfig来配置项目，但是网上的blog包括spring官网对springbatch的javaconfig都介绍的很少，本篇就帮大家通过javaconfig配置整个springbatch，并实现一些高级用法。

## 1. 加载数据到数据库

在批量过程中一般都需要将数据持久化，所以我们介绍下如何将批量数据加载到传统的RDBMS数据库中，我这里使用的是mysql，对于使用oracle或者其他数据库的同学，直接替换datasource就可以了。

首先在pom中加上jdbc和mysql—connect依赖：

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.44</version>
</dependency>
```

在配置文件中加上数据库的配置

```
spring.datasource.url=jdbc:mysql://localhost:3306/test
spring.datasource.username=root
spring.datasource.password=XXXX
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

加上DataSource Config配置类

```
@Configuration
public class MySQLDataSourceConfig {
    @Bean(name = "mysqlDataSource")
    @Qualifier("mysqlDataSource")
    @ConfigurationProperties(prefix="spring.datasource")
    public DataSource mysqlDataSource() {
        return DataSourceBuilder.create().build();
    }

    @Bean(name = "mysqlJdbcTemplate")
    public JdbcTemplate mysqlJdbcTemplate(
        @Qualifier("mysqlDataSource") DataSource dataSource) {
        return new JdbcTemplate(dataSource);
    }
}
```

这里实现一个数据库写入的功能，就是从支付宝账单中读取的数据写入数据库的ItemWriter类，之前有读者说账单格式不对，这里解释下这里用的账单是我从网上随便搜的支付宝账单和实际支付宝商户使用的账单格式是有比较大的差异，这里只是做个例子，让大家学习下springbatch，实际使用过程中还是要改下数据结构以官方提供的加载数据结构为准。新建AlipayDBItemWriter类，这个类实现了ItemWriter接口，上一篇文章也介绍了实现该接口需要实现write（list）方法，整体的逻辑比较简单就是将从ItemReader中读取到的数据加载到数据库中，代码一看就懂。



```
@Service
public class AlipayDBItemWriter implements ItemWriter<AlipayTranDO> {
    private static final String INSERT_ALYPAY_TRAN =
        "insert into alipay_tran_today(tran_id, channel, tran_type, counter_party, goods, amount, is_debit_credit, state) values(?, ?, ?, ?, ?, ?, ?, ?)";

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public void write(List<? extends AlipayTranDO> list) throws Exception {
        for(AlipayTranDO alipayTran : list){
            jdbcTemplate.update(INSERT_ALYPAY_TRAN,
                alipayTran.getTranId(),
                alipayTran.getChannel(),
                alipayTran.getTranType(),
                alipayTran.getCounterparty(),
                alipayTran.getGoods(),
                alipayTran.getAmount(),
                alipayTran.getIsDebitCredit(),
                alipayTran.getState());
        }
    }
}
```

(/apps/redirect?utm\_source=side-banner-click)

写好了ItemWriter最后就是写batchconfig了，直接将上一篇的step1 () 方法复制一份改为step2，将writer改为刚才新建的AlipayDBItemWriter，将step2 () 放到job中，直接运行就可以去数据库看结果了，这里要注意下要提前到数据库中建好表。

```
@Bean
public Step step2() {
    return stepBuilderFactory.get("step2")
        .<AlipayTranDO, AlipayTranDO> chunk(10)
        .reader(alipayFileItemReader.getMultiAliReader())
        .writer(alipayDBItemWriter)
        .build();
}
```

alipay\_tran\_today表结构

```
CREATE TABLE `alipay_tran_today` (
  `tran_id` varchar(40) DEFAULT NULL,
  `channel` varchar(20) DEFAULT NULL,
  `tran_type` varchar(10) DEFAULT NULL,
  `counter_party` varchar(20) DEFAULT NULL,
  `goods` varchar(40) DEFAULT NULL,
  `amount` varchar(20) DEFAULT NULL,
  `is_debit_credit` varchar(10) DEFAULT NULL,
  `state` varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

2. 批量数据校验清洗

下面我们再介绍下批量数据的校验清洗，如果通过springbatch来实现，在实现批量的过程中，我们在加载数据前都会去判断加载的数据格式、内容等是否符合要求，对于不符合要求的数据是抛出异常还是暂时将这些异常数据加载到一张异常数据表中待后续处理等等都需要自己去实现，提前将不符合要求的数据清洗出去后，再对这些格式正确的数据进行加载并进行处理，这样可以防止加工到最后一步发现数据有问题或者将错误数据加工出来出了报表，也可以防止因为脏数据导致后续批量中断等问题。

在springbatch中，我们可以将数据校验的过程写成一个processor，在数据读取后进行逻辑操作前进行数据校验，将不合格的数据剔除或者暂存或者进行修复等。

下面我们新建AlipayValidateProcessor类，用于对支付宝账单数据的校验，这里实现的逻辑非常简单，只是将金额字段小于0的作为异常抛出Exception并没有做其他特殊处理，这里我们最好是自定义EXception，然后在配置Step的时候对自定义的Exception进行skip exception处理。

```
public class AlipayValidateProcessor implements ItemProcessor<AlipayTranDO, AlipayTranDO> {
    private static final Logger log = LoggerFactory.getLogger(AlipayValidateProcessor.class);

    @Override
    public AlipayTranDO process(AlipayTranDO alipayTranDO) throws Exception {
        if(Double.parseDouble(alipayTranDO.getAmount()) < 0){
            log.info("validate error: " + alipayTranDO.toString());
            throw new Exception();
        }else{
            return alipayTranDO;
        }
    }
}
```



下面就是要将数据校验processor和数据加工processor多processor串联执行，如果要串联执行processor这就要用到CompositeItemProcessor类，将多个processor按顺序加入list，并将processor list赋值给CompositeItemProcessor的delegete，将这个CompositeItemProcessor复合处理器作为整个step的processor，这样在step执行的时候就会按照我们设置的复合processor先进行数据校验清洗，再进行数据加工，这里有个地方要注意下，顺序执行的processor的输出类型对应的是下一个processor的输入类型，第一个processor的输入类型一定要是step的输入类型，最后一个processor的输出类型一定是step的输出类型，例如例子中的step输入为AlipayTranDO，输出为HopPayTranDO，所以我们AlipayValidateProcessor的输入为AlipayTranDO，输出也为AlipayTranDO，AlipayItemProcessor的输入为AlipayTranDO，输出为HopPayTranDO。

(/apps/redirect?utm\_source=side-banner-click)

```
public Step step3() {  
  
    CompositeItemProcessor<AlipayTranDO, HopPayTranDO> compositeItemProcessor = new CompositeItemProcessor();  
    List compositeProcessors = new ArrayList();  
    compositeProcessors.add(new AlipayValidateProcessor());  
    compositeProcessors.add(new AlipayItemProcessor());  
    compositeItemProcessor.setDelegates(compositeProcessors);  
  
    return stepBuilderFactory.get("step3")  
        .<AlipayTranDO, HopPayTranDO> chunk(10)  
        .reader(alipayFileItemReader.getMultiAliReader())  
        .processor(compositeItemProcessor)  
        .writer(alipayFileItemWriter.getAlipayItemWriter())  
        .build();  
}
```

3. 异常数据处理

对于在处理过程中如果遇到异常数据并且异常数据不是特别多的情况下，为了保证批量的顺利执行，一般采取的做法是设置一个允许跳过的次数，这样在遇到一些可以容忍的错误类型并行错误次数比较少的环境下就可以继续执行，而不用中断批量，做过运维的同学一定深有感触，夜间批量中断是多么痛苦的事情，第二天运维的同学一定会顶着黑眼圈找开发负责人去投诉批量中断的事情，所以为了保证运维人员的身体健康，批量一定要有一定的异常数据容错机制。springbatch支持设置skip的记录最高次数，同样也可以设置哪些错误可以跳过，哪些不能跳过，如果不设置那么只要批量执行有Exception，那就会中断整个step。在springbatch中使用skip非常简单，我们这个例子中对于异常没有自定义，只是使用了Exception，这样可以捕获到大部分异常，实际使用过程中功能建议自定义Exception，这样处理错误更有针对性，对于跳过的记录，我们还需要设置一个SkipListener类用于监听当出现跳过时回调进行处理的动作。SkipListener接口需要实现3个回调方法，开发者可以分别在这3个回调方法中实现相关跳过操作处理，例子中只是简单记录日志信息，我们也可以记录到数据库中，等待后续运维和运营查看数据是否有问题需要处理。

```
public class AlipaySkipListener implements SkipListener<AlipayTranDO, AlipayTranDO> {  
    private static final Logger log = LoggerFactory.getLogger(AlipaySkipListener.class);  
  
    @Override  
    public void onSkipInProcess(AlipayTranDO alipayTranDO, Throwable throwable) {  
        log.info("AlipayTran was skipped in process: "+alipayTranDO);  
    }  
    @Override  
    public void onSkipInRead(Throwable arg0) {  
    }  
    @Override  
    public void onSkipInWrite(AlipayTranDO alipayTranDO, Throwable throwable) {  
        log.info("AlipayTran was skipped in process: "+alipayTranDO);  
    }  
}
```

除了可以设置skip我们还可以设置重试次数，例如我们需求中会去下载商户的支付宝账单，那么有可能因为网络原因导致批量的时候某个商户账单无法下载下来，那么为了保证批量能够执行下去，那么我们可以让批量程序进行一定次数的重试，如果重试多次后还不行那么将进行跳过或报错处理。



```
public Step step2() {
    return stepBuilderFactory.get("step2")
        .<AlipayTranDO, AlipayTranDO> chunk(10)
        .reader(alipayFileItemReader.getMultiAliReader())
        .writer(alipayDBItemWriter)
        .faultTolerant()
        .skipLimit(20)
        .skip(Exception.class)
        .listener(listener)
        .retryLimit(3)
        .retry(RuntimeException.class)
        .build();
}
```

(/apps/redirect?utm\_source=side-banner-click)

4. 并行配置

我们在日常处理批量的过程中，为了减少批量时间我们一般会将一些处理时间比较长的步骤并行执行充分利用系统资源，缩减批量执行时间，批量中一般有两种并行方式，一种是对单个step多线程并行处理，这种适用于单step数据量特别大的情况，可以利用线程池多线程并行执行数据加工；还有一种是不同step没有依赖关系并行处理，这种并行处理需要充分分析好这些并行step不存在资源争夺，同时程序也是线程安全的，否则会出现很多资源竞争或者串数据的情况，我们这里只介绍单step多线程的实现方式。

需要和其他并发编程一样，需要定义一个ThreadPoolExecutor

```
@Bean
public TaskExecutor taskExecutor() {
    ThreadPoolTaskExecutor taskExecutor = new ThreadPoolTaskExecutor();
    taskExecutor.setMaxPoolSize(4);
    taskExecutor.afterPropertiesSet();
    return taskExecutor;
}
```

对需要并行的step设置taskExecutor就可以实现任务多线程并行执行了，是不是很简单？

```
@Bean
public Step step4() {
    return stepBuilderFactory.get("step3")
        .<AlipayTranDO, HopPayTranDO> chunk(10)
        .reader(alipayFileItemReader.getMultiAliReader())
        .processor(alipayItemProcessor)
        .writer(alipayFileItemWriter.getAlipayItemWriter())
        .taskExecutor(taskExecutor())
        .throttleLimit(4)
        .build();
}
```

5. 定时自动任务配置

springbatch本身只是批量框架并没有定时执行的功能，这里我们需要借助spring的schedule实现定时任务功能，做到批量无人值守自动执行，如果要更强大的功能可以使用Quartz来实现更加花样百出的定时功能，这里我们需求没那么复杂使用scheduler就能很好的完成所有功能，只需要一句注解，设置下cron属性就可以了，cron属性实例见下：

```
一个cron表达式有至少6个（也可能7个）有空格分隔的时间元素。
按顺序依次为秒（0-59）、分钟（0-59）、小时（0-23）、天（月）（0-31，但是你需要考虑你月的天数）、月（0-12）和星期（0-6，0代表星期天，1-6代表星期一至星期六，7代表星期天）

0 0 10,14,16 * * ? 每天上午10点，下午2点，4点
0 0/30 9-17 * * ? 朝九晚五工作时间内每半小时
0 0 12 ? * WED 表示每个星期三中午12点
"0 0 12 * * ?" 每天中午12点触发
"0 15 10 * * ?" 每天上午10:15触发
"0 * 14 * * ?" 在每天下午2点到下午2:59期间的每1分钟触发
"0 0/5 14 * * ?" 在每天下午2点到下午2:55期间的每5分钟触发
"0 0/5 14,18 * * ?" 在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发
"0 0-5 14 * * ?" 在每天下午2点到下午2:05期间的每1分钟触发
"0 10,44 14 ? 3 WED" 每年三月的星期三的下午2:10和2:44触发
"0 15 10 ? * MON-FRI" 周一至周五的上午10:15触发
"0 15 10 15 * ?" 每月15日上午10:15触发
"0 15 10 L * ?" 每月最后一日的上午10:15触发
"0 15 10 ? * 6L" 每月的最后一个星期五上午10:15触发
"0 15 10 ? * 6L 2002-2005" 2002年至2005年的每月的最后一个星期五上午10:15触发
"0 15 10 ? * 6#3" 每月的第三个星期五上午10:15触发
```

除了使用cron属性，也可以使用fixedRate来设定应用启动后多久执行一次，单位为毫秒。



```
public class BillScheduler {
    @Autowired
    private BillBatchConfig billBatchConfig;

    private static final Logger log = LoggerFactory.getLogger(BillScheduler.class);

    private static final SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");

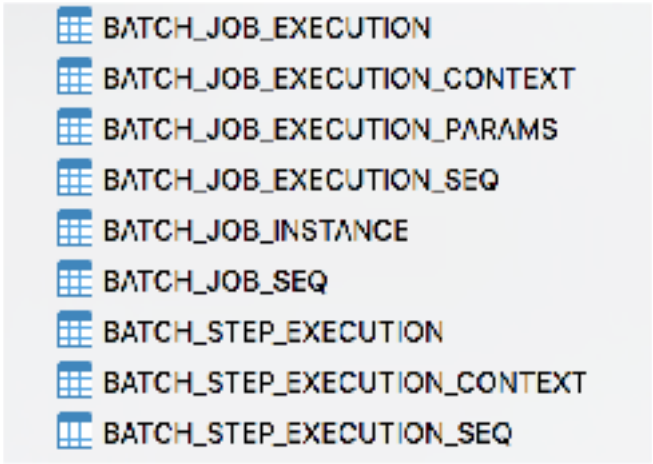
    @Scheduled(initialDelay=10000, fixedRate = 10000)
    public void fixedBillBatch() {
        log.info("job begin {}", dateFormat.format(new Date()));
        billBatchConfig.run();
        log.info("job end {}", dateFormat.format(new Date()));
    }

    @Scheduled(cron="0 15 10 ? * *")
    public void fixedTimePerDayBillBatch() {
        log.info("job begin {}", dateFormat.format(new Date()));
        billBatchConfig.run();
        log.info("job end {}", dateFormat.format(new Date()));
    }
}
```

(/apps/redirect?  
utm\_source=side-  
banner-click)

6. 批量监控设计

批量程序写完了，批量监控也是很大的一块内容，springbatch已经帮大家实现了大部分的监控功能，能够对job执行当前情况和历史情况进行监控并记录，同时也可以对每个job的所有step执行情况进行监控，对step的顺序也可以进行配置，所有的这一切功能都是依赖于springbatch的job repository模块来实现的，因为工程中使用了mysql作为datasource那么只要springbatch第一次启动的时候就会在datasource中新建这些用于批量监控的数据表：



Screenshot 2017-12-01 14.15.27.png

这些批量监控表的具体内容就由大家自己去研究吧，我这里就不一一赘述了，如果觉得springbatch监控做的不好，大家也可以自己去实现批量监控功能，自己实现批量监控的话就需要实现一些job和step的listener接口，编写回调函数，在step和job执行成功后回调这些方法记录执行情况。如果只是实现简单批量那么自带的监控已经够用了。

小结

这篇springbatch的进阶篇主要就是把日常批量程序中经常要用到的功能用springbatch javaconfig的方式实现了一把，对于刚接触springbatch的同学会有很大帮助，同样对于原来写xml配置batch的同学转javaconfig方式也有很大的帮助，代码都放到了github上欢迎下载。

github： <https://github.com/feiweiwei/BillSpringBatch.git> (<https://link.jianshu.com?t=https://github.com/feiweiwei/BillSpringBatch.git>)

喜欢文章的话就鼓励下呗！你的赞赏是我最大的动力！

赞赏支持





monkey01 (/u/808054b533e9)

写了 186577 字，被 251 人关注，获得了 272 个喜欢  
(/u/808054b533e9)

+ 关注

互联网金融架构师 www.monkey01.club


(/apps/redirect?utm\_source=side-banner-click)

喜欢 | 9




更多分享

(http://cwb.assets.jianshu.io/notes/images/20460925/weibo/im



下载简书 App ▶

随时随地发现和创作内容



(/apps/redirect?utm\_source=note-bottom-click)



(/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-comment-form)

评论

智慧如你，不想发表一点想法 (/sign\_in?utm\_source=desktop&utm\_medium=not-signed-in-nocomments-text) 咩~

被以下专题收入，发现更多相似内容

- 

Java开发那些事 (/c/bf318b68fddc?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

SpringF... (/c/1d2b61da81ad?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

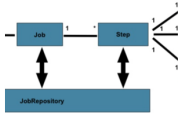
Spring ... (/c/f0cf6ae1754?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

springboot (/c/e0759dd5e6e9?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

starter (/c/5fbd760def60?utm\_source=desktop&utm\_medium=notes-included-collection)
- 


springc... (/c/8d67933d0b87?utm\_source=desktop&utm\_medium=notes-included-collection)

(/p/6f038c1f6037?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)  
**Spring Batch**批量处理支付宝账单实践-基础篇 (/p/6f038c1f6037?utm\_ca...

前言 最近项目要做聚合支付，聚合支付顾名思义就是将市面上常用的第三方支付进行聚合，这样开发者只需要对接我们一方就可以同时对接支付宝、微信支付等其他第三方支付平台，省去了挨个平台对接调试的时间...




monkey01 (/u/808054b533e9)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

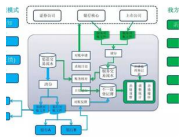
(/p/53ad6fd6e4ba?  
utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

大数据批处理框架 **Spring Batch**全面解析[转] (/p/53ad...

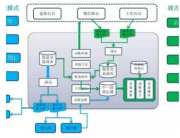
如今微服务架构讨论的如火如荼。但在企业架构里除了大量的OLTP交易外，还存在海量的批处理交易。在诸如银行的金融机构中，每天有3-4万笔的批处理...

 李家沱\_cc8c (/u/4fe7a0629a97?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation) (/apps/redirect?utm\_source=side-banner-click)




(/p/2293f1a6adf6?)



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

大数据批处理框架**Spring Batch**大解析 (/p/2293f1a6adf6?utm\_campaign=...


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，每天有3-4万笔的批处理作业需要处理。针对OLTP，业界有大量的开源框架、优秀...

 爱动脑的程序员 (/u/71104aa3c3f3?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

**Android - 收藏集** (/p/dad51f6c9c4d?utm\_campaign=maleskine&utm\_c...

用两张图告诉你，为什么你的 App 会卡顿？ - Android - 掘金 Cover 有什么料？从这篇文章中你能获得这些料：知道setContentView()之后发生了什么？ ... Android 获取 View 宽高的常用正确方式，避免为零 - 掘金...

 passiontim (/u/e946d18f163c?)


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

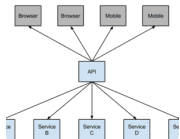
(/p/46fd0faecac1?)

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

**Spring Cloud** (/p/46fd0faecac1?utm\_campaign=maleskine&utm\_conte...

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式...


 卡卡罗2017 (/u/d90908cb0d85?)



utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

失败？成功？ (/p/94007789b789?utm\_campaign=maleskine&utm\_conte...


我们因完美心态、拖延、忽略人而失败，如何成功呢？承认自己的不完美，克服拖延，注重人际交往的反馈，找到问题的本质。我们都喜欢听别人成功的经验，都希望能从中受到启发。从而让自己更快成功。啊...

 呆萌叶子 (/u/36a6d28f5fc1?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

**UI动效学习阶段一** (/p/46f7ed03ed4e?utm\_campaign=maleskine&utm\_c...


这几天都有在学AE制作UI动效，今天用了半天装插件，真是快崩溃啊！

 二怪爱小鲸鱼和龙虾 (/u/e2a88bf0215e?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

**C# 静态方法与非静态方法的比较** (/p/dbb394448134?utm\_campaign=mal...


C#静态方法与非静态方法的区别不仅仅是概念上的，那么他们有什么具体的区别呢？让我们通过本文向你做一下解析。C#的类中可以包含两种方法：C#静态方法与非静态方法。那么他们的定义有什么不同呢？他...

 func\_老衲姓罗 (/u/d6454c17a145?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

**iOS蓝牙开发如何更好地收发数据** (/p/1f41e6fe06bf?utm\_campaign=male...

3月中旬跳槽了，一直在新公司「填坑」，看着「先人」写的代码，觉得是有改善空间的，所以这次想聊下这部分内容——iOS蓝牙开发中如何更好地收发数据。适读对象：想初步了解iOS蓝牙开发的朋友(最...

 AntonyWong (/u/94ddd4cc5002?)


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

(/p/985c49f78730?)

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation)

【古风】如果记忆是风（十） (/p/985c49f78730?utm\_campaign=maleski...

如果记忆是风 | 目录 上一章 | 第九章 (十) 第十章 辛右望着高塔，觉得自己的心像是要跳出来，甚至可以听到自己血脉里鲜血流动的声音。 我看他一直在...

 辰小夕 (/u/3e2750894d4d?)



(/apps/redirect?utm\_source=side-banner-click)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)