

[首页](#) [资讯](#) [精华](#) [论坛](#) [问答](#) [博客](#) [专栏](#) [群组](#) [下载](#)  
[您还未登录!](#) [登录](#) [注册](#)



m635674608

浏览: 3543710 次

性别:

来自: 南京



最近访客 [更多访客>>](#)



[雪山飞狐](#)



[1524031911](#)



[gaoshaoye](#)



[liaoliao2010](#)

博主相关

- [博客](#)
- [微博](#)
- [相册](#)
- [收藏](#)
- [留言](#)
- [关于我](#)

文章分类

[全部博客 \(2847\)](#)

[java \(1094\)](#)

[hadoop \(37\)](#)

[jvm \(39\)](#)

[hbase \(11\)](#)

[cat \(25\)](#)





[设计模式 \(2\)](#)  
[架构 \(275\)](#)  
[操作系统 \(91\)](#)  
[maven \(35\)](#)  
[tapestry \(1\)](#)  
[mybatis \(9\)](#)  
[MQ \(101\)](#)  
[zookeeper \(18\)](#)  
[搜索引擎, 爬虫 \(208\)](#)  
[分布式计算 \(45\)](#)  
[c# \(7\)](#)  
[抓包 \(28\)](#)  
[开源框架 \(45\)](#)  
[虚拟化 \(12\)](#)  
[mongodb \(15\)](#)  
[计算机网络 \(2\)](#)  
[缓存 \(97\)](#)  
[memcached \(6\)](#)  
[分布式存储 \(13\)](#)  
[scala \(5\)](#)  
[分词器 \(24\)](#)  
[spark \(104\)](#)  
[工具 \(23\)](#)  
[netty \(5\)](#)  
[Mahout \(6\)](#)  
[neo4j \(6\)](#)  
[dubbo \(36\)](#)  
[canal \(3\)](#)  
[Hive \(10\)](#)  
[Vert.x \(3\)](#)  
[docker \(115\)](#)  
[分布式追踪 \(2\)](#)  
[spring boot \(5\)](#)  
[微服务 \(56\)](#)  
[淘客 \(5\)](#)  
[mesos \(67\)](#)  
[php \(3\)](#)  
[etcd \(2\)](#)



[股票\(1\)](#)[Android\(2\)](#)[房产\(1\)](#)[运维\(6\)](#)[网关\(3\)](#)

社区版块

[我的资讯\(0\)](#)[我的论坛\(28\)](#)[我的问答\(20\)](#)

存档分类

[2017-12\(8\)](#)[2017-11\(75\)](#)[2017-10\(38\)](#)[更多存档...](#)

最新评论

[明兜3号](#)：部署落地+业务迁移 玩转k8s进阶与企业级实践技能（又名：Ku ...[Kubernetes系统常见运维技巧](#)[q328965539](#)：牛掰啊 资料收集的很全面[HDFS小文件处理解决方案总结+facebook\(HayStack\) + 淘宝 \(TFS\)](#)[guichou](#)：fluent挂载了/var/lib/kubelet/pods目 ...[kubernetes上部署Fluentd+Elasticsearch+kibana日志收集系统](#)[xu982604405](#)：System.setProperty("java.r ...[jmx rmi 穿越防火墙问题及jmxmp的替代方案](#)[大漠小帆](#)：麻烦问下，“获取每个Item相似性最高的前N个Item”，这个 ...[协同过滤推荐算法在MapReduce与Spark上实现对比](#)

JGit

博客分类：

[java](#)

如果你想知道像 `git init`, `git checkout` 等等的基本 Git 命令是如何在 JGit 上运行的话，请往下看。

本教程提供了最常用的 git 命令以及它们在 JGit 上对应部分的概述。本教程将依次介绍：创建一个存储库，从远程获取内容，向历史版本中添加和删除文件，检查历史操作，并最终将更改的文件推回到原始存储库。

JGit 提供了一个类似于 Git 高级命令的 API，你会在 JGit 里面的命令行用这个代码

```
git.commit().setMessage("My first commit").call();
```

代替

```
git commit -m "My first commit"
```

所有的 JGit 命令 有一个 `call()` 方法，在命令被建立之后用来实际执行它。这些类的命名是在各自的 Git 命令后加上后缀命令。虽然有些命令提供一个公共构造函数，还是建议使用 Git 的工厂类来创建命令，如在上面的示例中所示。

## 获取库

但是在更深入到 JGit API 之前，让我们先得到库。最常见的取得 JGit 的方法可能就是从 Maven 仓库中取。但是如果你更喜欢 OSGi 捆绑，也有 P2 仓库供你使用。 [下载页面](#) 上列出了整合该库的必要信息。

对于本文所涉及的范围来说，整合在 `project/bundle` 下的 `org.eclipse.jgit` 核心库已经足够了。如果你对其他 JGit 的源代码库感兴趣，我推荐阅读 [JGit 资源介绍](#)

## 创建一个仓库

首先，我们需要一个仓库。为了得到一个仓库，我们可以初始化一个新的存储库或克隆一个现有的。

我们可以用 `Init` 命令创建一个新的仓库。下面的语句

```
Git git = Git.init().setDirectory("/path/to/repo").call();
```

将在 `setDirectory()` 下设置的路径中创建一个带工作目录的仓库。`.git` 目录将会直接在 `/path/to/repo/.git` 路径下面。对 `Init` 命令的详细解释请参阅文章 [在 JGit 中初始化 git 仓库](#)。

```
Git git = Git.cloneRepository()  
    .setURI("https://github.com/eclipse/jgit.git")
```

```
.setDirectory( "/path/to/repo" )  
.call();
```

现有的仓库可以用 Clone 命令克隆

```
Git git = Git.cloneRepository()  
.setURI( "https://github.com/eclipse/jgit.git" )  
.setDirectory( "/path/to/repo" )  
.call();
```

上面的代码将克隆远程 JGit 仓库到本地目录的路径 'repath/to/po' 下。所有 Clone 命令的选项在 [如何在 JGit 中克隆 Git 仓库](#) 中进行了更加详细的解释。

结束时关闭 Git

注意那些会返回实例的命令例如 Init 命令或者 Clone 命令当不再需要时如果没有明确地关闭 (git.close()) 的话, 可能会造成文件句柄泄露。

幸运的是, Git实现了 Autocloseable 以便你可以使用 [try-with-resources 声明](#)。

## 填充仓库

现在我们有仓库了, 我们可以填充它的历史了。但为了提交文件首先需要将它添加到所谓的索引 (又名临时区)。只有在索引中增加或者删除的文件才会被 commit 命令考虑。

因此 JGit 命令就是 (你猜一下) Add 命令。

```
DirCache index = git.add().addFilePattern( "readme.txt" ).call();
```

因此上述代码把 readme.txt 这个文件加入到了索引中。值得注意的是文件的实际内容是被复制到了索引。这意味着后来的对这个文件的修改不会被包含在索引中, 除非你再次添加。

给 addfilepattern() 的路径必须相对于工作目录的根。如果一个路径没有指向一个现有的文件, 它就被忽略了。

虽然方法名称表明模式是可以接受的, 但是在这里 JGit 的支持是有限的。传一个 '.' 将在工作目录中添加所有文件。但是 \*.java 之类的用法是不支持的, 尽管在原生 Git 命令中可以这么用。

通过 call() 返回的索引, 在 JGit 命名为 DirCache, 通过检查验证了它实际上包含了我们所期望的东西。它的 getEntryCount() 方法返回了文件总数, 而且 getEntry() 方法返回了指定位置的入口。

现在用 Commit 命令把变化存入仓库的准备工作已经一切就绪。

```
RevCommit commit = git.commit().setMessage( "Create readme file" ).call();
```

至少提交消息必须被指定, 否则 call() 将会抛出一个 NoMessageException 异常。但是一个空的消息是被允许的。作者和提交者如果没有被相应的标记的方法指出的话, 将会从配置文件中取得。

返回的 RevCommit 描述了这条 commit 的信息作者, 提交者, 时间戳, 当然还有一个指针指向文件树和构成此提交的目录。

新增或者修改的文件需要被增加, 同样的删除的文件也需要被明确地被移除。Rm命令是相对于Add命令的, 使用方式也是一样 (当然是相反的结果)。

```
DirCache index = git.rm().addFilepattern( "readme.txt" ).call();
```

以上的命令会再次的删除指定的文件。因为这是仓库中的唯一的文件, 当查询该条目的编号时, 返回的索引值将为 0。

除非 setCached 被指定为 true, 文件也会在工作目录被删除。因为 Git 不会跟踪目录, Rm 命令也会删除指定文件的空的父目录。

试图删除一个不存在的文件会被忽略掉。但是和 Add 命令不一样的是, Rm 命令在 addFilepattern() 命令中不接受通配符。任何要被删除的文件需要被单独指定。

这些改变将会与下一个提交一起被存储在仓库中。请注意, 创建一个空的提交是完全合法的, 例如一个没有文件增加或者删除执行过的仓库。虽然我也没有想到一个得体的使用情况。

## 仓库的状态

状态命令会列出索引和目前最新的提交之间有差异的文件或者工作目录和索引之间有差异的文件或者没有被 Git 跟踪到的文件。

Status 命令以其最简单的形式, 收集了所有属于库文件的状态。

```
Status status = git.status().call();
```

对象状态的 `get` 类方法应该是意义自明的。它们返回在该方法名描述的状态中的文件名的集合。例如，如前文所述中在 `readme.txt` 文件被增加到索引后，`status.getAdded()`这个方法会返回一个包含刚刚添加的文件的集合。如果没有任何差别，也没有未被跟踪到的文件，那么 `Status.isClean()` 这个方法会返回 `true`。而顾名思义，如果有未提交的改变的话 `Status.hasUncommittedChanges()` 这个方法会返回 `true`。

`Status` 命令可以用 `addPath()` 方法来配置为只显示某些文件状态。给定的路径必须指明一个文件或者目录名。不存在的路径会被忽视，并且正则表达式或通配符是不被支持的。

```
Status status = git.status().addPath( "documentation" ).call();
```

在以上的例子中，在 `documentation` 目录下递归的所有文件的状态将被计算出。

## 探索库

现在，该库有一个（小的）历史，我们将查看列出已存在的提交的命令。

最简单的 `git log` 对应在 JGit 中的形式允许列出所有的可以从目前的 `HEAD` 中得到的提交。

```
Iterable iterable = git.log().call();
```

返回的迭代器可以用来循环遍历所有的用 `Log` 命令找到的提交。

对于更高级的使用情况，我建议直接使用 `RevWalk` API，这个同样类也被 `Log` 命令所使用。除了提供更大的灵活性，也避免了可能的资源泄漏，因为 `Log` 命令内部使用的 `RevWalk` 是永远不会关闭。

例如，它的 `markStart()` 方法也可以用来列出可从其他分支得到的提交（或概括地说，从其他参考）。

不幸的是，只有 `ObjectId` 会被接受，因此要求的参考需要首先被解决。在 JGit 中的 `ObjectId` 封装了一个指向 Gits `object` 数据库中的对象的 SHA-1 hash。在这里，指向提交的 `ObjectId` 是需要的，解决在这里的意思是要得到这个特定的 `ref` 指向的 `ObjectId`。

全部放在一起，看起来像下面的代码段：

```
Repository repository = git.getRepository()
try( RevWalk revWalk = new RevWalk( repository ) ) {
    ObjectId commitId = repository.resolve( "refs/heads/side-branch" );
    revWalk.markStart( revWalk.parseCommit( commitId ) );
    for( RevCommit commit : revWalk ) {
        System.out.println( commit.getFullMessage );
    }
}
```

分支的 `side-branch` 分支点的提交 `id` 被获取，然后 `RevWalk` 被委派去遍历历史记录。因为 `markStart()` 方法需要 `Rev` 命令，`RevWalk` 的 `parse` 命令被用来将提交 `id` 转化为实际提交。

当 `RevWalk` 设置完毕后，上述代码段循环打印出每条提交的信息。`try-with-resource` 声明确保了 `RevWalk` 会在结束后被关闭。需要注意的是，多次调用 `markStart()` 方法来包含多次的参考之间的往返移动是合法的。

一个 `RevWalk` 也可以被设定用来过滤提交，既可以通过匹配提交对象的属性，也可以通过匹配它代表的目录树的路径。如果提前知道的话，不感兴趣的提交和他们的祖先链可以从输出中排除。当然，输出可以进行排序，例如按日期或拓扑（所有子类在父类前）。但是这些特性已经在本文的范畴之外，但可能会安排在将来它自己的文章中。

## 和远程仓库交互

本地库经常是从远程库中克隆来的。本地的改变最终应该发布到原始仓库中。为了完成这一操作，JGit 提供了一个 `push` 命令。

最简单的形式将当前分支推到其相应的远程分支。

```
Iterable iterable = local.push().call();
PushResult pushResult = iterable.iterator().next();
Status status
    = pushResult.getRemoteUpdate( "refs/heads/master" ).getStatus();
```

该命令将返回一个可迭代的 `PushResults`。在上述例子中的迭代只有一个元素。为了验证推送是否成功，`pushResult` 可以被要求返回一个 `RemoteRefUpdate` 给指定的分支。

一个 `RemoteRefUpdate` 详细描述了更新了什么和如何更新的。但它同时也包含了一个状态属性，用以对结果状态进行总结性表述。如果状态返回 `OK`，我们可以放心的确认操作是成功的。

即使命令不给任何建议也能工作，它有很多的选项，以下只列出了最常用的。默认情况下，该命令推送到了默认的叫作 `'origin'` 的远程库。使用 `setRemote()` 指定一个不同的远程存储库的 URL 或名称。如果其他分支比当前分支

更应该推送的话，`refspecs` 可以用 `setRefSpec()` 来指定。标签是否需要被转移可以用 `setPushTags()` 来控制。最后，如果你不确定结果是否理想，则有一个 `dry-run` 选项，可以模拟一个推送操作。

现在，我们已经看到了如何将本地对象传输到远程存储库中，我们将看一下反向操作如何工作。`Fetch` 命令的使用和它相对应的推送命令一样，而且也能用默认设置操作成功。

```
FetchResult fetchResult = local.fetch().call();
TrackingRefUpdate refUpdate
    = fetchResult.getTrackingRefUpdate( "refs/remotes/origin/master" );
Result result = refUpdate.getResult();
```

没有进一步的配置，该命令得到的改变是当前分支所对应的默认远程仓库。

`FetchResult` 提供了详细的操作结果的信息。每一个受到影响的分支可以获得一个 `TrackingRefUpdate` 实例。最有趣的可能是 `getResult()` 的返回值对更新操作的结果进行了总结。此外，它还包括了哪个本地 `ref` (`getLocalBame()`) 和哪个远程 `ref` (`getRemoteName()`) 更新的信息，还有本地 `ref` 在更新前和更新后 (`getOldObjectId()` 和 `getNewObjectid()`) 指向的对象 `id` 的信息。

如果远程存储库需要认证，和其他所有和远程库通信的命令一样，`Push` 命令和 `Fetch` 命令可以同样的方式来准备。详细的讨论可以在 [JGit Authentication Explained](#) 这篇文章中找到。

## JGit新手指南总结

现在轮到你来玩转 JGit 了。高级的 JGit API 不难理解。如果你知道如何使用 `git` 命令，你可以轻松的猜出 JGit 中哪个类和方法的使用。

虽然不是细到支持所有的 `git` 命令，最常用的功能还是有可靠的支持。如果有一些关键的东西丢失，你可以求助于较低级别的 JGit APIs 来排除故障。

此文的代码片段是一个学习测试的摘录集合。



完整的版本可以在这里被找到: <https://gist.github.com/rherrmann/433adb44b3d15ed0f0c7>

如果你还是有困难或者问题, 请留下评论或者向友好和乐于助人的 [JGit 社区](#) 需求帮助

<http://www.tuicool.com/articles/M7ZBJby>

[https://git-scm.com/book/zh/v2/%E5%B0%86-Git-](https://git-scm.com/book/zh/v2/%E5%B0%86-Git-%E5%B5%8C%E5%85%A5%E4%BD%A0%E7%9A%84%E5%BA%94%E7%94%A8-JGit)

[%E5%B5%8C%E5%85%A5%E4%BD%A0%E7%9A%84%E5%BA%94%E7%94%A8-JGit](https://git-scm.com/book/zh/v2/%E5%B5%8C%E5%85%A5%E4%BD%A0%E7%9A%84%E5%BA%94%E7%94%A8-JGit)

分享到:  

[rocketmq\\_cluster下concurrently重试机制...](#) | [Redis内存淘汰机制](#)

2016-09-05 11:11

浏览 892

[评论\(0\)](#)

分类: [编程语言](#)

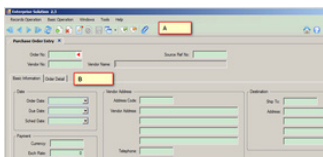
[查看更多](#)

评论

发表评论




[您还没有登录,请您登录后再发表评论](#)



**500强企业都在用的erp管理系统，你还在等什么**

erp系统公司

Kryo 使用指南 2017-12-05 20:14  883

1、Kryo 的简介 Kryo 是一个快速序列化/ ...

spring session序列化问题排查 2017-12-01 19:07  3257

严重: Servlet.service() for ser ...

利用junit对springMVC的Controller进行测试

2017-11-30 16:26 👁 678

平时对junit测试service/D ...

Java内存模型之重排序 2017-11-29 09:44 👁

356

在执行程序时，为了提供性能，处理器和编译器常...

pmd spotbugs 文档 2017-11-28 10:02 👁 0

[https://pmd.github.io/pmd/pmd ...](https://pmd.github.io/pmd/pmd...)

PMD、FindBug、checkstyle、sonar这些代

2017-11-28 10:01 👁 1540

可以说都是代码静态分析工具，但侧重点不同。pm...

阿里巴巴Java代码规约插件p3c-pmd使用指南

2017-11-23 17:09 👁 941

阿里巴巴Java代码规约插件安装 阿里Java代码规 ...

静态分析工具PMD使用说明 (文章来源: Java E

2017-11-23 17:07 👁 593

质量是衡量一个软件是否成功的关键要素。而对于...

MyBatis 使用 MyCat 实现多租户的一种简单思

2017-11-20 18:27 👁 1861

本文的多租户是基于多数据库进行实现的，数据是...

Spring+MyBatis实现数据库读写分离方案 20

17-11-20 17:15 👁 444

百度关键词：spring mybatis 多数据源 读写分离 ...

数据库连接池druid wallfilter配置 2017-11-2

0 11:38 👁 485

使用缺省配置的WallFilter <be ...

java restful 实体封装 2017-11-16 09:47 👁 9

92

`package com.mogoroom.bs.commo ...`

dak 2017-11-15 11:21 👁 0

`package zzm; import jodd.ht ...`

Java内存模型之从JMM角度分析DCL 2017-1

1-15 09:35 👁 142

DCL，即Double Check Lock，中卫双重检查锁 ...

Java 打印堆栈的几种方法 2017-11-14 09:36


👁 2983

java 中可以通过 eclipse 等工具直接打印堆栈， ...


Servlet Session学习 2017-11-10 09:25 👁 18

8

HTTP 是一种"无状 ...

浅析Cookie中的Path与domain 2017-11-10 09:26  484


Path – 路径。指定与co ...

入分析volatile的实现原理 2017-11-08 09:47  260

通过前面一章我们了解了synchronized是一个重量...

Spring MVC-ContextLoaderListener和Dispac 2017-11-15 09:35  274

Tomcat或Jetty作为Servlet ...

搭建spring框架的时候，web.xml中的spring木 2017-11-07 18:27  671

搭建spring框架的时候，web.xml中的sprin ...

Global site tag (gtag.js) - Google Analytics