

支持功能：

自动打Patch

同时提交多仓、多分支代码、一笔提交生成多个提交记录

前置条件

window 电脑配置java环境变量，支持 java -jar 命令

使用方式

java -jar bianbian_codeCommit.jar [option]

--help, --h, --H <输出帮助文档>

--codecommit_bininit 初始化提交代码需要的 bin 文件

--codecommit 填写好配置信息后进行代码提交

--createpatch 自动生成patch

--opendes 打开详细说明文档 pdf

--help, --h, --H

<输出帮助文档>

输入如下图

```
C:\Users\l\Desktop\桌面内容\codeCommitTest>java -jar bianbian_codeCommit.jar --h
--help, --h, --H          <输出帮助文档>

代码提交功能：
如果需要提交代码请按下面步骤先执行：
1、请先执行一次 “java -jar bianbian_codeCommit.jar --codecommit_bininit” 初始化bin文件
2、然后将“git-bash.exe”安装路径修改配置到 “./bin/Git安装目录_写在这里.txt” 文件中

--codecommit_bininit
    初始化提交代码需要的 bin 文件

--codecommit
    填写好配置信息后进行代码提交

--createPatchTag
    自动对比打Patch，每笔提交单独Patch，生成的patch及配置文件可直接使用命令 “--codecommit” 进行代码提交
    使用方法：“命令样例：--createpatch D:/xx/xx/xx/xx nativebranch remotebranch”
    使用方法：“参数解释：--createpatch 代码位置目录 本地分支名称 远程分支名称”

--opendes
    打开 提交代码 需要配置的详细说明文档，或者打开路径 “./bin/xxx” 查看

1功能开发中：
    说明____

2功能开发中：
    说明____
```

--codecommit_bininit 初始化提交代码需要的 bin 文件

需要使用到电脑安装的 gitbash ，通过java代码调用 shell 脚本实现生成patch或提交代码

名称	修改日期	类型	大小
bin	2023/12/7 2:18	文件夹	
codeTmp	2023/12/7 2:14	文件夹	
config	2023/6/27 2:06	文件夹	
createPatch	2023/7/1 1:13	文件夹	
bianbian_codeCommit.jar	2023/12/7 2:56	WinRAR 压缩文...	25 KB
result.txt	2023/7/1 3:18	TXT 文件	1 KB

名称	修改日期	类型	大小
applyPatch.sh	2023/12/7 2:34	SH 文件	1 KB
cloneCode.sh	2023/12/7 2:34	SH 文件	2 KB
createPatch.sh	2023/12/7 2:34	SH 文件	1 KB
getCommitMsg.sh	2023/12/7 2:34	SH 文件	1 KB
Git安装目录_写在这里.txt	2023/12/7 2:34	TXT 文件	1 KB

同时需要将 电脑上 安装的 git 配置到文件 “Git安装目录_写在这里.txt” 中

--createpatch 自动生成patch


--createpatch D:/xx/xx/xx/xx nativebranch remotebranch

--createpatch 本地代码仓目录 本地分支名称 远程分支名称(不带remote)

生成的patch会在 “createPatch” 目录

名称	修改日期	类型	大小
bin	2023/12/7 2:18	文件夹	
codeTmp	2023/12/7 2:14	文件夹	
config	2023/6/27 2:06	文件夹	
createPatch	2023/7/1 1:13	文件夹	
bianbian_codeCommit.jar	2023/12/7 2:56	WinRAR 压缩文...	25 KB
result.txt	2023/7/1 3:18	TXT 文件	1 KB

桌面内容 > codeCommitTest > createPatch

名称	修改日期	类型	大小
 JZEasyAndroid	2023/7/1 1:13	文件夹	

在此目录下，会以代码仓 目录名称 内生成对应 patch 文件，和对应的提交信息

名称	修改日期	类型	大小
0001-readMe.patch	2023/7/1 1:13	PATCH 文件	1 KB
0002-2.patch	2023/7/1 1:13	PATCH 文件	1 KB
0003-3.patch	2023/7/1 1:13	PATCH 文件	1 KB
0004-.patch	2023/7/1 1:13	PATCH 文件	1 KB
commitFormatMsg.txt	2023/7/1 1:13	TXT 文件	1 KB
commitMsg.txt	2023/7/1 1:13	TXT 文件	1 KB

其中 "commitFormatMsg.txt" 中的内容为如下

```
第0条
create patch 1

第1条
create patch 2
```

其中 “commitMsg.txt” 中内容 为 本地已 commit 未提交的 msg 信息，供查看

```
commit 9345b78e9ee6ded5e73d76fd2e888f6b6aedb384
Author: bian <email@Str>
Date: Thu Dec 7 01:31:50 2023 +0800

    create patch 2

commit 7e62240c718728a843bccebfe82e2e47aec0c160
Author: bian <email@Str>
Date: Thu Dec 7 01:31:15 2023 +0800

    create patch 1
```

--codecommit

填写好配置信息后进行代码提交

需要在config目录下配置对应需要提交的代码仓信息，如无此目录，请新建 “config”

名称	修改日期	类型	大小
bin	2023/12/7 2:18	文件夹	
codeTmp	2023/12/7 2:14	文件夹	
config	2023/6/27 2:06	文件夹	
createPatch	2023/7/1 1:13	文件夹	
bianbian_codeCommit.jar	2023/12/7 2:56	WinRAR 压缩文...	25 KB
result.txt	2023/7/1 3:18	TXT 文件	1 KB

桌面内容 > codeCommitTest > config

名称	修改日期	类型	大小
project_1	2023/7/1 1:14	文件夹	
project_MyApplication	2023/7/1 2:13	文件夹	
如果需要提交多个代码仓, 新建.....txt	2023/6/27 0:33	TXT 文件	1 KB

如上图，对应提交的代码仓，文件夹目录以“**project_**”为开头，后面跟的是对应的工程名称；工具Jar 识别的是“**project_**”开头的目录字符串，每有一个对应文件夹，视为 一个仓需要提交；待提交的代码仓生成的 patch 放入对应工程目录下：

桌面内容 > codeCommitTest > config > project_MyApplication

名称	修改日期	类型	大小
0001-readMe.patch	2023/7/1 1:13	PATCH 文件	1 KB
0002-2.patch	2023/7/1 1:13	PATCH 文件	1 KB
0003-3.patch	2023/7/1 1:13	PATCH 文件	1 KB
0004-.patch	2023/7/1 1:13	PATCH 文件	1 KB
config.txt	2023/6/29 1:19	TXT 文件	1 KB
patchMsgs.txt	2023/7/1 1:55	TXT 文件	1 KB

“**config.txt**” 文件为对应需要提交的 代码仓的配置，如下图

```
ssh = https://[redacted]@[redacted]:[redacted].git
projectName = JZEasyAndroid
branch = dev
committer = commiterValue
reviewer = reviewer
```

“**patchMsgs.txt**” 为对应需要提交的 需要提交的 patch 信息

```
--itemStart--
dtsOrAR = dtsOrAR1
msg = msg1
--itemEnd--

--itemStart--
dtsOrAR = dtsOrAR2
msg = msg2
--itemEnd--

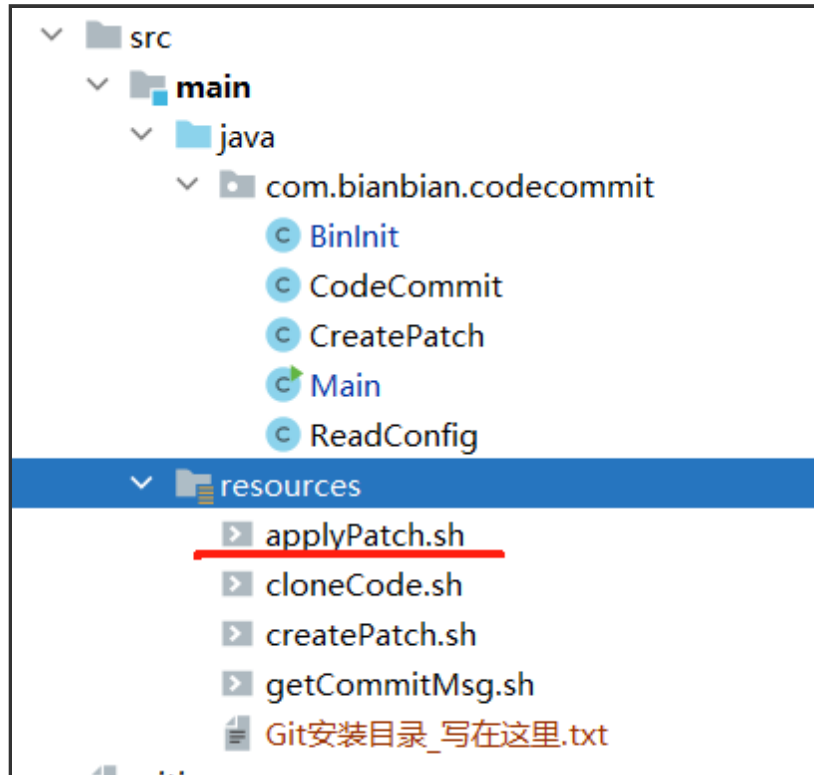
--itemStart--
dtsOrAR = dtsOrAR3
msg = msg3
--itemEnd--

--itemStart--
dtsOrAR = dtsOrAR4
msg = msg4
--itemEnd--
```

--codecommit 使用其他说明（需要自己做一下实现）

代码提交功能在最后一步提交 git push 未实现，以为此功能用作在 黄区等，自己无代码提交权限，需要对应田主等进行提交的情况，需要自己将最后一步完成：

1、打开源码文件：applyPatch.sh



```

echo $dtsOrAR >> $1/../clonelog1.txt
echo $msg >> $1/../clonelog1.txt

cd $codePath

echo " 应用patch: " >>../clonelog1.txt
git apply $patchPath >>../clonelog1.txt

echo "  add全部: " >>../clonelog1.txt
git add . >>../clonelog1.txt

echo " 提交: " >>../clonelog1.txt
git commit -m "
[🔥tsOrAR]:$dtsOrAR
[msg]:msg"

```

根据公司自己公司代码提交规范，修改 commit 信息提交；
及添加 git push 或者 git mr 等操作；

2、修改源码 JAVA 文件中的实现：CodeCommit.java

```

68      LogUtils.info(TAG, logContent: "提交完毕");
69      try {
70          /* result */
71          /* 将成功的mr 汇总 放入 最开头 */
72          /* 文件中编写一共提交的代码仓库情况 */
73          /* 输出每个 代码仓库 提交最后的日志*/
74          String path = FileUtil.getRunningPath() + File.separator + "result.txt";
75          LogUtils.info(TAG, logContent: "提交完毕: " + path);
76          LogUtils.printList( tag1: "", tag2: "", FileUtil.readTextFileLineToListStr(path));
77          Process p = Runtime.getRuntime().exec( command: "notepad " + path); //调用系统的记事本
78      } catch (Exception a) {
79          System.out.println("Error exec notepad");
80      }
81  }

```

这里自己实现后会漂亮点，不做实现 仅做了上面的第一步，也不影响代码提交功能的使用