## 1. 前言

目前大部分公司都在使用Git作为版本控制，每个程序员每天都要进行代码的提交。很多开发者也包括我自己，有时候赶时间或者图省事，就这么提交：

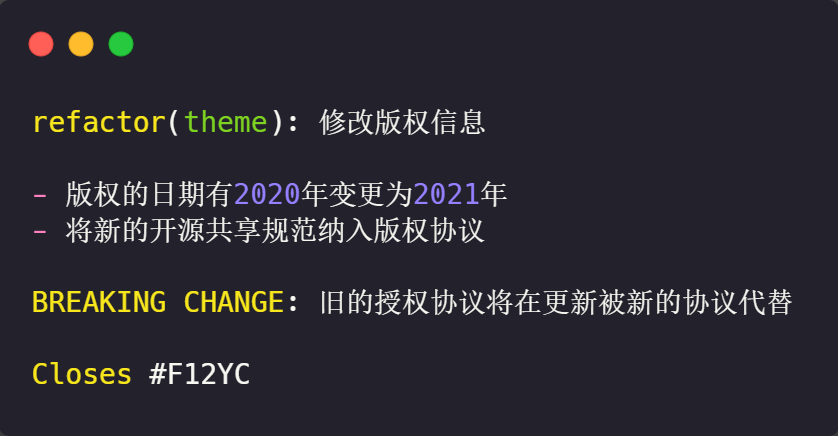
git commit -m "修改bug,优化代码"复制代码

过了一段，突然去查找一个具体的提交你会发现不是特别好找。因此我们需要规范我们的代码提交来避免这种情况。同时良好的提交规范也有助于我们生成清晰的ChangeLog，更利于同事之间的协作。

如果你想成为知名开源项目的贡献者更要规范自己的代码提交。

## 2. Git提交规范

目前业内做的比较好的，比较具有参考价值的就是知名前端框架**AngularJS**的提交规范。我们先来看一个例子：



对应的格式：

<type>[optional scope]: <description># 空行[optional body]# 空行[optional footer]复制代码

更严格的项目可能提交要求使用英文描述，特别是国际化的开源项目。

根据上面这个例子我们来了解一下这个业界比较认可的Git提交规范。

### type

refactor 表示本次提交的是**重构**代码，也就是它是一个提交的类型type，除了refactor还有：

* feat 新功能，顾名思义就是新需求的实现。
* fix 修复，就是对bug的修复。
* docs 文档，主要用来描述文档的变更。
* style 主要是代码风格相关的提交，比如格式化等。
* refactor 重构代码，对已有功能的重构，但是区别于bugfix。
* test 测试相关的提交，不太常用。
* chore 构建过程或辅助工具的变动，不太常用，比如之前用Maven，后面换成了Gradle。

每次提交声明提交的type是必须的，它让本次提交的作用一目了然。

### scope（可选）

用来表明本次提交影响的范围，方便快速定位。你可以写明影响的是哪个模块（通常是模块名称）或者是哪个层（数据层、服务层、还是视图层）。

### subject

就是上面的修改版权信息，是对本次提交的简短描述概括。就像胖哥写文章要起一个标题一样，不要过长。

### body（可选）

就是比较详细描述本次提交涉及的条目，罗列代码功能，这里胖哥习惯用**markdown**的列表语法，也就是用中划线换行隔开条目。当然body不是必选的，如果subject能够描述清楚的话。

### foot（可选）

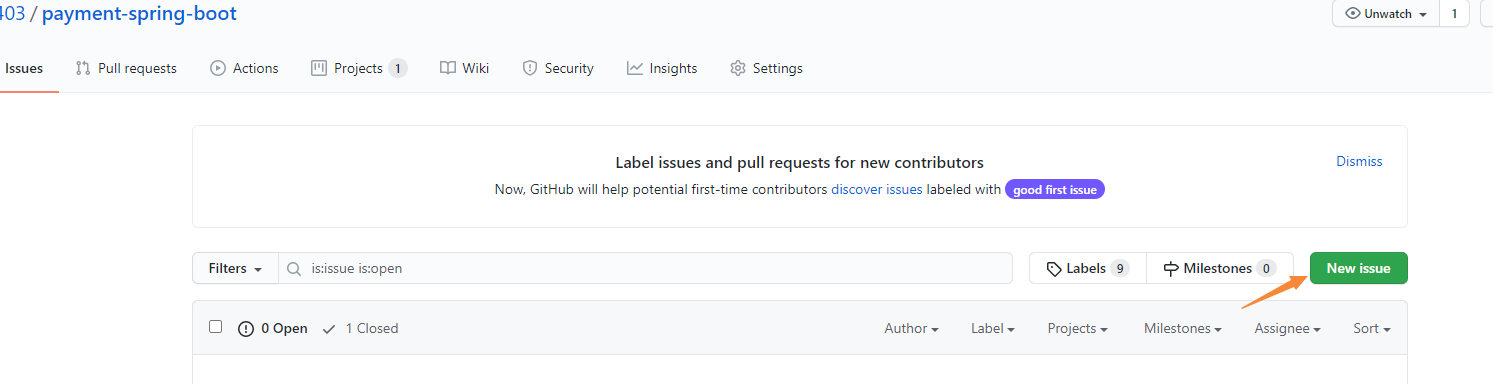
描述与本次提交相关联的**break change**或**issue** 。

#### break change

指明本次提交是否产生了破坏性修改，类似版本升级、接口参数减少、接口删除、迁移等。如果产生了上述的影响强烈建议在提交信息中写明**break change**，有利于出问题时快速定位，回滚，复盘。

#### issue

如果发现项目有bug、或者有优化的建议、甚至新增一个任务，就可以利用**issue**给项目提交一个任务。



**issue**不是一些Git平台的专属功能，JIRA等平台也有类似功能，它们的作用大同小异，都可以很好地反应项目的成长状况和参与度。那么在Git提交时，我们可以在**foot**区域关联本次提交涉及的**issue**。

# 涉及

issues #F12YC,#F45JW

# 关闭

Closes #F12YC复制代码

这里没有固定格式，不过尽量去参考一些知名项目去做。

## 3. 工具安利

说了这么多，相信你已经对Git提交的规范有所了解了。这里推荐一些有用的工具来帮助你将这些规范落实到位。在**Intellij IDEA**的插件市场有很多Git Commit Message模板插件，可以可视化的实现这些规范。

