# **【敏捷开发】使用Reviewboard进行代码Review**

作者：[韩宇飞](http://gad.qq.com/user/index?id=2044697)

链接：<http://gad.qq.com/article/detail/286013>

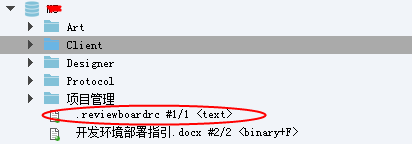
**本文首发于知乎专栏：[MACK的游戏开发笔记](https://www.zhihu.com/people/han-yu-25-35/activities)，欢迎各位关注。**

**前言**  
作为敏捷开发的一系列重要实践，单元测试、持续集成、CodeReview……CodeReview是我们日常开发中的一个重要环节。理想状况下每次提交会必须先经过Review，通过之后才能提交。提交之后系统会自动构建并执行自动化测试，来保证持续集成以及版本的稳定性。  
  
目前我们在项目中使用P4做为版本管理工具，使用Jenkins进行自动构建，使用Reviewboard管理CodeReview。Review使用Pre-commit  
Review的方式，每次开发提交必须先自测，然后通过P4插件提交Review Quset，等组员都Review通过之后再进行提交。  
  
**为什么要CodeReview**  
个人感觉Review对一个项目开发特别是多人的大型项目开发来说还是必不可少的，虽然很多人会说Review会降低开发效率。磨刀不误砍柴工，虽然开发前期的时间好像变长了也更繁琐了，但是能让代码写的更扎实以后的问题更少，降低技术债务，总体来说还是节省时间的，并且对整个项目成员的成长也是有极大的好处的。以下总结了一些CodeReview的好处：

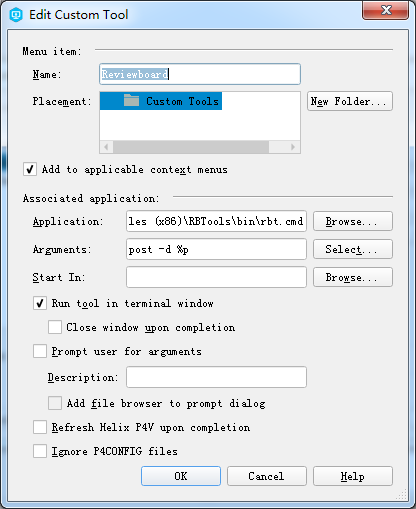
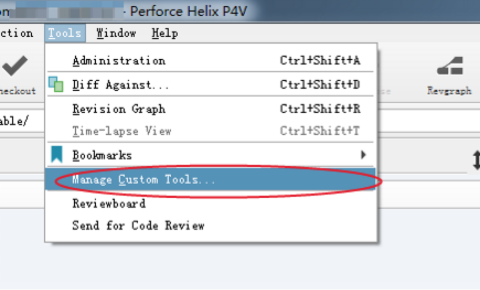
* 版本的稳定性更强。相比直接提交，提交者会更小心，检查者也会多一层把关，更容易减少Bug的产生。一个Bug的修复时间是随着Bug的发现时间指数增长的，发现的越慢需要修复的成本越高，产生的问题约严重。
* 代码的设计更好。通过Review开发者会更了解整个项目，Review的时候各个模块的同学也势必会互相讨论，这样一些设计会更全面兼容性更强，也更不容易出现重复冗余的问题。
* 代码质量更好。因为需要被Review，代码通常会写的更符合代码规范，注释更清晰，变量名函数名起的更好。
* 组员的交流更强。因为阅读他人代码讲解自己的代码，每个成员都需要沟通和讨论。这样一方面能大幅提升组员的表达总结能力，另一方面也能提升项目组的氛围，而不是每天默默开发相互都不了解。当有任务变动的时候也更容易交叉替代，让开发变的更灵活。
* 个人能力提升更强。对程序员来说，很大的一个问题就是个人得不到成长，始终做一块的内容，经常做重复性劳动。通过Review每个人都可以了解更多的内容，能力得到全面提升。资深的能力强的同学可以指导新同学，新同学也可以通过Review资深同学的代码更好的学习。

**如何使用Reviewboard的部署和配置**

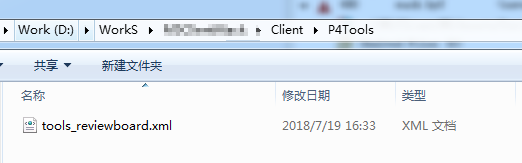
* 首先下载并安装RBTools-1.0.exe和diffutils-2.8.7-1.exe(用来生产Diff文件)。
* 把diff加到PATH里，一般是这个路径"C:\Program  
  Files (x86)\GnuWin32"，如果报diff找不到需要重启P4。
* 更新配置文件，在P4根目录：



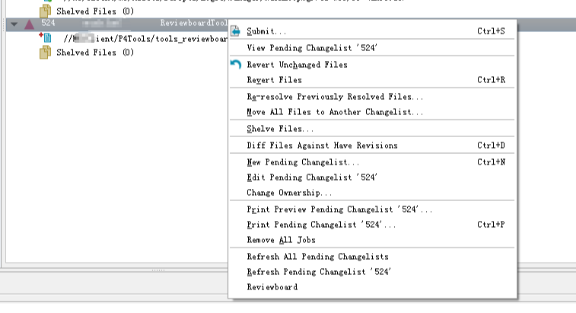
* 添加P4插件：



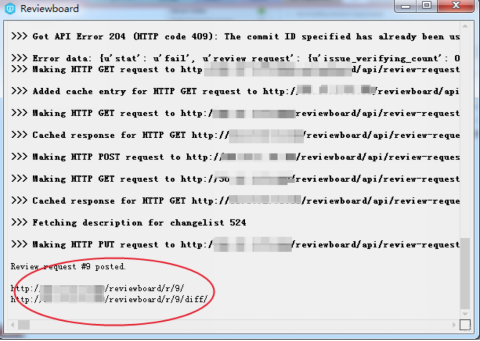
或者在目录Client/P4Tools目录下直接导入工具：



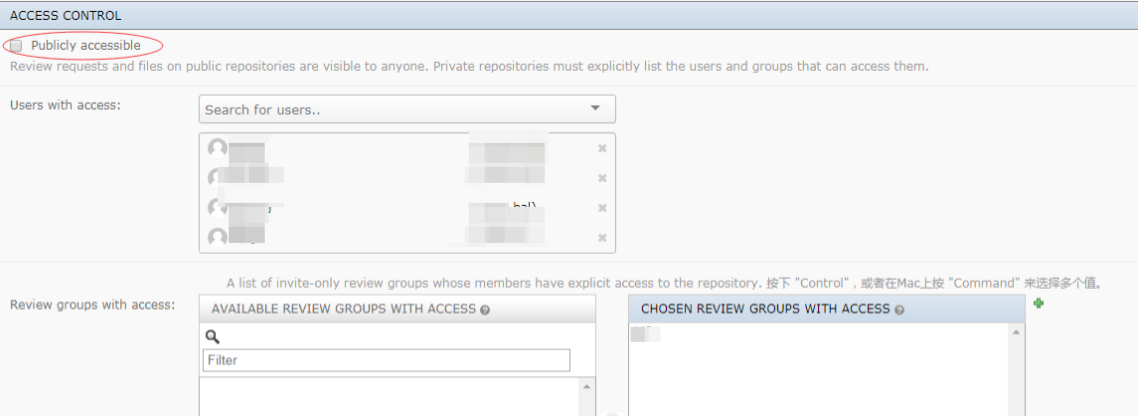
* 在P4中创建一条Pending，例如Reviewboard测试，然后在Pending上右键选择我们添加的工具，Reviewboard：



* 如果成功了会有输出Review  
  request #6 posted的提示。如果文件没有修改，或者这条Pending已经有了会提示相应的错误信息。如果成功还会输出两个路劲，一个是这次review  
  request的地址，一个是diff的地址，例如：



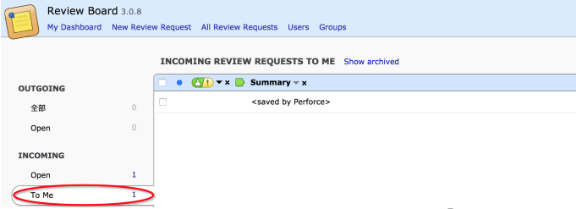
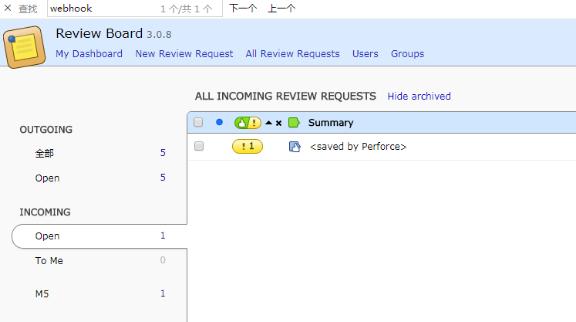
* 这里会出现登陆错误的问题，需要关闭Publicly  
  accessible指定用户，这里有些奇怪：



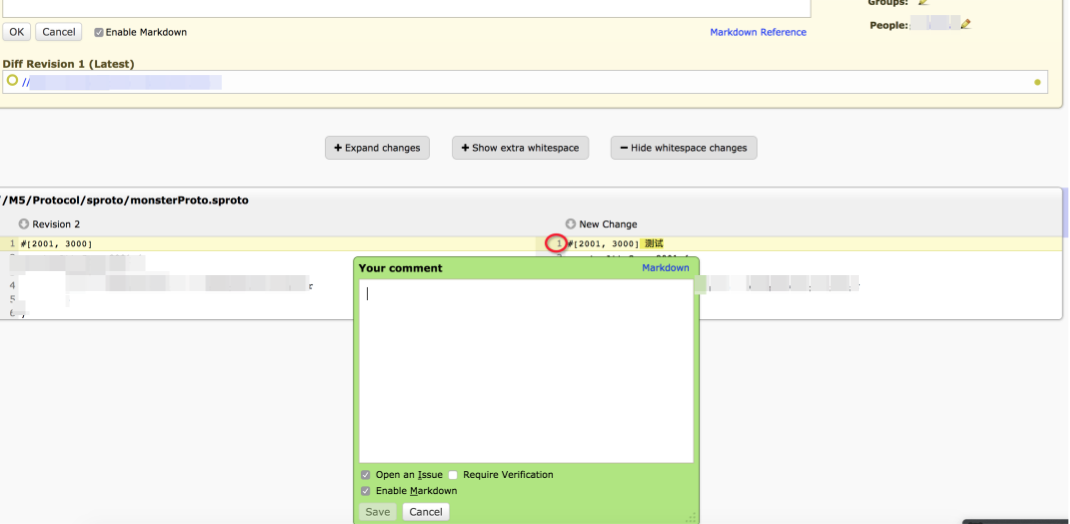
* 填写这条Quest的相关信息，例如描述，内容，以及谁来review这三个地方，谁来review可以选择某个人也可以选择一个组，这一步可以在.reviewboardrc文件中设置成自动设置，不需要人工设置，也可以写脚本控制：
* 
* 当填写完毕并自测之后就可以点击Publish发布，提交之后会自动发送邮件通知：



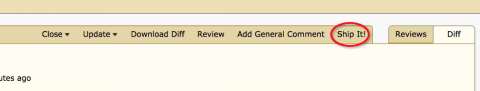
* 在主面板就可以看到OUTGOING自己提交但未发布的，INCOMING，OPEN别人审核过的，ToMe别人发布给我审核的：



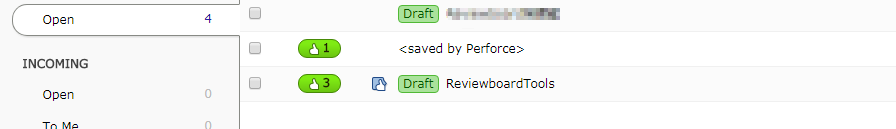
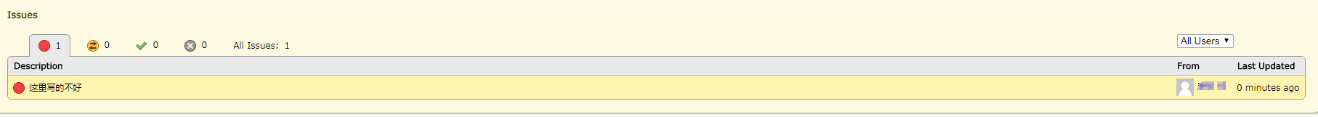
然后针对diff可以点前面的ID写备注，对代码的修改意见。



* 然后ShipIt。如果有修改意见则这条quest不会通过，会自动转给开发者，并发送邮件：



* 如果不通过提交方会受到一条不通过的提示，否则可以看到已经通过，如果都通过即可提交了。这一步需要自己开发插件，在提交的时候自动检查是否通过，只有通过才能上传：



以上就是reviewboard和P4的大致集成流程。不过在实际使用过程中也有一些需要讨论的点。首先就是虽然整个流程是大部分是自动的，但是实际在review的时候只靠看diff文件的差异还是很难review出问题的。没有上下文看的人可能不明白，只能review出一下代码规范，保护等错误。在这上面我们最终还是灵活一些，对于大功能，新开发的系统还是当面详细review，对迭代修复bug的内容才会直接通过diff文件review。另外就是如果每次提交都会产生ReviewQuest触发邮件通知，并且不通过就不能提交的话，可能会影响进度以及打断其他人的工作。这点上我们是通过限制Review的成员，增加随机性即可以保证每个人都能相互review有不会每个人的提交让所有人都review来解决，另外也可以通过限制提交时间的方式缓解。此外还有其他一些问题，这里就不进行深入讨论了。