代码审查

目录

典型代码审查工作流程
什么是代码审查

如何审查代码
如何接受代码审查
延伸阅读

代码审查是 WP-CLI
项目软件开发流程的核心环节。通过代码审查与结对编程，我们既能保障产出质量，也能在工作中形成统一的
代码风格。清晰一致的代码风格能让新老开发者都更轻松地理解和维护项目。若执行得当，代码审查还能成为
所有参与者的学习过程。以下是 典型的代码审查工作流程 ：

代码审查的定义
如何进行代码审查
如何接受代码审查

典型代码审查工作流程

我们目前使用 GitHub 处理所有事务。请阅读 我们的 GitHub 工作流程了解流程、命名规范、议题和拉取请求的使用细节。一个中等复杂度的议题通常通过多个拉取请求解决，每个
拉取请求处理该议题的不同部分。这使得审查更简单，因为每个审查阶段只需查看少量代码。

每次提交推送到 GitHub 分支时，我们的 CI
自动化测试（所有代码的语法检查、功能代码的单元测试，理想情况下还包括行为和自动化验收测试
）会在 Travis 上运行。
如果构建通过，拉取请求即可接受审查。如果未通过，原始开发人员负责修复构建直至通过。
当原始开发人员对其工作满意时，可通过分配 @wp-cli/committers 来请求 提交者团队 进行审查。
简单的拉取请求通常可由审查它们的开发人员直接合并。更复杂的变更集通常需要审查者与开发人员
之间反复讨论，并应有次级审查者参与。
GitHub 的“文件更改 ”
选项卡适合对变更集的特定部分添加行内评论。更一般的评论可留在拉取请求的 “对话 ”选项卡中。审查者可通过从被审查分支创建拉取请求或发表评论的形式提出修改建议。

- 开发人员将实施建议的更改，为澄清问题展开讨论，并在对其工作满意时提及审查者。
- 如果拉取请求在合并前需要最终清理或已被放弃，审查者可直接提交到该分支。但应避免未经协商重写代码。
- 当审查者对更改满意时，他们可以合并拉取请求，或将其分配给次级审查者进行合并。原始开发人员（理想情况下包括审查者）应在合并后数日内保持待命，以解决可能出现的任何问题。

什么是代码审查

理解代码审查目标的一个好方法是参考这个类比，它对应马斯洛的“需求层次”金字塔。从最基础到最高层级，审查者会检查代码是否正确、安全、可读、优雅以及利他。保持这种优先级意识很重要：如果变更引入了未处理的边界情况、错误或安全漏洞，那么在考虑编码风格指南或美化实践偏好之前，必须先解决这些问题。

code-review-pyramid

如何审查代码

作为审查者，你的首要任务是理解提议的变更做了什么以及为何必要。在理解其功能、必要性及构建方式背后的决策之前，任何批评都毫无意义。

接着，检查其正确性。所有接收参数并产生输出的函数是否都有功能单元测试覆盖？它们是否确实完成了预期任务？你能设想出函数会意外出错或返回非预期结果的边界情况吗？如果该命令有输出，它在所有情况下都能正确呈现吗？所有全局标志都处理了吗？对于不常见的情况是否有合理的回退机制？错误处理是否提供了清晰的输出信息？

如果它满足了业务和用户体验需求，下一步要检查的是安全性。是否对所有不可信输入遵循了基本的清理和转义实践？如果它与代码库的其他部分交互，是否对输入宽容而对输出严格，确保只传递预期值？像攻击者一样思考。如果恶意行为者有可能利用此代码，或不幸的用户可能触发导致崩溃或不良表现的错误，你的职责就是发现它。

接下来，检查可读性。函数、变量和文件的命名应根据其含义清晰明了。所有内容都应遵循周围的代码风格。

> 任何代码库中的所有代码都应看起来像是同一个人编写的，无论有多少人贡献过。
>
> -- 编写地道 JavaScript 的原则

由于可读性本质上是主观的，这需要能够从眼前的代码变更回溯到更大的图景。设想几个月后有人试图通过当前变更集追踪特定代码路径。是否存在可以简化的不必要步骤？代码注释和内联文档是否足够详尽，能够重现代码背后的思考过程？

最后，检查优雅性和整体质量。代码应遵循现有且已知的模式，以便他人一目了然。如果变更引入了重构周边功能、将旧代码抽象并标准化为新模式的机会，请提出这些建议。

如何接受代码审查

作为接受代码审查的一方
，你的职责是从建议中学习。防御性、固执或不耐烦会阻碍你充分吸收建议。如果你觉得审查者误解了你的意
图，可以适当解释。但争论语义或坚持某种方法的正确性几乎总是不良习惯，会分散团队协作的注意力。如果
今天熟悉代码库的审查者都看不明白，那么六个月后新加入的开发者肯定更无法理解。

最好尽快回应审查者提出的问题。请记住，审查过程需要审查者和你自己进行上下文切换，这个过程越及时，
上下文切换带来的干扰就越小。

延伸阅读

Glen Sanford：论代码审查 - "待处理的代码审查代表着被阻塞的执行线程 [,
代码审查应始终是你的首要任务]"

无我编程十诫 - "理解并接受你会犯错。关键在于在它们进入生产环境之前尽早发现它们。 "

版本号 #2
由 壳壳虫 创建于 2026-03-26 17:08:46 UTC
由 壳壳虫 更新于 2026-03-26 17:08:46 UTC