

# 如何做好代码审查

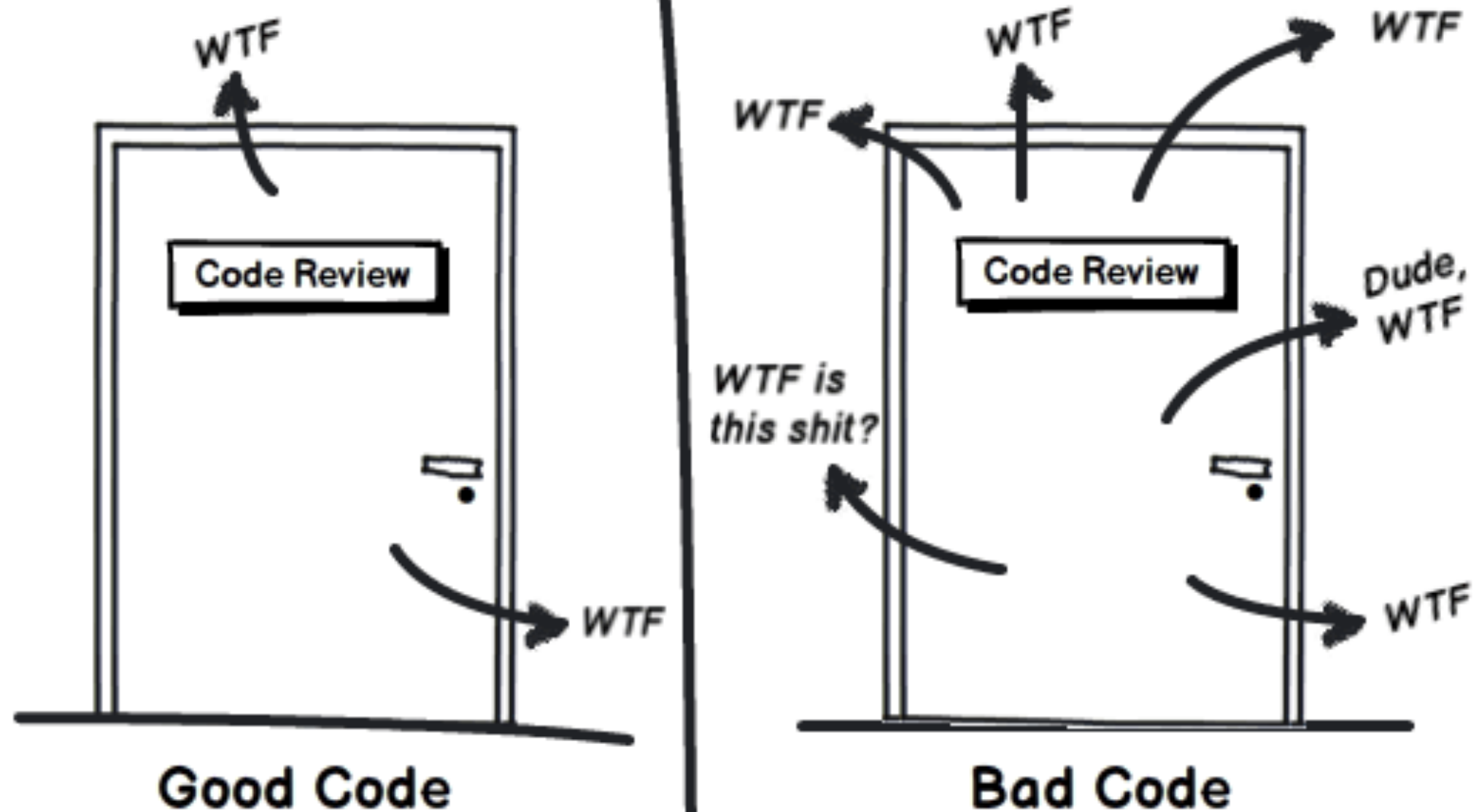
@dylanninin

@2016-03-28

**维基百科**

**系统性地审查源代码**

# Code Quality Measurement: WTFs/Minute



# 代码审查有哪些好处

# 代码审查有哪些好处

**提早发现设计、实现错误**

# 示例

## OpenPlay 1.0 角色/权限控制

- 只有领队可以邀请、剔除、编辑成员
- 只有领队可以编辑球队信息
- ...

传送门:

- <http://bit.ly/1TfZFpV>

# 示例

## OpenPlay 1.0 邀请和通知

- 各种邀请接口
- 带交互通知的处理
- ...

传送门:

- <http://bit.ly/1Tk7zOX>

# 代码审查有哪些好处

增进开发者的责任心、羞耻心



# 示例

球员版 更新用户信息接口

- if else 嵌套层次
- 代码的形状

传送门:

- <http://bit.ly/1VUBI8f>

# 示例

## 球员版 Rating 开关

- 开：球员版展示 rating/motm
- 关：球员版不展示 rating/motm
- ...

## 传送门：

- <http://bit.ly/1UE2CCj>
- <http://bit.ly/1PGTPaa>

# 代码审查有哪些好处

分享最佳实践，促进共同成长

# 示例

## Cub 接入 Elasticsearch

- 搜索：各类搜索特性
- 模型：建立数据模型
- 索引：如何同步索引
- ...

## 传送门：

- <http://bit.ly/1M25XI1>
- <http://bit.ly/1M25wxz>

# 代码审查有哪些好处

**共享信息，避免单点故障**

# 示例

Redis pubsub

- 生产者/消费者模型
- 实现的改进: pubsub, list
- 订阅者维护

传送门:

- <http://bit.ly/1M26wSg>

# 代码审查有哪些好处

长期投资、快速迭代

# 示例

Admin、Stats

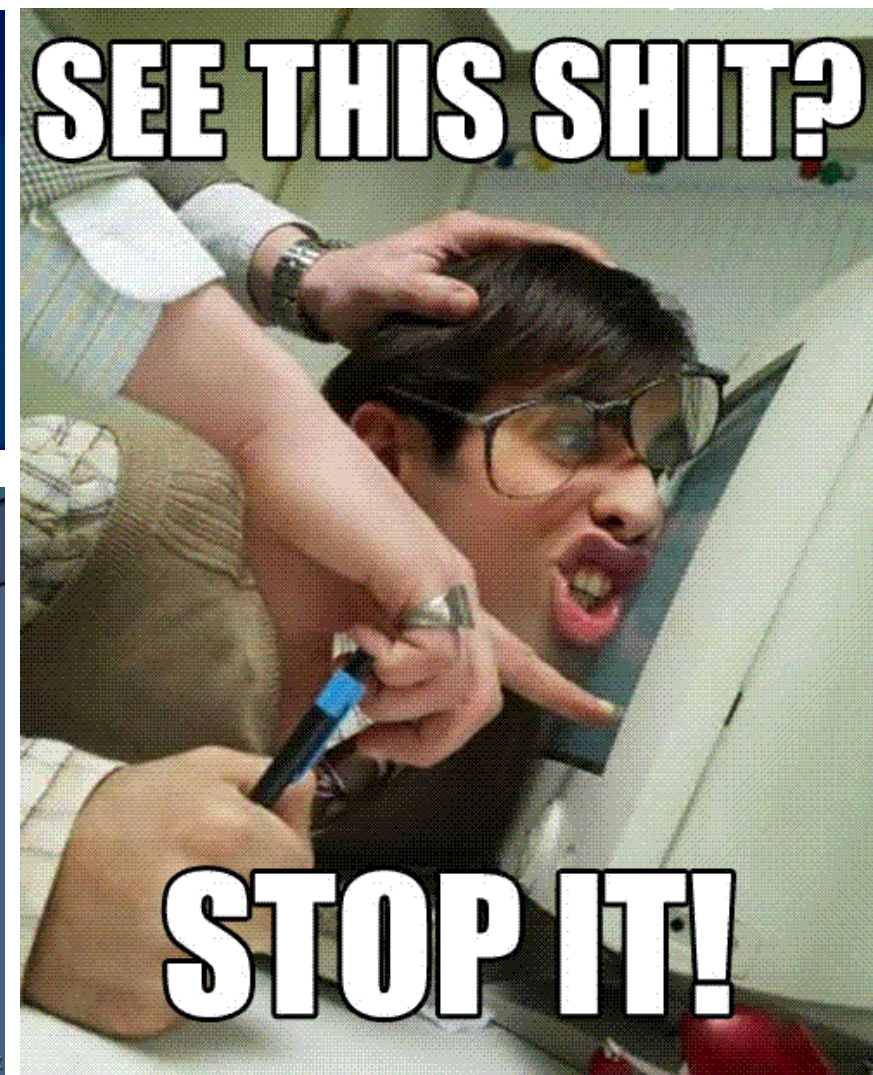
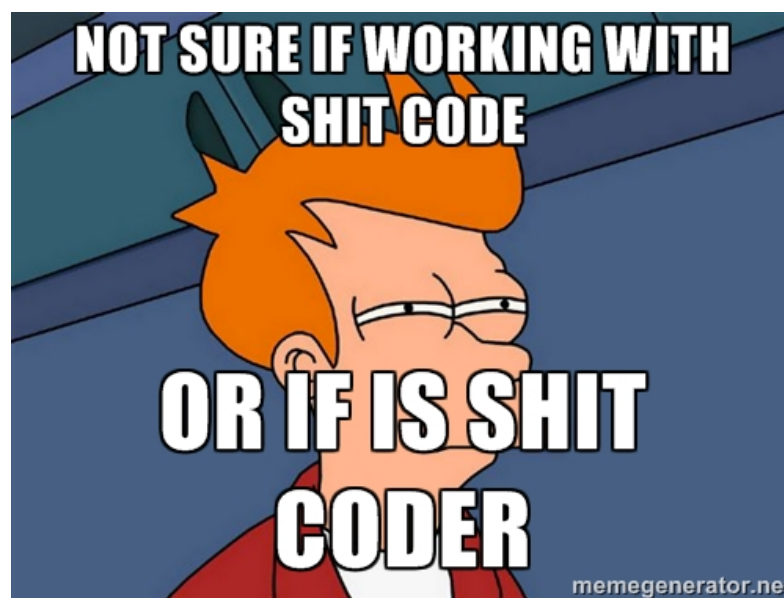
- 搜索接口： 1.x (MongoDB) —> 2.x (ElasticSearch)
- 文件下载： DownloadMixin
- 事件的实时统计： crontab -> queue -> queue/msg

传送门：

- <http://bit.ly/1VUBrSM>
- <http://bit.ly/1PGTPaa>










**代码审查有没有坏处？**



# 代码的坏味道

# Bristol Stool Chart

Type 1		Separate hard lumps, like nuts (hard to pass)
Type 2		Sausage-shaped but lumpy
Type 3		Like a sausage but with cracks on its surface
Type 4		Like a sausage or snake, smooth and soft
Type 5		Soft blobs with clear-cut edges (passed easily)
Type 6		Fluffy pieces with ragged edges, a mushy stool
Type 7		Watery, no solid pieces. <b>Entirely Liquid</b>

**什么才是优秀的代码？**

**代码级别**

# 代码级别

- 可编译
- 可运行
- 可测试
- 可读
- 可维护
- 可重用

# Python 之禅



```
>>> import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than \*right\* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!

优美胜于丑陋（Python 以编写优美的代码为目标）

明了胜于晦涩（优美的代码应当是明了的，命名规范，风格相似）

简洁胜于复杂（优美的代码应当是简洁的，不要有复杂的内部实现）

复杂胜于凌乱（如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁）

扁平胜于嵌套（优美的代码应当是扁平的，不能有太多的嵌套）

间隔胜于紧凑（优美的代码有适当的间隔，不要奢望一行代码解决问题）

可读性很重要（优美的代码是可读的）

即便假借特例的实用性之名，也不可违背这些规则（这些规则至高无上）

不要包容所有错误，除非你确定需要这样做（精准地捕获异常，不写 `except:pass` 风格的代码）

当存在多种可能，不要尝试去猜测

而是尽量找一种，最好是唯一一种明显的解决方案（如果不确定，就用穷举法）

虽然这并不容易，因为你不是 Python 之父（这里的 Dutch 是指 Guido）

做也许好过不做，但不假思索就动手还不如不做（动手之前要细思量）

如果你无法向人描述你的方案，那肯定不是一个好方案；反之亦然（方案测评标准）

命名空间是一种绝妙的理念，我们应当多加利用（倡导与号召）

# 心得体会

**Hello World**



**KEEP  
CALM  
AND**

**`System.out.println`  
`("Hello World")`**

# 搭建基本框架

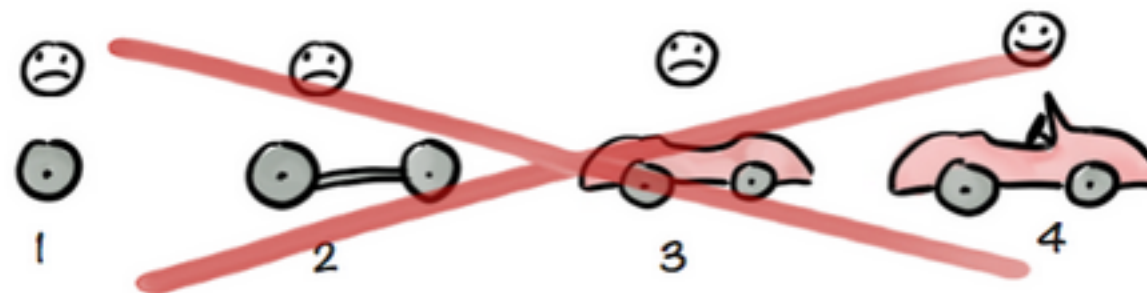




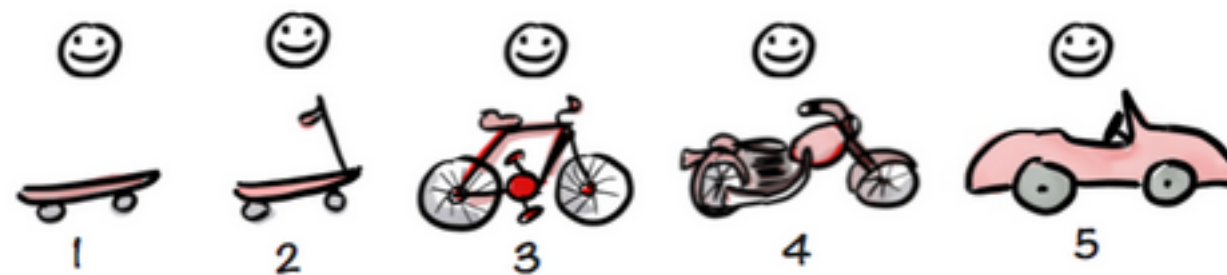
# 迭代式开发



Not like this....



Like this!



**从现在开始做代码审查**

- 少而精：关注某个特性，使审查过程愉快
- 审查清单：定制适合的清单，有目的地审查
- 使用专业的工具：GitHub + Pull Request + Issue
- 人人参与，相互审查：人都会犯错，多视角剖析
- 及时反馈，及时合并：使审查结果落地，而非走秀
- 学会享受：保持积极正面的态度

# 代码审查清单

# 代码审查清单 - 常规

- 代码能够工作么？
- 所有的代码是否简单易懂？
- 代码符合你所遵循的编程规范么？
- 是否存在多余的或是重复的代码？
- 代码是否尽可能的模块化了？
- 是否有可以被替换的全局变量？
- 是否有被注释掉的代码？
- 循环是否设置了长度和正确的终止条件？
- 是否有可以被库函数替代的代码？
- 是否有可以删除的日志或调试代码？

# 代码审查清单 - 安全

- 所有的数据输入是否都进行了检查并且进行了编码？
- 在哪里使用了第三方工具，返回的错误是否被捕获？
- 输出的值是否进行了检查并且编码？
- 无效的参数值是否能够处理？

# 代码审查清单 - 文档

- 是否有注释，并且描述了代码的意图？
- 所有的函数都有注释吗？
- 对非常规行为和边界情况处理是否有描述？
- 第三方库的使用和函数是否有文档？
- 数据结构和计量单位是否进行了解释？
- 是否有未完成的代码？
- 是否用合适的标记进行标记比如‘TODO’？

# 代码审查清单 - 测试

- 代码是否可以测试？
- 是否存在测试，它们是否可以被理解？
- 单元测试是否真正的测试了代码可以完成预期的功能？
- 是否检查了数组的“越界”错误？
- 是否有可以被已经存在的API所替代的测试代码？



# 代码审查清单 - 优化

- 优化清单
- 得到团队认可
- 保持更新

**Just Do It**

**Q & A**

# Reference

- [https://en.wikipedia.org/wiki/Code\\_review](https://en.wikipedia.org/wiki/Code_review)
- [从 Code Review 谈如何做技术](#)
- [What to look for in a Code Review](#)
- [Stop More Bugs with our Code Review Checklist](#)
- [code-review-best-practices](#)