祝兴志 2019-03-01

**软件代码正式测试前必须按照本指南的规定实行代码审查，以提高软件代码质量**

北京德辰科技股份有限公司

C#开发人员代码审查指南V1.0

C#开发人员代码审查指南

# 概述

## 制定原则

这是为C#开发人员制定的代码审查指南，该指南仅用于编程时的参考，可以确保编程时能考虑到绝大多数的编码规则。尤其是对刚入门和开发经验不足（0-3年）的开发人员而言，参考该审查清单有助于养成良好的编码习惯；该指南对经验丰富，但编程习惯不够严谨的开发人员，同样有极大帮助，有助于逐步纠正不好的编码及思维习惯，编写出高质量的软件代码。

本指南的制定，以不影响工作效率，快捷简便的实现代码审查，提高软件代码质量为基本原则。**正式提交测试的代码，务必实施代码审查，至少两人参与，并填写《C#代码审查表》**。**《C#代码审查表》文件名称为：代码审查表-项目名称-审查人员全名-YYYYMMDD**

文档出处：[http://www.codeproject.com/Articles/593751/Code-Review-Checklist-and-Guidelines-forCsharp-De](http://www.codeproject.com/Articles/593751/Code-Review-Checklist-and-Guidelines-for-Csharp-De)

## 术语定义

### Pascal 命名规则

将标识符的首字母和后面连接的每个单词的首字母都大写。可以对三字符或更多字符的标识符使用Pascal 大小写。例如：BackColor**对属性或公共变量的定义，请采用Pascal命名规则**

### Camel 命名规则

标识符的首字母小写，而每个后面连接的单词的首字母都大写。例如：backColor**用于对局部变量、方法参数的定义**

### \_Camel命名规则

用这种风格用于表示类private或protected成员变量。例如：\_backColor，常**用于对成员变量的定义**

# 检查表清单

* **系统方案**

1. 了解你所参与的工程的设计方案和框架流程以更好地理解要开发的程序流程；
2. 避免直接从其他来源复制/粘贴代码。即使您是从某些源代码引用代码，也始终建议手工编写代码。通过这种方式，您将获得编写代码的良好实践，并且您将了解该代码的正确用法；最后，您永远不会忘记它；

* **代码编译**

1. 不能在项目中人为设置屏蔽一些严重的警告信息，如果有必要，在代码中进行编写；
2. 确保没有任何警告；
3. 如果工程中有Code Analysis工具辅助分析，可以根据分析结果移除所有的警告；

* **代码规范**

1. **代码可读性**：对代码可读性进行维护，以便其他开发人员能够轻松理解你的代码；

参考: <http://msdn.microsoft.com/en-IN/library/aa291591(v=vs.100).aspx>

1. 在每一个类文件、方法的上面添加注释，并描述清楚期望的输入和返回的输出信息；
2. 始终牢记命名规则，一般情况下，类/方法/属性/公共变量/公共常量，采用Pascal命名规则；成员变量采用\_Camel命名规则；局部变量和参数采用Camel命名规则；

参考: [http://msdn.microsoft.com/en-us/library/ms229043.aspx.](http://msdn.microsoft.com/en-us/library/ms229043.aspx)

1. 确保所有函数的代码行数不太多，最好不超过100行；
2. 检查是否存在无法访问的代码，如果存在，则修改代码；
3. 不管何时都应执行“null”检查，以避免在运行时产生“空引用”异常；
4. 需要移除所有没有用到的using语句，清理无用代码是一个良好的习惯；

参考: [http://msdn.microsoft.com/en-us/magazine/ee335722.aspx.](http://msdn.microsoft.com/en-us/magazine/ee335722.aspx)

1. 数字和字符串常量是否用符号代替；
2. **代码复用**：如果存在代码片段重复使用或代码可能还会在其他地方使用，则应该将该代码提取成一个方法。将执行重复任务的代码提取成方法后放置在功能相关的类中，以供其他开发人员使用；

参考: <http://msdn.microsoft.com/en-us/library/office/aa140806(v=office.10).aspx>

<http://blogs.msdn.com/b/frice/archive/2004/06/11/153709.aspx>

1. **代码一致性**：整个应用程序应保持编码风格一致，例如，.Net中Int32和int及String和string是一样的，在应用程序中应该保持使用一致，避免有时用Int32，有时用int；
2. 如果参数（变量）只使用一次，则应定义为匿名类型；

参考: <http://msdn.microsoft.com/en-us/library/vstudio/bb397696.aspx>

1. 尽量使用LINQ查询和Lambda表达式提高代码可读性；

参考: <http://msdn.microsoft.com/en-us/library/bb308959.aspx>

1. 正确使用var、object和dynamic关键字。它们有一些相似之处，大多数开发人员对它们都有困惑或不太了解，但要注意他们并不能相互替换；

参考:

[http://blogs.msdn.com/b/csharpfaq/archive/2010/01/25/what-is-the-difference-betweendynamic-and-object-keywords.aspx](http://blogs.msdn.com/b/csharpfaq/archive/2010/01/25/what-is-the-difference-between-dynamic-and-object-keywords.aspx)

* **代码性能**

1. 尽可能避免嵌套for、foreach循环语句和if条件语句；
2. **释放非托管资源**：像I/O操作和网络资源等非托管对象一样，使用完后要立即手动释放。如果想要实现离开作用域后自动释放非托管资源，则使用using语法；
3. 检查程序是否有内存泄漏，如有必须及时修改；

参考: <http://blogs.msdn.com/b/davidklinems/archive/2005/11/16/493580.aspx>

参考: <http://msdn.microsoft.com/en-us/library/498928w2.aspx>

1. 通过try/catch和finally语句块**合理处理异常**并记录异常日志；

参考: <http://msdn.microsoft.com/en-us/library/vstudio/ms229005(v=vs.100).aspx>

1. 如果需要多级级联字符串，为节省内存消耗应该使用Stringbuilder而不是string；
2. 尽量避免做数据类型转换，否则会带来性能上的损失；

参考: <http://msdn.microsoft.com/en-us/library/ms173105.aspx>

1. 经常使用常量(const)和只读(readonly)关键字修饰字段；

参考: <http://msdn.microsoft.com/en-us/library/acdd6hb7(v=vs.100).aspx>

<http://msdn.microsoft.com/en-us/library/e6w8fe1b(v=vs.100).aspx>

1. 对于像int、decimal等已知数据类型应该避免使用default关键字。default关键字一般用于无法确定类型是数值类型还是引用类型的情况下；

参考: <http://msdn.microsoft.com/en-us/library/xwth0h0d(v=vs.100).aspx>

* **面向对象**

1. 理解透彻面向对象(OOP)的概念并应用到自己的代码中；
2. **牢记“SOLID”开发准则**：在面向对象的计算机编程语言中，“SOLID”是编程的5项基本原则，由[Robert C. Martin](http://en.wikipedia.org/wiki/Robert_C._Martin)于2000年提出，旨在建立健壮且易扩展的软件架构；

S：单一职责原则，Single responsibility principle

O：开闭原则，Open-close principle

L：里氏替换原则，Liskov substitution principle

I：接口隔离原则，Interface segregation principle

D: 依赖倒置原则，Dependence Inversion principle

参考: [http://en.wikipedia.org/wiki/SOLID\_(object-oriented\_design)](http://en.wikipedia.org/wiki/SOLID_(object-oriented_design)

1. 根据方法、类或变量的作用域合理使用访问说明符（private、public、protected、internal、protected internal）。假设一个类只在程序集内部使用，那么将该类标记为internal就足够了；

参考: <http://msdn.microsoft.com/en-us/library/kktasw36.aspx>

1. 通过使用接口可以降低程序耦合性，一些设计模式也是依附接口应运而生；

参考: <http://msdn.microsoft.com/en-IN/library/3b5b8ezk(v=vs.100).aspx>

1. 根据应用场景的需要，将类标记为密封类、静态类或抽象类；

参考: <http://msdn.microsoft.com/en-us/library/ms173150(v=vs.100).aspx>

1. 如果要让一个类显示自定义信息可以重载ToString方法；

参考: <http://msdn.microsoft.com/en-us/library/ms173154(v=vs.100).aspx>

* **防御性编程**

1. 索引或下标必须正确初始化，并且充分进行边界测试；
2. 导入的数据必须做合法性和完整性验证；
3. 方法/函数必须对输入参数进行合法性验证；
4. 在操作文件夹/文件时，对文件夹/文件是否存在进行判断；
5. 在程序结束时，所有的数据是否保证正确的进行了保存；

* **测试**

1. **单元测试**：根据开发人员使用的自测用例做单元测试来保证在进行QA测试之前完成了基本的功能测试；

参考: <http://msdn.microsoft.com/en-us/magazine/cc163665.aspx>

* **其它**

1. 及时地迁入或迁出源代码文件；

参考: [http://www.codeproject.com/Tips/593014/Steps-Check-in-Check-Out-Mechanism-for-TFS-Toavoi](http://www.codeproject.com/Tips/593014/Steps-Check-in-Check-Out-Mechanism-for-TFS-To-avoi)

1. 同事相互审查代码，与同事交换代码源文件以执行内部代码审查；
2. 使用一个类似于Silverlight Spy的工具来检查和处理SilverLight应用程序中的XAML界面以提高效率。这样在设计和运行模式下可以节约很多前后台的交互时间；
3. 使用fiddler工具检查HTTP网络的网络流量和带宽信息，以实时查看Web应用程序和服务的性能；
4. 无论是在visual studio外验证WCF服务方法是否符合预期，还是在visual studio中以附加进程的方式调试WCF服务，都可以用WCFTestClient.exe程序来完成；
5. 养成良好的阅读习惯，并不断学习业内专家和知名作者提出的最新编程指导准则来提高自己，如Martin Fowler、Kent Beck、Jeffrey Ritcher、Ward Cunningham、Scott Hanselman、Scott Guthrie、Donald E Knuth等；
6. 尝试参加专家举办的技术研讨会，以了解最新的软件发展趋势、最新的软件技术和最佳实践方式；
7. 采取必要的措施阻止并避免跨脚本攻击、SQL注入和其他安全漏洞；
8. 当在数据库中存储信息或在web.config中存储数据库连接字符串时要始终加密敏感信息，以避免未经授权的用户操作；

注：另外一篇关于代码审查的文章，请参考：

参考: <http://www.codeproject.com/Articles/524235/Codeplusreviewplusguidelines>