# 代码规范

**目录**

[代码规范 1](#_Toc169555522)

[一、引言 2](#_Toc169555523)

[二、编码风格 2](#_Toc169555524)

[1、通用原则 2](#_Toc169555525)

[2、缩进和空格 2](#_Toc169555526)

[3、命名规则 2](#_Toc169555527)

[4、注释规则 3](#_Toc169555528)

[5、格式化规则 3](#_Toc169555529)

[三、错误处理 3](#_Toc169555530)

[四、版本控制 3](#_Toc169555531)

## 一、引言

在本文档中，我们将定义一系列代码规范，以确保我们的代码库易于管理和维护。这些规范将帮助我们的团队成员编写清晰、一致的代码。

## 二、编码风格

### 1、通用原则

可读性优先：代码应该易于阅读和理解。

一致性：整个代码库应该遵循同样的风格。

简洁性：避免不必要的复杂性和冗余。

### 2、缩进和空格

使用4个空格进行缩进，不使用制表符（Tab）。

操作符两侧应添加空格，例如 `a + b` 而不是 `a+b`。

### 3、命名规则

类名：使用大驼峰命名，例如 `MyClass`。

函数/方法名：使用小驼峰命名，例如 `myFunction`。

常量：使用全大写字母和下划线，例如 `MAX\_COUNT`。

变量：使用 `camelCase`，并确保名称具有描述性，例如 `userList`。

### 4、注释规则

所有的公共接口（API）都必须有文档注释。

使用行注释 `//` 来解释代码的特定部分。

避免过多的注释，只对复杂的逻辑或不明显的代码部分进行注释。

### 5、格式化规则

每行代码长度不超过80个字符。

使用空行来分隔逻辑相关的代码块。

## 三、错误处理

使用异常来处理错误，并提供清晰的错误信息。

避免使用异常来控制流程。

## 四、版本控制

提交信息应该清晰、具有描述性。

每次提交应该是一个逻辑上独立的更改集。

提交时要明确提交的分支，提交的内容，并且提交后提出合并请求。