# 附件4 软件项目的编码案例说明

**目录**

[附件4 软件项目的编码案例说明 1](#_Toc138967113)

[1.导言 3](#_Toc138967114)

[1.1 目的 3](#_Toc138967115)

[1.2 范围 3](#_Toc138967116)

[1.3 编写说明 3](#_Toc138967117)

[1.4 术语定义 3](#_Toc138967118)

[1.5 引用标准 3](#_Toc138967119)

[1.6 参考资料 3](#_Toc138967120)

[1.7 版本更新信息 3](#_Toc138967121)

[2. 编码格式规范 3](#_Toc138967122)

[2.1缩进排版 3](#_Toc138967123)

[2.2行工度 3](#_Toc138967124)

[2.3断行规则 3](#_Toc138967125)

[2.4空行 4](#_Toc138967126)

[3.命名规范 4](#_Toc138967127)

[3.1类（Classes） 4](#_Toc138967128)

[3.2接口（Interfaces） 4](#_Toc138967129)

[3.3方法（Methods） 4](#_Toc138967130)

[3.4变量（Variabies） 4](#_Toc138967131)

[3.5常量（Constants） 4](#_Toc138967132)

[3.6 文件名（File Names） 4](#_Toc138967133)

[3.7 函数和方法参数（Function and Method Parameters） 4](#_Toc138967134)

[3.8 布尔变量（Boolean Variables） 4](#_Toc138967135)

[3.9 Vue组件命名（Vue Component Naming） 4](#_Toc138967136)

[4. 声明规范 5](#_Toc138967137)

[4.1 每行声明变量的数量 5](#_Toc138967138)

[4.2 初始化 5](#_Toc138967139)

[4.3 布局 5](#_Toc138967140)

[4.4 类和接口的声明 6](#_Toc138967141)

[5. 语句规范 6](#_Toc138967142)

[5.1简单语句 6](#_Toc138967143)

[5.2复合语句 6](#_Toc138967144)

[6. 注释规范 7](#_Toc138967145)

[6.1 Python注释规范 7](#_Toc138967146)

[6.2 Vue注释规范 7](#_Toc138967147)

[7. 代码范例 8](#_Toc138967148)

[7.1 FastAPI代码范例 8](#_Toc138967149)

[7.2 Vue代码范例 8](#_Toc138967150)

[8. 目录规范 10](#_Toc138967151)

[8.1 FastAPI的目录结构 10](#_Toc138967152)

[8.2 Vue的目录结构 10](#_Toc138967153)

# 1.导言

## 1.1 目的

该文档的目的是描述电子商务平台项目的编码规范和对代码的说明，其主要内容包括：

·编码规范

·命名规范

·注释规范

·语句规范

·声明规范

·目录设置

·代码说明

本文档的预期读者是

·开发人员

·项目管理人员

·质量保证人员

## 1.2 范围

该文档定义了本项目的代码编写规范以及部分代码描述和所有代码的说明.

## 1.3 编写说明

MVC：Model-View-Control（模式—视图—控制）的缩写，表示一个三层的结构体系。

## 1.4 术语定义

无

## 1.5 引用标准

无

## 1.6 参考资料

无

## 1.7 版本更新信息

本文档版更新记录如表D—1所示。

表D-1 版本更新记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 000 | 2023.6.27 | 0. 1 | 全部 | 初始发布版本 |
| 001 | 2023.6.29 | 1.0 | 6章 | 修改代码 |

# 2. 编码格式规范

## 2.1缩进排版

2个空格作为缩进排版的一个单位。

## 2.2行工度

尽量避免一行的长度超过80个字符,用于文档中的例子应该使用更短的行长,长度一般不超过70个字符。

## 2.3断行规则

当一个表达式无法容纳在一行内时,可以依据如下一般规则断开：

·在一个逗号后面断开；

·在一个操作符前面断开；

·宁可选择较高级别(higher-level)的断开,而非较低级别(lower-level)的断开；

·新的一行应该与上一行同一级别表达式的开关处对齐；

·如果以上规则导致代码混乱或者使代码都堆挤在右边,那就代之以缩进8个空格。

## 2.4空行

空行将逻辑相关的代码段分隔开，以提高可读性。下列情况应该总是使用两个空行：

·一个源文件的两个片段（section）之间；

·类声明和接口声明之间。

下列情况应该总是使用一个空行：

·两个方法之间；

·方法内的局部变量和方法的第一条语句之间；

·块注释或单行注释之前；

·一个方法内的两个逻辑段之间，用以提高可读性。

# 3.命名规范

命名规范使程序更易读，从而更易于理解。它们也可以提供一些有关标识符功能的信息，以助于理解代码。

## 3.1类（Classes）

* 类名应采用大驼峰命名法，每个单词的首字母都应大写。
* 类名应该是名词，并尽量简洁而富有描述性。
* 避免使用缩写词，除非缩写词被广泛接受和使用，如URL、HTML。
* 例如：class MyClassName

## 3.2接口（Interfaces）

大小写规则与类名相似。

## 3.3方法（Methods）

方法名是一个动词，采用大小写混合的方式，第一个单词的首字母大写，其后单词的首字母大写。

## 3.4变量（Variabies）

采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。变量名不应以下划线或美元符号开头，尽管这在语法上是允许的。变量名应简短且富于描述。变量名的选用应该易于记忆，即能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为i, j, k, m和n,它们一般用于整型；c, d, e,它们一般用于字符型。

## 3.5常量（Constants）

类常量和ANSI常量的声明，应该全部大写，单词间用下划线隔开。

## 3.6 文件名（File Names）

文件名应该使用小写字母，可以使用下划线或连字符作为单词分隔符。文件名应该能够清楚地表示文件的内容或用途。例如：my\_module.py

## 3.7 函数和方法参数（Function and Method Parameters）

函数和方法参数的命名规则应与变量名相同。参数名应该简洁而具有描述性，能够清楚地指出其用途。避免使用单个字符作为参数名，除非用作临时变量。例如：def my\_function(parameter1, parameter2)

## 3.8 布尔变量（Boolean Variables）

布尔变量通常用于表示真（True）或假（False）的状态。布尔变量的命名应具有描述性，能够清楚地表示其含义。例如：is\_active = True

## 3.9 Vue组件命名（Vue Component Naming）

Vue组件的命名应该采用短横线分隔的小写字母命名法。组件名应该具有描述性，能够清楚地表示其用途。例如：my-component。

# 4. 声明规范

**在 Python 和 Vue 中，声明规范略有不同，请参考以下修改后的声明规范：**

## 4.1 每行声明变量的数量

推荐每行只声明一个变量，这样可以更容易添加注释，例如：

```python

level1 = 0 # Indentation level

size = 10 # Size of the table

```

优于以下写法：

```python

foo, foo\_array = 0, [] # WRONG!

```

注意：在上面的示例中，类型和标识符之间有一个空格，可以使用制表符或空格来缩进。

## 4.2 初始化

尽量在声明局部变量时进行初始化，除非变量的初始值依赖于先前的计算。

在 Python 中：

```python

name = "John" # Initialize string variable

count = 0 # Initialize integer variable

```

在 Vue 的 JavaScript 中：

```javascript

let name = "John"; // Initialize string variable

let count = 0; // Initialize integer variable

```

## 4.3 布局

在代码块的开头进行变量声明（代码块是指在大括号 "{}" 内的任何代码）。不要在首次使用变量时才声明它，这会让程序员难以理解代码，也会影响代码的可移植性。

在 Python 中：

```python

def my\_method():

int1 = 0 # Declare variable at the beginning of the method (code block)

if condition:

int2 = 0 # Declare variable at the beginning of the if block

# ...

```

在 Vue 的 JavaScript 中：

```javascript

function myMethod() {

let int1 = 0; // Declare variable at the beginning of the function (code block)

if (condition) {

let int2 = 0; // Declare variable at the beginning of the if block

// ...

}

}

```

在 Vue 的单文件组件中，也遵循相似的规范。

## 4.4 类和接口的声明

在编写类和接口的声明时，应遵循以下格式规范：

- 方法名与参数列表之间的左括号 "(" 前不要有空格。

- 左大括号 "{" 与声明语句在同一行的末尾。

- 右大括号 "}" 单独成行，并与相应的声明语句对齐，除非是一个空语句。在一个类或接口的结束处，右大括号 "}" 应紧跟在上一个右大括号之后。

- 方法之间使用空行分隔。

在 Python 中：

```python

class MyClass:

def method1(self):

# Method 1 code

def method2(self):

# Method 2 code

```

在 Vue 的单文件组件中的 JavaScript 部分：

```javascript

export default {

methods: {

method1() {

// Method 1 code

},

method2() {

// Method 2 code

},

},

}

```

# 5. 语句规范

## 5.1简单语句

每行至多包含一条语句，例如：

argv++;//推荐

argc--;//推荐

argv++argc--;//避免

## 5.2复合语句

复合语句是包含在大括号的语句序列，形如“｛语句｝”。遵循原则如下：

·被括其中的语句应该较之复合语句缩进一个层次；

·左大括号“｛”应位于复合语句起始行的行尾；右大括号“｝”应另起一行并与复合语句首行对齐；

·大括号可以被用于所有语句，包括单个语句，只要这些语句是诸如if-else或for控制结构的一部分。这样便于添加语句而无须担心由于忘了加括号而引入bug。

# 6. 注释规范

## 6.1 Python注释规范

1. 单行注释：

- 使用 `#` 符号表示单行注释。

- 注释应与代码保持适当的缩进，以增加可读性。

2. 多行注释：

- 使用三个引号（"""）或三个单引号（'''）将多行注释括起来。

- 多行注释通常用于函数、类或模块的文档字符串（docstring）。

3. 函数和方法注释：

- 在函数或方法定义的下一行使用文档字符串（docstring）来提供对函数或方法的描述。

- 文档字符串应包含函数的输入、输出和功能说明，以及必要的参数和返回值的说明。

4. 类注释：

- 在类定义的下一行使用文档字符串（docstring）来提供对类的描述。

- 文档字符串应包含类的功能、属性和方法的说明。

5. 注释风格：

- 注释应该清晰、简洁并具有描述性。

- 使用正确的语法和拼写。

- 避免使用无意义或冗长的注释。

- 注释应该解释代码的目的、意图和重要细节，而不仅仅是重复代码本身。

6. 注释位置：

- 注释应与其描述的代码行紧密相关，位于代码行上方或右侧。

- 在复杂的代码块或关键算法的关键部分添加注释。

7. 注释内容：

- 注释应该解释代码的意图和目的，而不仅仅是重复代码的功能。

- 提供必要的上下文信息，例如输入和输出的格式、单位或范围。

- 解释代码的限制、假设和依赖关系。

8. 更新注释：

- 当代码发生更改时，应及时更新注释以保持其准确性。

- 更新注释可以帮助其他开发人员了解代码的变更和影响。

## 6.2 Vue注释规范

1. 单行注释：

- 使用 `//` 符号表示单行注释。

- 注释应与代码保持适当的缩进，以增加可读性。

2. 多行注释：

- 使用 `/\* \*/` 将多行注释括起来。

- 多行注释通常用于解释代码块的功能、实现细节或相关说明。

3. 模板注释：

- 在Vue模板中，可以使用 `<!-- -->` 将注释括起来。

- 模板注释通常用于解释特定部分的功能或提供一些上下文信息。

4. 组件注释：

- 在Vue组件的代码块上方使用多行注释，描述组件的功能、用途和重要细节。

- 组件注释应包括组件的名称、作者、版本和许可等相关信息。

5. 更新注释：

- 当代码发生更改时，特别是在修复错误或添加新功能时，应更新注释以反映代码的变更。

- 更新注释可以帮助团队成员了解代码的修改和影响。

6. 注释内容：

- 注释应该解释代码的目的、意图和重要细节。

- 提供关于组件、模板或特定功能的说明和使用方法。

- 解释代码的限制、假设和依赖关系。

- 提供其他开发人员可能需要了解的上下文信息。

7. 注释风格：

- 注释应该清晰、简洁并具有描述性。

- 使用正确的语法和拼写。

- 避免使用无意义或冗长的注释。

- 注释应该解释代码的意图和设计决策，而不仅仅是重复代码本身。

# 7. 代码范例

## 7.1 FastAPI代码范例

from fastapi import FastAPI

app = FastAPI()

@app.get("/items/{item\_id}")

async def read\_item(item\_id: int, q: str = None):

"""

根据item\_id获取特定项目的信息

Args:

item\_id (int): 项目的唯一标识符

q (str, optional): 查询参数，用于过滤项目信息

Returns:

dict: 包含项目信息的字典

"""

item = {"item\_id": item\_id}

if q:

item.update({"q": q})

return item

## 7.2 Vue代码范例

<template>

<div>

<!-- 用户列表 -->

<ul>

<li v-for="user in userList" :key="user.id">

{{ user.name }} - {{ user.email }}

</li>

</ul>

<!-- 添加用户表单 -->

<form @submit.prevent="addUser">

<input v-model="newUser.name" placeholder="姓名" required>

<input v-model="newUser.email" placeholder="邮箱" required>

<button type="submit">添加用户</button>

</form>

</div>

</template>

<script>

export default {

data() {

return {

userList: [], // 用户列表

newUser: { // 新用户对象

name: '',

email: ''

}

}

},

methods: {

// 添加用户

addUser() {

// 执行添加用户的逻辑

// ...

// 清空输入框

this.newUser.name = '';

this.newUser.email = '';

}

}

}

</script>

<style scoped>

/\* 组件样式 \*/

ul {

list-style: none;

}

</style>

# 8. 目录规范

## 8.1 FastAPI的目录结构

1. main.py

· 项目的入口文件，包含 FastAPI 应用的初始化和配置。

2. api 目录

· 包含定义 API 路由和处理程序的模块。

3. models 目录

· 包含数据模型的定义和包含与数据库交互相关的模块，负责处理数据的持久化和查询操作。

4. router 目录

· 包含与业务逻辑相关的服务模块，用于处理复杂的业务逻辑。

5. utils 目录

· 包含一些通用的工具函数、辅助函数或中间件。

6. tests 目录

· 包含单元测试和集成测试的模块，用于确保代码的正确性和稳定性。

7. static 目录（可选）

· 包含静态文件，如图像、CSS、JavaScript 等。

8. config 目录

· 包含配置文件，如数据库连接配置、日志配置等。

## 8.2 Vue的目录结构

1. assets 目录

· 包含项目中使用的静态资源文件，如图像、字体文件等。

2. components 目录

· 包含可复用的 Vue 组件，组件可以根据功能或类型进行分组。

3. views 目录

· 包含页面级别的组件，通常与路由对应。

4. router 目录

· 包含 Vue Router 相关的配置文件和路由定义。

5. store 目录

· 包含 Vuex 相关的配置文件和状态管理模块。

6. services 目录（可选）

· 包含与后端 API 通信的服务模块。

7. plugins 目录（可选）

· 包含 Vue 插件的定义和配置文件。

8. filters 目录（可选）

· 包含全局过滤器的定义。

9. utils 目录

· 包含通用的工具函数或辅助函数。

10. tests 目录

· 包含单元测试和集成测试的文件。

11. public 目录

· 包含不需要经过构建过程的静态资源文件，如 index.html、favicon.ico 等。

12. config 目录

· 包含项目的配置文件，如环境变量配置、构建配置等。