广州因特信息科技有限公司

软件开发规范 - 设计编码部分

目录

[1 框架 2](#_Toc480554093)

[1.1 开发环境 2](#_Toc480554094)

[1.2 模块化原则和要求 2](#_Toc480554095)

[1.3 多语言原则和要求 2](#_Toc480554096)

[2 数据库 2](#_Toc480554097)

[2.1 Database 2](#_Toc480554098)

[2.2 Table 2](#_Toc480554099)

[2.3 StoreProcessur 2](#_Toc480554100)

[2.4 Triger 3](#_Toc480554101)

[2.5 Function 3](#_Toc480554102)

[2.6 Setting 3](#_Toc480554103)

[2.7 Job 3](#_Toc480554104)

[3 后端 4](#_Toc480554105)

[3.1 Windows 4](#_Toc480554106)

[3.1.1 服务Service 4](#_Toc480554107)

[3.1.2 逻辑层Logic 4](#_Toc480554108)

[3.1.3 接口 4](#_Toc480554109)

[3.1.4 辅助工具 4](#_Toc480554110)

[3.1.5 通用规范 4](#_Toc480554111)

[3.2 PHP 4](#_Toc480554112)

[3.2.1 服务Service/定时服务 4](#_Toc480554113)

[3.2.2 逻辑层Logic 5](#_Toc480554114)

[3.2.3 接口 5](#_Toc480554115)

[3.2.4 辅助工具 5](#_Toc480554116)

[3.2.5 通用规范 5](#_Toc480554117)

[4 前端 5](#_Toc480554118)

[4.1 UI页面 5](#_Toc480554119)

[4.2 JS框架 6](#_Toc480554120)

[4.3 CSS框架 6](#_Toc480554121)

[4.4 通用规范 6](#_Toc480554122)

# 框架

## 1.1 开发环境

## 1.2 模块化原则和要求

## 1.3 多语言原则和要求

# 2 数据库

## 2.1 Database

## 2.2 Table

## 2.3 StoreProcessur

## 2.4 Triger

## 2.5 Function

## 2.6 Setting

## 2.7 Job

# 3 后端

## 3.1 Windows

### 3.1.1 服务Service

### 3.1.2 逻辑层Logic

### 3.1.3 接口

### 3.1.4 辅助工具

### 3.1.5 通用规范

* 命名
* 异常捕捉
* Log输出
* 技术文档要求

## 3.2 PHP

### 3.2.1 服务Service/定时服务

### 3.2.2 逻辑层Logic

### 3.2.3 接口

### 3.2.4 辅助工具

### 3.2.5 通用规范

* 命名
* 异常捕捉
* Log输出
* 技术文档要求

# 4 前端

## 4.1 UI页面

## 4.2 JS框架

## 4.3 CSS框架

## 4.4 通用规范

* 命名
* 异常捕捉
* Log输出
* 技术文档要求

你想问编码规范。还是设计规范 还是界面规范 还是操作规范  
  
编码规范：  
 代码书写规范（如程序结构。。。）  
 代码注释规范（文件头注释、类注释、方法注释、注释的类型。。。）  
 代码命名规范（属性、方法、字段、事件、。。。如何是规范命名）

一般系统的功能项目文件夹结构

Common

DLL

Entity

"项目"+BLL

"项目"+DAL

市场API或功能

DLL 放置引用的第三方DLL和生成的DLL在这个文件夹

"功能"+API 开发的API功能的项目

Document 文档

TestUi 测试UI

现在我们C# code要优化的方向有三个: 总框架由大到小的控制，系统配置的统一化，异常捕捉和处理。

另外一个是我们之前没有定义系统各模块的单据、数据项和数据流，就会造成开发的过程中经常要返工补漏洞。导致测试的时候才发现漏掉一些功能。