## 编码规范说明书

**一、 模块头部注释规范**

以一个物理文件为单元的都需要有模块头部注释规范，用于每个模块开头的说明，主要包括以下几部分：

1. 文件名称(File Name)： 文件的名称

2. 功能描述(Description)： 模块的功能描述与大概流程说明

3. 数据表(Tables)： 用到的数据表，视图，存储过程的说明。

4. 作者(Author)：创建人

5. 日期(Create Date)：创建时间

6. 参考文档(Reference)(可选)： 文档所对应的分析文档，设计文档。

7. 引用(Using) (可选)﹕ 开发的系统中引用其它系统的 Dll、对象时，要列出其对应的出处，是否与系统有关，以方便制作安装文档。

8. 修改记录(Revision History)：若档案的所有者改变，则需要有修改人员的名字、修改 日期及修改理由。

二、 方法注释规范

1、类、接口、属性、方法必须有<Summary>节，另外方法如果有参数及返回值，则必须 有<Param>及<Returns>节。

格式如下：

/// <summary>/// …

/// </summary>

/// <param name=””></param>

/// <returns></returns>

2、公用类库中的公用方法需要在一般方法的注释后添加作者、日期及修改记录信息，统

一采用 XML 标签的格式加注。

格式如下：

<Author></Author> 作者

<CreateDate></CreateDate> 建立日期

<RevisionHistory> 修改记录

<ModifyBy></ModifyBy> 修改作者

<ModifyDate></ModifyDate> 修改日期

<ModifyReason></ModifyReason> 修改理由

<ModifyBy></ModifyBy> 修改作者

<ModifyDate></ModifyDate> 修改日期

<ModifyReason></ModifyReason> 修改理由

<ModifyBy></ModifyBy> 修改作者

<ModifyDate></ModifyDate> 修改日期

<ModifyReason></ModifyReason> 修改理由

</RevisionHistory>

<LastModifyDate></LastModifyDate> 最后修改日期

**二、 代码格式**

<1>、文件头声明

源文件的头部需要功能说明描述。该段定义在 package 之上，例如：

／\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* copyright@2009 youngfar Co. Ltd.

\* All right reserved.

\* 功能描述

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

<2>、Using 顺序

按以下顺序：

a)、.net 标准包

b)、.net 扩展包

c)、使用的外部库的包（例如 Microsoft.Practices.EnterpriseLibrary.Data）

d)、使用的项目的公共包

e)、使用的模块的其他包

每一类 Using 后面加一个换行。

<3>、类头声明

在类的头部需要功能说明，作者，创建时间，版本进行描述。该段定义在类声明之上，

例如：

/\*\*

\* 功能描述

\*

\* @author sometimes

\* @since 2009.04.01

\* @version 1.0.0

\* \*/

注：如若非本人修改需要标注，修改人，修改原因，修改时间，修改函数名称；若为

本人修改只需按要求升级版本号即可。

例如：

/\*\*

\* 功能描述

\*

\* @author sometimes

\* @since 2009.04.01

\* @version 1.0.1

\* @modify 1.0.1 by gongbin since 2009.04.01 method = test() 修改样例

\* \*/

**三、其他规范**

<1>、ASPX 页面编码规范。

a)、Request、session、application

web 应该有 request session application 的概念不要这三个不分，不要把所有的

数据都存储在 session 中。而且 session 应该做一个 session 容器内，方便管理

与监控,不要把 session 乱存。

b)、文件命名与存放位置

应该分功能和使用类型将对应文件分类存放，方便管理。例如：

WERROOT

Js

Css

imagesjsp

+---modeA

+---addUser.aspx

+---addUser.css

+---addUser.js

c)、开头注释

写出 aspx 页面所展示的大概功能

<%--

用户信息的显示页面

auther wang xxx

--%>

d)、ASPX 声明语句

如果非特殊要求程序的 ASPX 页面 ，html 统一用成 UTF-8 编码

<%@

page

language="C#"

contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<2>、JAVASCRIPT 编码规范

a)、单个文件长度不能超过 800 行（C#也是这样） // 目的：支持尽可能的模块化封

装，强制精炼简化代码。利于维护

b)、不要调用平台封装以外的方法，经由平台统一提供方法调用.// 利于统一管理，

功能修改，迁移。

c)、注释完备（功能，参数，返回值的说明）

EX:

/\*\*

\*@description: 去除字符串前后的空格

\*@param des {string} 要清理的字符串

\*@return result {string} 清理结果

\*/

function trim(des){

var result=null;

... some code ...

return result;

}

d)、JS 的返回值类型约束较弱，要求在方法设计之初提供严格的对外调用返

回.{Number ,boolean, String/null, Object/null 禁止 undefine 返回} //严格

的类型可在后续的编程活动中精炼，清晰的定义流程

**四、资源释放**

所有外部资源都必须显式释放。例如：数据库连接对象、IO 对象等。

**五、 错误处理**

<1>、不要“捕捉了异常却什么也不做“。如果隐藏了一个异常，你将永远不知道异常到底

发生了没有。

<2>、发生异常时，给出友好的消息给用户，但要精确记录错误的所有可能细节，包括发生

的时间，和相关方法，类名等。

<3>、只捕捉特定的异常，而不是一般的异常。

**六、其他约定**

<1>、一个方法只完成一个任务。不要把多个任务组合到一个方法中，即使那些任务非常小。

<2>、使用 C#的特有类型，而不是 System 命名空间中定义的别名类型。

<3>、别在程序中使用固定数值，用常量代替。

<4>、避免使用很多成员变量。声明局部变量，并传递给方法。不要在方法间共享成员变量。

如果在几个方法间共享一个成员变量，那就很难知道是哪个方法在什么 时候修改了它

的值。

<5> 、 别 把 成 员 变 量 声 明 为 public 或 protected 。都声明为 private 而使用

public/protected 的属性

<6> 、不在代码中使用具体的路径和驱动器名。 使用相对路径，并使路径可编程。

<7> 、应用程序启动时作些“自检”并确保所需文件和附件在指定的位置。必要时检查数据

库连接。出现任何问题给用户一个友好的提示。

<8>、如果需要的配置文件找不到，应用程序需能自己创建使用默认值的一份。

<9>、如果在配置文件中发现错误值，应用程序要抛出错误，给出提示消息告诉用户正确值。

<10>、DataColumn 取其列时要用字段名，不要用索引号。

例： 正确 DataColumn[“Name”]

不好 DataColumn[0]

<11>、在一个类中，字段定义全部统一放在 class 的头部、所有方法或属性的前面。

1. **系统运行环境**

1.本系统需要的硬件环境如下：

操作系统：Windows 10

RAM：8GB或以上

2.测试环境

Eclipse,MY SQL

3.软件环境

本系统运行所需要的软件环境为：

操作系统：Windows 7及以上

语言软件：Eclipse

数据库：MY SQL

8、**系统维护**

对软件进行维护，是为了通过必要的维护工作使得系统能够持久的满足用户的需要。本系统的维护将从以下四个方面进行：

1.改正性维护

该整形维护是指在软件交付使用后，因开发时测试的不彻底、不完全，必然会有部分隐藏的错误遗留到运行阶段。这些隐藏下来的错误在某些特定的使用环境下就会暴露出来。为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应当进行的诊断和改正错误的过程就叫做改正性维护。

2.适应性维护

是指在使用过程中，外部环境，如新的硬件或者软件配置等，以及数据环境，如数据库、数据格式、数据输入/输出方式、数据存储介质等可能发生变化时岁软件进行的维护。

3.完善性维护

是指在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为满求了足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。

4.预防性维护

采用先进的软件采用先进的软件工程方法对需要维护的软件或软件中的某一部分进行设计、编制和测试。

通过使用以上维护方式，从而使得软件能够更好的使用，同时更进一步提高工作效率。