开发框架设计文档

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 说明 | 修订人 | 修订时间 |
| 1.0 | 框架初步设计 | 许豪鹏 | 2017.06.01 |

**1.引言**

**1.1 背景**

软件系统随着业务的发展需要增加的功能模块越来越多，就会变得越来越复杂。再加上不停地开发新项目，各个项目之间涉及到的知识，内容，非常广泛，而底层功能实现又具有极高的相似性，在团队协作开发时，没有统一标准和规范，开发人员就需要一遍又一遍地编写已经实现过的代码模块，造成大量代码冗余，且彼此很难看懂对方写的代码，导致最后出现Bug或二次开发维护时，根本无从下手。对项目来说，如果每次都从头开发，那将会是一个极为漫长而艰难的事情。这不仅极大地增加了开发周期和后期项目维护成本，而且会影响到产品最终的质量。

在这个快餐年代，敏捷开发是所有企业都向往的目标，只有提高效率，才能够获得更多市场。拥有一个成熟的框架，会帮开发人员免去很多基础性功能的工作，让他们只需要专注于业务逻辑便可快速开发出高质量的项目产品。另外企业可以无需担心开发成员的变动导致项目维护难度增加，因为框架的代码规范让新加入的开发人员仅需简单的学习便可快速融入团队。

**1.2 读者对象**

本文档提供给研发部后台（C#）开发人员，新成员开发项目前请务必熟悉本文档相关内容

**1.3 术语与缩写解释**

|  |  |
| --- | --- |
| **缩写、术语** | **解释** |
| ORM | 对象关系映射（英语：(Object Relational Mapping，简称ORM，或O/RM，或O/R mapping），是一种数据交互技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换 |
| Dapper | 一款开源且性能较为卓越的轻量级ORM框架 |
| Newtonsoft.Json | 一款开源的C# Json解析程序集 |
| NPOI | NPOI 是 POI 项目的 .NET 版本。POI是一个开源的Java读写Excel、WORD等微软OLE2组件文档的项目。 |
| Log4Net | log4net库是Apache [log4j](http://baike.baidu.com/item/log4j)框架在Microsoft .NET平台的实现，是一个帮助程序员将日志信息输出到各种目标（控制台、文件、数据库等）的工具。 |
| Rpository（仓储模式） | 仓储(Respository)是存在于工作单元和数据库之间单独分离出来的一层，是对数据访问的封装。其优点：业务层不需要知道它的具体实现，达到了分离关注点。提高了对数据库访问的维护，对于仓储的改变并不会改变业务的逻辑 |
| Redis | Redis是一个开源的使用ANSI [C语言](http://baike.baidu.com/item/C%E8%AF%AD%E8%A8%80)编写、支持网络、可基于内存亦可持久化的日志型、Key-Value[数据库](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)，并提供多种语言的API。可用于服务器缓存技术的实现，搭配Sql使用 |

**2.框架介绍**

**2.1 框架整体组织结构概览**

Sqlserver

# 缓存ICache<T>

# Interface

其他可扩展功能类库（封装常用的方法）

# 仓储IRpository<T>

Log4Net

Excel

JSON

NPOI

调用

接口层

Newtonsoft.Json

客户端

Oracle

SqLite

Dapper

数据访问

业务

领域层

C/S

B/S

调用

数据库

开发工作域

开发框架

代码生成工具

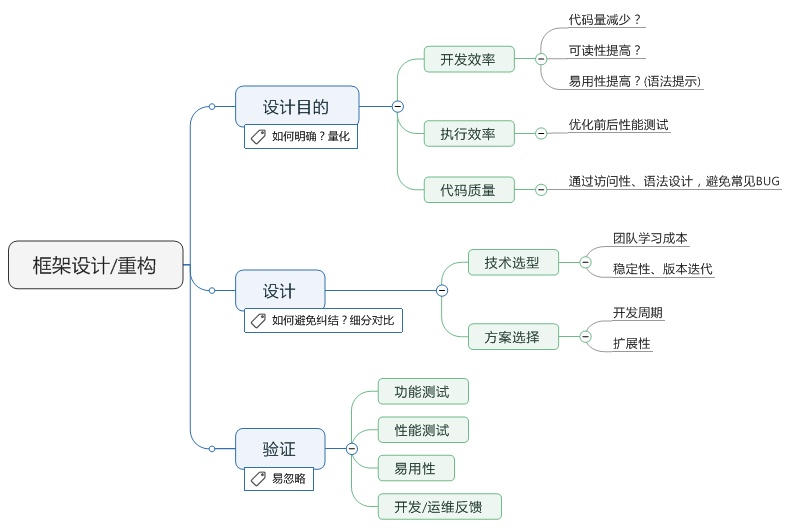
操作

实体

**2.2 框架设计图解释**

* 图中所有右箭头意为调用/请求，左箭头为响应/输出
* 开发工作域为开发人员在实际开发中需要进行编写代码的模块
* Interface(接口层) 主要封装可扩展实现方式的功能接口，其余功能均封装在可扩展功能类库中
* ICache 为缓存层接口，封装服务器缓存数据功能，减轻服务器对数据库读写压力，可采用Redis缓存技术、MongoDB
* IRpository 为仓储接口，负责与DAL层(封装各数据库的SQL语句解析、表达式解析等功能)对接
* Log4net负责程序日志记录，方便开发人员更好地发现程序异常
* NPOI负责常用的Excel格式导出
* Newtonsoft.Json 负责Json序列化和反序列化
* 操作数据库实体时，由代码生成工具生成实体，开发人员再对实体进行操作（无需人工创建实体）

**2.3 框架设计思路**



**2.4 框架规范**

**2.4.1 公共规范**

* **命名应该尽量为英文名词或者是名词短语缩写，保证有完整的意义，不要使用任何拼音、特殊符号或者无意义的单词，如：abc,x,y,z，xueji,yey……**
* **代码的缩进使用Tab，保证代码页的整洁规范**
* **所有的类、方法体、属性等对象都必须写上对应的短注释，有特殊意义的要添加明确的长注释**
* **代码文件中，有相似功能或者意义的方法、属性、字段、事件等，可适当使用#region ….#endregion进行分组，方便后期维护扩展**
* **程序集的命名空间不要超过3级**
* **框架进行维护时，所有新增、或者改动方法均需要测试才可发布版本，并在文档中留下备案**

**2.4.2 类规范**

* 类名使用Pascal规则
* 常用类的命名空间设置为顶级，即程序集的名称/程序集的命名空间，较少使用的可放于二级命名空间，最好不要超过三级命名空间  
     
  **2.4.3 接口规范**
* 类名使用Pascal规则
* 命名以大写字符I开头。如IFly,IWalk,IConnection  
     
  **2.4.4 枚举规范**
* 类名使用Pascal规则
* 不要在 Enum 类型名称上使用 Enum 后缀  
     
  **2.4.5 变量规范**
* 使用 Camel 命名规则
* 在简单的循环语句中计数器变量使用 i, j, k, l, m, n

**2.4.6 字段规范**

* 使用 Camel 命名规则
* 以下划线\_开头，例如 \_name,\_Id
* 字段需声明在类的顶部

**2.4.6 方法规范**

* 使用Pascal规则
* 对方法名采用一致的动词/宾语或宾语/动词顺序。

-例如，将动词置于前面时，所使用的名称诸如 InsertWidget 和 InsertSprocket；将宾语置于前面时，所使用的名

* 不要在方法中重复类的名称。

-例如，如果某个类已命名为 Book，则不要将某个方法称为 Book.CloseBook，而可以将方法命名为 Book.Close。

* 方法体的实现代码最好不要超过50行，对于需要大量代码实现的模块，将功能点细化封装

**2.4.7 属性/特性规范**

* 使用Pascal规则
* 对于bool型属性或者变量使用Is（is）作为前缀，不要使用Flag后缀。
* -例如应该使用IsDeleted,而不要使用DeleteFlag
* 自定义特性需以Attribute结尾

**2.4.8 事件命名**

* 使用Pascal规则
* event handlers命名使用 EventHandler 后缀
* 事件参数使用EventArgs 后缀
* 事件命名使用语法时态反映其激发的状态，例如 Changed，Changing  
  考虑使用动词命名. 变量命名

**2.5 框架命名规范对照表**

**2.5.1 基础命名**

|  |  |  |
| --- | --- | --- |
| **标识符** | **大小写** | **示例** |
| 类/结构 | Pascal | AppDomain |
| 枚举类型 | Pascal | ErrorLevel |
| 枚举值 | Pascal | FatalError |
| 事件 | Pascal | ValueChange |
| 异常类 | Pascal | WebException总是以 Exception 后缀结尾 |
| 接口 | Pascal | IDisposable 总是以 I 前缀开始。 |
| 方法 | Pascal | GetUserInfo |
| 命名空间 | Pascal | KrFrameWork.Core |
| 参数 | Camel | typeName |
| 属性 | Pascal | BackColor |
| 字段 | Camel | \_name |

**2.5.2 与页面控件有关(html标签/winform/Wpf)**

|  |  |  |
| --- | --- | --- |
| **数据类型** | **数据类型简写** | **标准命名举例** |
| Label | Lbl | Message\_Lbl |
| LinkLabel | LLbl | ToBaidu\_LLbl |
| Button | Btn | Save\_Btn |
| TextBox | Txt | Name\_TBox |
| MainMenu | Menu | UserInfo\_Menu |
| CheckBox | CBox | IsAdmin\_CBox |
| RadioButton | Rdo | IsAdmin\_Rdo |
| GroupBox | GBox | User\_GBox |
| PictureBox | Pic | Icon\_Pic |
| Panel | Panel | UserInfo\_Panel |
| DataGridView | Dgv | UserInfo\_Dgv |
| ListBox | LBox | UserInfo\_LBox |
| ComboBox | ComBox | UserInfo\_ComBox |
| ListView | LView | UserInfo\_LView |
| TreeView | TView | UserInfo\_TView |
| DateTimePicker | DTPicker | UserInfo\_DTPicker |
| Timer | Timer | UserInfo\_Timer |
| ImageList | LImg | UserInfo\_Limg |
| 其他控件 | 根据样例命名 | 根据样例命名 |

**2.6 缓存模块（ICache）关系流程图**

ICache

其他缓存技术

IMongoDB

IMemcache

IRedis

MongoDb，文档型NoSql,读写性能高

一套开放源的分布式高速缓存系统，将数据存在内存中大大提高读取速度

与memcached一样，为了保证效率，数据都是缓存在内存中。比前者拥有更多特性以及数据结构

没有数据时从数据库提取并储存

IRpository<T>

ORM接口，提供数据的读写功能

**2.7 ORM组件图**

IRpository <T> 仓储接口（+ Add(),+ GetList(), + PageList() ……）

IUnitTransaction 单元事务接口

IPartialEvaluator 表达式解析入口类

DbMethod 数据库常用函数（没有具体实现，用于解析转换Sql函数）

DbSort 返回数据库Order By 语句

FieldSelector 字段选择器：将表达式转换成Sql 语句操作字段

PredicateBuilder Where表达式解析(将表达式经过计算之后得到更简单的表达式)

UnitAction 工作单元单位（Action、Conn）存放每个工作单元要执行的sql方法和连接对象

UnitOfWork 工作单元（+BeginTransaction(),+Commit()），创建单元事务，批量执行单元事务,通过使用UnitOfWork的Commit启动，出错时回滚所有事务方法

UnitTransaction 单元事务（+Commit（））执行Sql事务,通过使用Commit方法启动，出错时回滚当前事务方法

其他数据库技术

DAL.Sqlserver

SqlserverRepository<T>: IRpository<T>

SqlBuilder Sql语句构建

MethodSqlBuilder Sql函数转换（C#函数转换Sql函数）

PredicateReader 表达式解析（负责将表达式转化弄成Sql的Where条件语句）

SqlCommand Sql命令对象

SqlServerAddRepository：IRepository<T> 常用数据插入方法实现

SqlServerDeleteRepository：IRepository<T> 常用数据删除方法

SqlServerRepository：IRepository<T> 常用语句执行方法（Query、Excute）

SqlServerSelectRepository：IRepository<T> 常用查询（Get、GetList、GetListEx）

SqlServerUpdateRepository：IRepository<T> 常用更新（Update、UpdateSelect）