DEGAGE

开发平台

## 前言

此开发平台有以下几点目标

* 减少常规代码编写的工作量、以往复杂代码的编写难度。
* 减少功能代码之间的相关性
* 减少部署与维护的难度。（提供自动化工具，减少人工参与）
* 减少系统后续维护、查错的难度。（提供定制的调式工具）

## 大纲

[前言 2](#_Toc21754)

[大纲 3](#_Toc2133)

[1.核心类库 4](#_Toc11956)

[1.1扩展 4](#_Toc10930)

[2.数据模型 6](#_Toc6165)

[2.1数据的结构模型（Schema） 6](#_Toc24520)

[2.2 ORM 6](#_Toc32077)

[3. 服务模型 8](#_Toc6739)

[3.1远程调用技术 8](#_Toc10288)

[3.2 服务发现 8](#_Toc16061)

[4. 文件管理 9](#_Toc11741)

[4.1运作机制 9](#_Toc11928)

[4.2 文件索引结点 9](#_Toc20164)

[4.3 文件存储结点 9](#_Toc8470)

[5. 消息组件 10](#_Toc972)

[6. 界面库 11](#_Toc14359)

[7.定时作业 12](#_Toc2462)

[8.辅助功能 13](#_Toc32448)

[8.1 工具类库 13](#_Toc872)

[8.2 网络代理 13](#_Toc9659)

[9.调式工具 14](#_Toc22672)

[10.互联互通平台 15](#_Toc12947)

[10.1互联域 15](#_Toc19764)

[10.2调用转发 15](#_Toc17092)

[10.3负载均衡 15](#_Toc4967)

## 组件关系图

用户界面（Concison）

Windows

Xamarin

Mono.Form

Core Razor Page

权限管理服务

客户端

配置管理（Agility）本地

服务端

配置管理（Agility）全局

功能组件客户端

功能组件服务端

消息

组件

权限管理配置

工作流组件

服务模型

管理平台（部分与VS集成）

字典信息管理

客户端部署、监控

客户端、服务端升级

模型、视图代码生成

工作流流程配置

数据模型

文件管理

调式工具

工具库

客户端前哨

定时作业（Oscillator）

缓存组件

互联互通平台

## 1.核心类库

### 1.1常用类型的扩展

#### 1.1.1字符串

#### 1.1.2 枚举

## 2.数据模型

对象、Lamdba、Linq 到SQL 的转化，结果集到对象的转化

对不同数据库的支持

从数据库生成模型类，结构文档

热点数据缓存，支持自行计算、手动设置

### 2.1数据的结构模型（Schema）

从类的Attribute、文件、服务获取结构模型

2.2.1 导出 Schema 信息

* 常规导出：

Excel、Word

* 代码导出：

代码导出模板、不同语言的支持

#### 2.2.2 Schema信息对比

目前同种数据库结构信息的对比

### 2.2 ORM

### 2.3数据的缓存

2.3.1 缓存依据

缓存依据列，主键列默认

2.3.2 缓存一致性

缓存在存储时使用多级结构

例如：现有表 Student 其信息如下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student | | | | | |
| No（PK） | Age | Dept | Sex | Name | Born |
| 1 | 21 | 计算机系 | 男 | 王大毛 | 1994-10-20 |
| 2 | 19 | 数学系 | 女 | 王二毛 | 1996-10-12 |
| 3 | 20 | 历史系 | 女 | 王三毛 | 1995-08-18 |
| 4 | 19 | 计算机系 | 女 | 王四毛 | 1996-09-12 |

单条缓存数据的一致性

总的是使用先更新数据库，在删除缓存的策略，也就是[《Cache-Aside pattern》](https://docs.microsoft.com/en-us/azure/architecture/patterns/cache-aside)

另外针对一些情景也相应给出一些解决方案。

1. 数据已写入数据库但是删缓存失败（例如网络原因等），此时其他请求是不知道旧的缓存无效的。

方案1：在写入数据库前设置相应缓存的状态为不确定，并设置其在短时间内过期，其他请求读取时，如果发现缓存的不确定状态，则请求提交至数据库，且不再写回数据库。

2.

## 服务模型

为各平台提供通用的远程调用功能，支持安全验证、数据加密、压缩传输

等辅助功能。

服务端会话信息的维护、自行寄宿。

可自定义对通信接口的实现以兼容 Web Service、WCF 等传统通信模型或者其他服务模型。

流量控制，连接数控制，并发数控制，超时控制，调用间隔控制（访问频次控制）

### 3.1远程调用技术

传输加密、数据压缩、调用安全性配置

返回值封装 Return<T>

#### 3.1.1参数检查器

为服务操作提供参数检查器，已使得操作被调用时，可以检查传入参数，若参数非法，则终止调用，并向调用方返回参数无效的信息，接口定义如下：

IParameterInspector

{

Boolean Validation(Object[] args)

}

### 3.2 服务发现

## 文件管理

提供文件操作接口支持 HTTP，.NET。

不提供更新接口

### 4.1 运作机制

### 4.2 文件索引结点（Master Server）

1.生成 Chunk 块的 ID（递增的顺序号），并为 Chunk 块分配存储结点（副本）

2.收集存储结点服务器上的 Chunk 块元数据信息，以及 Chunk Server 状态信息

3.客户端读取文件时为其选择合适的 Chunk Server

通过Chunk 的存储服务器，以及其负载

1. 客户端上传文件时为其选择合适的 Chunk Server

通过 Chunk 元数据信息（剩余空间），文件大小、Chunk 的存储服务器，服务器负载

1. 客户端删除文件时，将删除请求转发到各个拥有 Chunk 的存储服务器上，此操作不需要保证实时。
2. 为文件生成 File ID

最大 128 位

前 6 位为 Chunk ID 的36 进制字符串

//暂定

### 4.3 文件存储结点（Chunk Server）

1.客户端读取文件时，通过 Chunk 块元数据从 Chunk 块中读取压缩的数据，并解压写入值请求流中，处理时递增服务器 Processing Count 。

2.客户端写入文件时，通过 File ID 确定写入块（假如因为 Master Server与 Chunk Server，Chunk 元数据同步不及时导致块剩余空间不足，则中止写入请求，并发送最新的块元数据信息至 Master Server）。写入块后更新块元数据，并发送至 Master Server。处理时递增服务器 Processing Count 。

#### Chunk 块

用于存放文件数据，一个文件只会被存放在一个块中，相同标识的块可以在多个存储结点上存在。

Chunk 块默认大小：100MB，可设置。

每个Chunk 块对应有一个 Schema 文件，用以保存Chunk 块的元数据信息。

Chunk 块的元数据信息：块的大小，块中包含文件的数量、当前数据末的偏移量、当前块剩余空间的大小、存储空间压缩的次数、元数据最后修改日期。

Chunk块元数据信息-文件存储信息：文件标识、文件数据起始点的偏移量、文件压缩后大小、文件大小。

文件写入块时的压缩与读出时的解压: GZip、Brotli、Deflate

## 消息组件

基于 Redis 实现

客户端封装提供统一的访问接口

支持短信、邮件推送

支持消息的持久化

### Web平台的消息推送

Asp.net core singalR

## 用户界面

界面库，提供大量基本控件，统一各个平台界面API，尽量保证类型之间的关系以及

类型的行为相似。

### 6.1逻辑帧和渲染帧

### 6.2 动画

### 6.2 Windows 平台

Winform、WPF

#### 6.2.1使用 Direc2D 加速图形绘制

Direct2D 包装器，托管C++

### 6.2 Linux平台

Mono.Form

### 6.3移动平台

Xamarin

### 6.4 Web平台

Core Razor Page

## 定时作业

### 组织结构

JobHost

JobTask

JobTask2

JobExecutor

JobExecuteEnvironment

JobEnvironment

### 触发模式

一个完整的触发模式总共有， 秒 分 时 天 月 年，六个字段，不同字段可以使用的模式符也不同，

只有指定时间与每个字段上的模式都匹配，才会计为有效触发时间。

首先先对模式符做一个说明：

\* 表示不在此字段上加任何限制，也就是说如果使用 ” \* \* \* \* \* \* “ 则触发模式每秒都会被匹配。

此符号可以在任意字段上使用，不可与其他模式符一起使用。

/ 表示从指定字段值开始，每间隔一定的字段值，就匹配一次，例如在秒上使用 ”2/10“ 表示

从第二秒开始，每过 10 秒，匹配一次，不可与其他模式符一起使用。

^ 表示开始值，例如在 秒字段 上，此值为 0，而在 天字段 上此值为 1。不可与其他模式符一起使用。

$ 表示结束值，例如在 秒字段 上，此值为 59，而在 天字段 上此值为当月的天数。不可与其他模式符一起使用。

, 使用此模式符，可以在字段上指定多个固定值，例如在 秒字段 上 ”2,5,10“ 表示在第 2 秒，

第 5 秒，第 10 秒，匹配一次触发模式。不可与其他模式符一起使用。

- 表示一个范围，例如在 秒字段 上 ”2-10“ 表示在 第2秒至第10秒内（包含2与10），模式都会被匹配。不可与其他模式符一起使用。

w 星期说明，此符号只能使用于 天字段 上，例如，”w3“ 表示每周三匹配一次模式，”w1-w5“ 。

此符号只能与 ”,“ 这个模式符混用。

## 8.辅助功能

### 8.1 工具类库

#### 8.1.1 绩效工具

#### 8.1.2 URT事务统计工具

#### 8.1.3截图以及图像文字识别工具

网络识别采用百度文字识别服务

<https://cloud.baidu.com/doc/OCR/index.html>

本地识别使用 Tesseract

<https://github.com/charlesw/tesseract/>

#### 8.1.4颜色选取工具

### 8.2 网络代理

### 8.3 配置管理

#### 8.3.1 本地配置

提供配置文件信息与类型实例的映射，简化配置信息的读写，在此组件中配置

文件与类型的映射是强制的，以便于统一管理配置信息（杂乱不堪、神出鬼没的配置

对系统的维护与部署造成大量的困难），开发人员可通过此组件详细说明其主要含义

并可视其敏感程度决定是否加密，或者设置其可见性。本地配置通常只由客户端管理，但可在远程管理平台中对指定客

户端的配置文件进行修改。

#### 8.3.2 可视化配置

开发人员对配置类型的定义、说明会显示在相应工具中，维护人员使用

组件提供的工具设置所有配置信息，并且开发人员还可为相应属性开发更为人性化的

设置工具。例如 “数据库连接字符串”。

#### 8.3.3 全局配置

行为与本地配置类似，但可被所有客户端读取，只能在管理平台中写入

### 8.4 知识库

### 8.5 客户端前哨

环境依赖程度最低

从管理平台获取客户端环境要求，并下载安装环境

负责客户端模块的升级、修复

收集、监控客户机信息

提供对客户端行为调用的支持

### 8.6博客系统

与 vs 集成在开发过程中随时记录经验、灵感一键推送到个人博客

### 8.7远程桌面

这里使用 <https://www.remoteutilities.com> 提供的工具以及相应的免费许可，并自行搭建 self-hosted server 提供对互联网远程的支持

## 9.调式工具

基于 [CLR Profiling API](https://docs.microsoft.com/zh-cn/dotnet/framework/unmanaged-api/profiling/setting-up-a-profiling-environment)

提供对CLR程序运行时线程、调用堆栈、对象信息的收集，以及对部分对象信息

的修改

支持远程操作

## 10.互联互通平台

提供对传统各项数据交互技术联通的支持，以及对服务模型的调度管理

### 10.1互联域

### 10.2调用转发

### 10.3负载均衡

## 缓存组件

### 跨服务器会话信息

### 热点数据的缓存

## 12.功能组件

分为服务端、客户端两部分，分别实现不同接口

通过适当的抽象，减少功能之间的关联性

提供功能之间的交互

## 权限管理组件

基于角色的访问控制

RBAC Server 通过服务模型开放服务接口

用户、角色、项目、资源