**Mini轻量级框架设计说明**

2017-12-26

**目录**

[导言 3](#_Toc502075639)

[概述 3](#_Toc502075640)

[范围 3](#_Toc502075641)

[简写与词汇 3](#_Toc502075642)

[参考资料 3](#_Toc502075643)

[系统结构 3](#_Toc502075644)

[功能描述 3](#_Toc502075645)

[物理结构图 4](#_Toc502075646)

[系统结构图 4](#_Toc502075647)

[数据库设计 5](#_Toc502075648)

[表结构图 5](#_Toc502075649)

[数据库设计说明 5](#_Toc502075650)

[框架设计思路 6](#_Toc502075651)

[数据缓存模块 6](#_Toc502075652)

[Session记录模块 6](#_Toc502075653)

[权限控制 6](#_Toc502075654)

[模块设计思路 6](#_Toc502075655)

[登录注销 6](#_Toc502075656)

[组织架构岗位管理 7](#_Toc502075657)

[用户管理 7](#_Toc502075658)

[角色权限管理 7](#_Toc502075659)

[字典选项 7](#_Toc502075660)

[数据库备份 7](#_Toc502075661)

[技术路线 7](#_Toc502075662)

[.net技术实现 7](#_Toc502075663)

# 导言

## 概述

该文档为工业互联网系统Mini轻量框架的设计以及实现思路进行编写。同时对于不同语言实现Mini轻量框架所采用的技术、不同点进行了总体的说明。

本文档仅限公司内部使用。

## 范围

文档通过数据库方面的设计、总体实现模块方面的设计思路、不同语言的技术框架设计等多个方向对Mini轻量框架设计思路进行编写。

对于文档的描述一切就简要明了进行。

## 简写与词汇

表 ‑ 简写和词汇

| **简写** | **定义** |
| --- | --- |
|  |  |
|  |  |
|  |  |

## 参考资料

# 系统结构

## 功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| **系统模块** | **功能点** | **说明** | **备注** |
| 框架功能 | 登录注销 | 登录以及注销系统登录。可能涉及到单一登录功能。 |  |
| 用户信息修改 | 登陆后修改个人信息、修改个人密码。 |  |
| 系统主题切换 | 切换系统当前的主题。 |  |
| ~~菜单样式设置~~ |  |  |
| ~~个人主页配置~~ |  |  |
| ~~常用功能设置~~ |  |  |
| 基本信息 | 区域信息管理 | 区域规划功能的管理。 |  |
| 系统管理 | 组织架构管理 | 对于公司的组织架构管理，树状结构数据。 |  |
| 岗位管理 | 组织架构下的岗位信息进行管理。 |  |
| 用户管理 | 对于用户的信息进行管理，如果是系统用户需要设置用户密码等信息。具有密码重置功能。用户岗位、部门可以进行设置。 | 做一个用户选择的通用功能 |
| 角色管理 | 角色以及角色权限进行管理。 |  |
| 菜单管理 | 对系统的菜单进行管理，系统、菜单、按钮 |  |
| 日志管理 | 对于系统记录的操作日志进行查看、删除。 |  |
| 字典信息管理 | 对于字典类型信息进行管理。 |  |
| 字典值管理 | 对于字典类型下的值信息进行管理。 |  |
| 数据库备份 | 系统数据库表的备份、还原操作。 |  |

## 物理结构图



## 系统结构图



# 数据库设计

## 表结构图



## 数据库设计说明

**用户登录表**：主要是对于用户的密码以及登录的时间IP等信息进行记录。用户登录表中，保存用户的登录状态信息，如果已经有用户登录了，则标记为已登录。用户注销时，修改用户登录状态。

**用户表**： 主要是对于用户的信心进行记录，用户表中的用户，可以不作为系统用户。根据用户登录表是否有对应的记录进行确定。用户可以指定为管理员，此时可以忽略用户的权限进行系统访问。

**用户角色权限表**：对于用户的权限，本系统还是通过角色来进行。所以需要经典的系统管理结构：用户—角色—系统权限。复杂的不再进行说明。对于角色表也包括岗位信息。为了方便后期将岗位作为角色来进行看待。

**组织架构表**：根据组织架构的类型的不同分为机构、部门。机构即哪个公司，哪个集团之类的。部门即为公司下的。

以上为系统权限管理所有功能涉及到的表信息。

**字典表-字典值表**：对于系统的字典信息进行管理的功能。字典管理的主要是一些功能的选项方面的功能。可以考虑根据字典的编号，获取所有字典信息。字典表为对于字典的信息进行管理，字典值是对字典下所有的值以及代码进行管理。字典以及字典值都是的数据结构。便于分组处理。

**数据库备份表**：对于有效的数据库备份文件信息进行管理。数据库的备份文件只能保存到服务器端指定位置。

系统日志表：对于系统的日志信息进行管理，这里对于登录操作日志都可以进行记录。

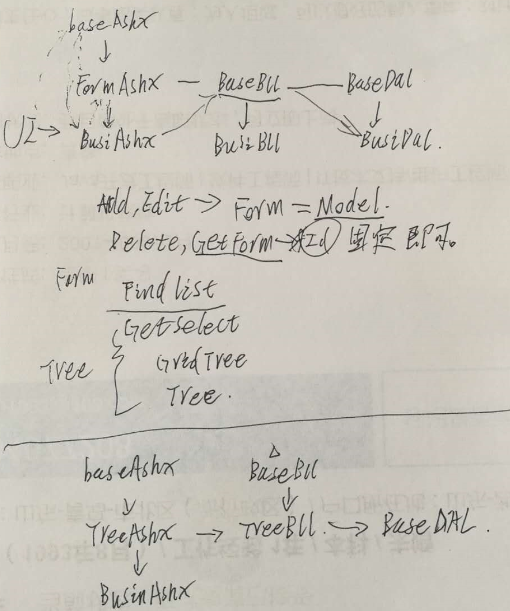
**报警信息表、消息信息表、区域表**：对于这三个表，与框架的关系不是太大，系统后期根据实际情况扩展框架模块时，才会用到。这里考虑后边对于具有不同模块功能的系统框架，自动生成不同的所需表格。

# 框架设计思路

## 基类设计

1. 自定义异常处理类，这里可以不需要看作是基类。可以直接用异常类做基类。
2. 实体类基类、定义实体必要的添加、修改、删除等操作的必要字段以及字段赋值，在进行使用时，可以直接调用对应的方法实现基类。
3. Ashx基类：文件请求的跳转进行梳理，异常的统一处理，返回方式的固定。

对于一些必要的验证也加入到该文件中，包括用户登录认证、授权认证等。



1. Page基类：这里就要求一般逻辑处理的方法都采用aspx页面进行实现。基类中定义的相对应的认证，与ashx相类似即可。
2. 自定义属性基类接口，强制实现Ignore属性、OnCheck方法。

在ashx等基类中，加入相对应的属性的检测方法，在Oncheck中做对于检测出现的问题中通过异常的方式或者返回值的方式进行返回对应的信息，便于基类中进行处理。这里暂时只是创建登录验证的属性、对于基类的属性，暂时先不加入。

**其他说明**

1. 对于基类中相关的方法可以定义成virtual类，一般与后续子类中有必要的话进行重写。
2. 对于一些方法可以定义成protect的，以便于子类可以调用，控制调用的作用域。

## 数据缓存模块

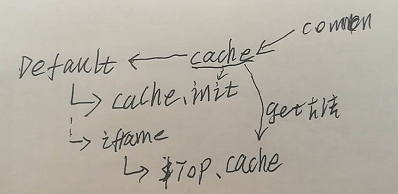
后台缓存

1. 对于用户登录信息，直接缓存在Session中。
2. 其他信息缓存暂时未考虑

### 前端缓存

#### 第一部分：关于前端cache文件的考虑

1. cache对象通过requirejs进行引入。



借用requirejs的声明方法生命cache对象，但是为了在iframe中可以通过top获取，要在define之外声明一个全局的变量，作为cache数据的载体。

Define中定义cache具体的内容以及初始化和调用的方法。调用方法直接通过common中的变量$top访问。

在iframe中调用时，添加cache的引用，直接调用cache中的方法即可。

这样在使用时，require中引入cache文件，对cache文件进行操作。

1. cache对象作为外部的js引入，只是调用requirejs的引用。

这样在defalut中直接引入cachejs，然后调用初始化方法，初始化cache对象。

cache对象同样是要全局的。

在后边调用是，直接使用top调用cache中的内容。

**最终确定**

在defalut中还是添加cache模块，然后Window对象预留一个外部变量。

Defalut中引入cache方法，初始化。

后边调用时，直接使用top调用顶层的预留的外部变量即可。

## Session记录模块

个人session

共享session

## 权限控制

### 功能权限

根据用户分配的菜单权限信息。在用户登陆时获取用户信息。在登录到主界面后，获取前台缓存时，将用户的菜单权限一并获取到前台，其中获取用户权限时要去重处理。

可以获取到的菜单都是该用户所具有的功能权限。用户没有的权限根本获取不到。

然后将获取到的模块，菜单等信息直接渲染到界面进行展示相应功能即可。

功能权限的区分，对于功能需要考虑是否设置“虚拟功能”，即该类功能也需要单独的权限控制，但是，在系统界面中不需要进行列表的展示。

### 操作权限

在用户登录后，对用户的按钮权限进行获取，直接存放在系统缓存中。

然后在进行相应的功能时，对于用户的权限内容进行判断。权限判断可以考虑在权限需要配置的按钮上设置上一个固定的属性，然后判断在进入界面后，直接根据这个特定的属性，检测该按钮是否具有权限。对于有的则展示，对于没有的按钮直接删除掉即可。

特殊属性设置两个，一个正常属性：

|  |  |
| --- | --- |
| btnauth | 值为true和false， 用于判断该按钮是否属于权限控制的按钮。 |
| authcode | 值为数据库设置的按钮权限的编号。用于对该按钮进行权限判定用。 |
| id | 在authcode未设置时，作为授权验证使用的编号。 |

### 数据权限

暂时没有想法，根据实际的业务要求再考虑添加即可。

## 授权认证

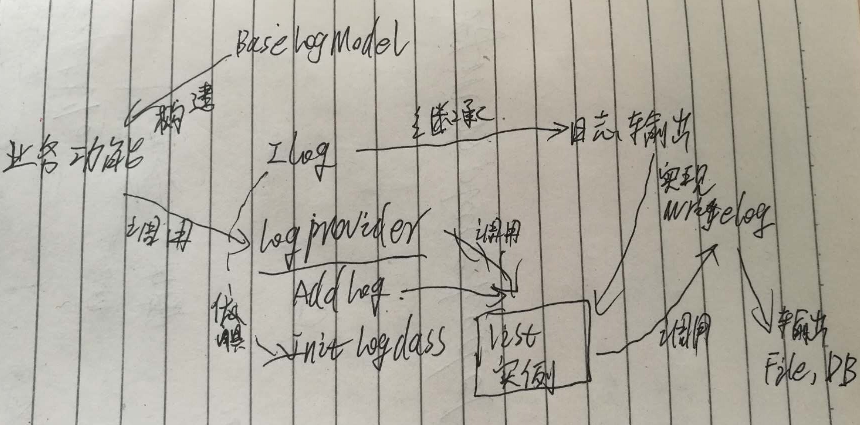
1. 系统加入授权认证管理功能，在管理界面提供机器码生成功能。对于当前设备的机器码进行生成，机器码根据用户机器的mac地址、Ip地址进行AES加密，Ip做内容、Mac作为key实现，加密结果在进行16为的md5加密。作为机器码。
2. 创建授权生成软件。根据机器码、授权类型、最终日期、用户量等信息，编辑生成对应的内容，日期去掉间隔符，授权类型前1位、之后机器码、在之后用&符号分割日期与用户量信息。采用RSA进行加密处理，将公钥与加密后的内容一并写入到lic文件中。生成为lic认证文件。
3. 在程序端进行对应的解密，实现授权信息的获取。
4. 授权信息在系统已启动时就获取到了系统的授权部分的缓存中，然后在指定位置进行授权的认证。
5. 对于lic文件可以考虑采用固定的公钥和私钥，进行加密签名。
6. 对于lic文件，可以设置在系统启动时，对licence文件进行解析，将解析后的信息缓存到系统的后台，然后在用户登录时，进行licence的登录验证。

在系统快到期时给出相应的提醒，提前15天。在系统到期后，禁止用户登录系统。

在各页面的请求时，给出相应的到期提示，并返回登录界面。Ashx就不添加验证了。

## 日志输出

日志输出的思路：



框架提供基础的日志操作类的基类，后续日志输出的类需要集成该基类，或者直接使用该基类。基类中提供转换方法Copy,可以将该基类内容转化到子类中。

框架提供ILog接口，定义日志输出方式以及输出方法两部分内容。输出端需要集成该接口。并制定实现类日志输出的方式，这里的输出方式与系统框架的日志输出配置对应。对于实现该接口的类，在日志提供转换方法中会通过反射进行实例化，进而进行调用。

框架提供主要的调度类：LogProvider类。该类采用单例模式实现，保证日志输出的口只有一个；该类起到了一个呈上启下的作用。用于初始化所有该程序中集成了ILog接口的实现类，认定是进行日志输出的正式类。此外该类提供日志输出调用的公共方法，供需要日志输出的位置进行日志的输出。

对于后续如果需要优化日志输出的性能问题时，可以针对LogProvider类进行优化，主要是考虑多线程输出处理。

对与后续需要接入的日志输出等内容，可以直接调用logprovider的公共方法实现即可。需要添加输出的位置，直接集成Ilog接口，直接生成一个实现类即可。

# 模块设计思路

## 登录注销

### 系统登录

登录界面，做到UI与JS分离，为以后对于界面修改时，可以不影响业务逻辑。

打开登录界面，系统首先加载界面，然后对验证码信息进行获取。

**验证码获取**：获取验证码请求，后台生成验证码图片，图片加入混淆处理，验证码可为汉字。

后台生成验证码后，缓存在Session中，登录时进行判断验证码情况。

登录界面提供用户名、密码、验证码信息，登录按钮。

**登录表单控制**：验证用户名、密码、验证码为空，定位到最先的为空项；用户名不存在，定位到用户名；密码错误，清空密码定位到密码；验证码不对，清空验证码，定位到验证码。

登录提交，对密码以及验证码进行加密传输。加密方式采用MD5即可。

首先根据用户帐号获取用户信息，获取不到，直接报用户不存在错误。

根据用户信息，获取用户登录情况，验证密码是否一致，不一致，则报密码错误。

用户验证通过后，对用户的相关信息缓存到Session。

有必要的话可以扩展帐套之类信息。

## 组织架构岗位管理

## 用户管理

## 角色权限管理

## 字典选项

## 数据库备份

# 技术路线

## .net技术实现

### 文件结构

LT.Code: 系统公共类库。包括成熟的类库及自己封装的工具类。可以dll形式提供。

LT.Core：框架的核心类库。

BaseBLL: BLL业务处理层的基类。对初步的增删该查方法进行定义封装。对于日志输出进行封装。

BaseDAL：DAL数据处理层的基类。

BaseEntity：实体类的基类。对于实体的增、删、改固定的方法进行了封装。

CommonEntity： 使用到的公共的实体类。

CusException： 自定义异常信息。

Operator： 对于用户session信息管理类。

LT.DAL：数据层操作。其下直接跟着对应的模块即可。模块下分文件。

LT.BLL：逻辑处理层操作。其下直接跟着对应的模块即可。模块下分文件。

LT.Model：实体类层。其下直接跟着对应的模块即可。模块下分文件。

LT.WebApp：UI处理层。

Ashx：一般处理程序做的后台入口。下文件夹划分模块即可。也可以直接文件也可以。

Config：配置文件存放文件夹。包括系统配置、用户配置等文件。

Resources：系统的静态资源存放的位置。

css：自定义的css样式文件，对系统框架的样式进行补充。

img：使用到的系统静态图片。

icon：所有使用到的图标，自定义的图标。

js：使用到的静态js类库，以及自己封装的公共类库。requerid的类库以及配置也存于此。

WebApp：系统功能的所有界面以及js处理文件存放位置。包括框架的以及后边自己添加的功能界面。

Default.aspx：默认的系统入口界面，框架的主界面。

Login.aspx：登录界面，与登录主界面脱离。

Global.asax：系统入口扩展，可以加载系统启动时事件。