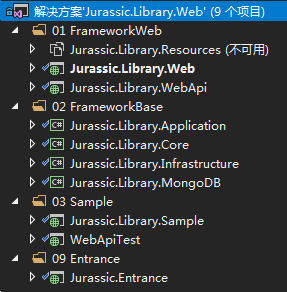
# 框架结构



Jurassic.Library.Resources 已废弃(静态资源文件)

Jurassic.Library.Web 框架Ui页面层(提供所有框架页面的视图及静态资源文件)

Jurassic.Library.WebApi 框架WebApi服务层(提供基础服务的动态生成)

Jurassic.Library.Application 框架服务层(主要处理业务逻辑的中间件)

Jurassic.Library.Core 框核心业务层(提供所有业务相关的处理及业务实现)

Jurassic.Library.Infrastructure 框架基础支持层(提供EF CodeFirst数据库动态生成)

Jurassic.Library.MongoDB 框架MongDB数据持久化基础服务类

Jurassic.Library.Sample 框架例程序

WebApiTest 框架WebApi测试页面

Jurassic.Entrance 框架入口程序集样板

# 项目Web层目录

项目的启动项目是 “**Jurassic.Library.Web**” 该程序集中各个目录的内容释义:

Abp 主要提供Abp框架相关的静态js以及css文件

App\_Data 该目录通常只有Log文件夹存放系统日志

App\_Start Mvc路由以及静态资源注册、项目启动类等。

Content 框架所有的静态资源js、css、图片等。

Controllers 框架模块控制器

Localization 框架本地化json对象文件

Models 框架基础接口类重写

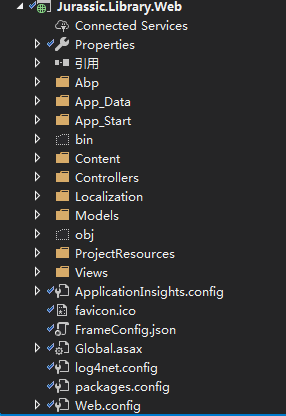
ProjectResources 该目录主要作用是示例，指在建立项目引用框架时如果有自己项目相关的静态资源文件，需要单独注册，并且把文件放在该自定义目录下（该目录名称不限定）。

Views 框架项目基础视图文件。（每个视图对应js文件为一组页面）

FrameConfig.json 框架一些基础配置信息，采用json格式。

log4net.config log4net日志管理组件的配置信息

注：其他未提交的文件或文件夹均为常规项目中必须或经常使用的不做详细解释 。



# 项目模板搭建

框架本身包含Web页面层，以及业务程序集两大部分。

**Web页面层**需要设置视图目录下所有文件为**嵌入式资源**，这样可以吧所有视图打包到程序本身dll中。



**业务程序集**只需要直接引用相关dll文件即可

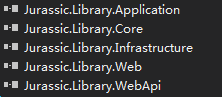
其中**Jurassic.Entrance** 程序集就是新建项目的示例:

此处根据现有项目为例

1. 首先通过解决方案建立一个Mvc 项目



1. 项目建立后引用



1. 添加Abp相关的配置信息主要包含以下几个地方.

Global.asax

注册启动项目以及日志组件

log4net.config

配置日志记录组件

**Web.config**

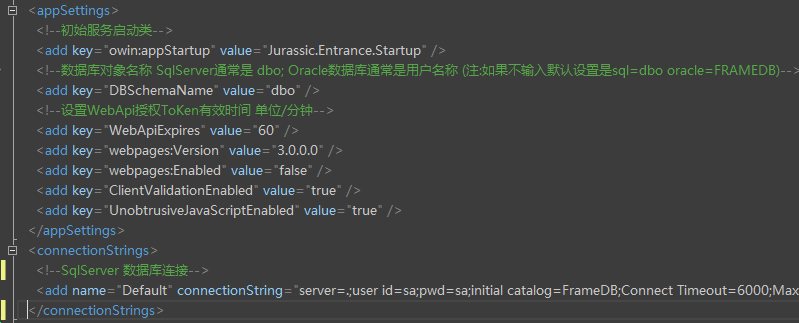
主要集中在appSetting中

**owin:appStartup**  设置启动类的命名

**DBSchemaName**  设置数据库的实例名称 dbo通常是sqlserver数据库默认,oracle具体根据数据库名称调整

**WebApiExpires** 设置Webapitoken有效时间 单位/分钟

其他配置均为ABP系统默认设置不做调整.



**EntranceStartModule.cs**

项目启动类,名称可以根据项目来修改

**Startup.cs**

启动类的配置注意与webconfig里面的配置一致如图:

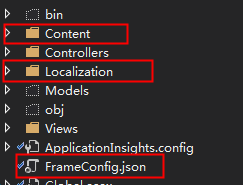




注:以上配置可以参考**Jurassic.Entrance**为模板。

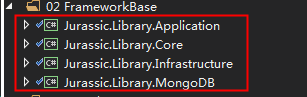
1. 复制原**Jurassic.Library.Web** 原框架的几个静态资源文件

该部分静态资源文件没有包含内容均为js或css因此未打包在dll文件中需要单独考本到新建的项目中即可运行。



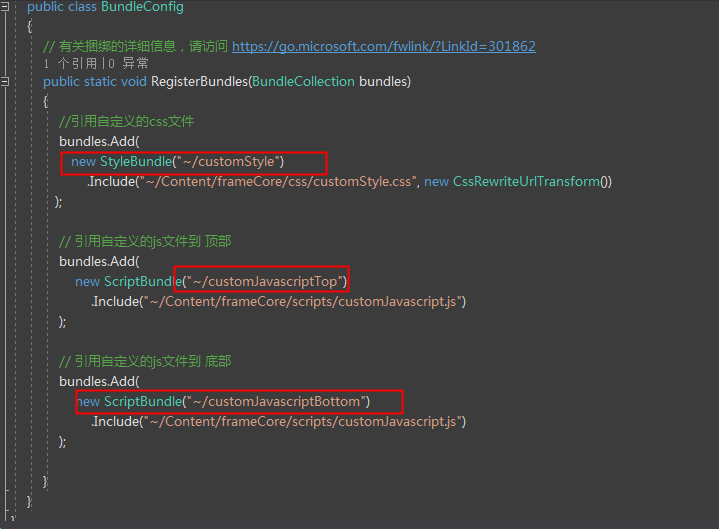
1. 项目其他业务程序集针对引用框架

实际项目开发过程中由于还会新增相应的业务层以及数据仓储层等等，这部分建议采用ABP的领域驱动的分层方式进行，并且可以根据项目情况分别获全部应用如下程序集，具体程序集的释义文档开头有解释。

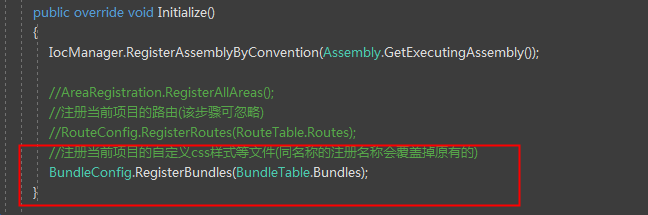


1. 项目静态资源文件(js.css)的注册

首先建立项目资源文件的目录,然后通过BundleConfig,注册相关js以及css文件.



完成注册后可以通过Abp启动类来进行注册该对象。



注：注册后的js根据注册名称不同会分为注册在页面<head>标签与<body>尾部两种方式。

# 项目扩展实现

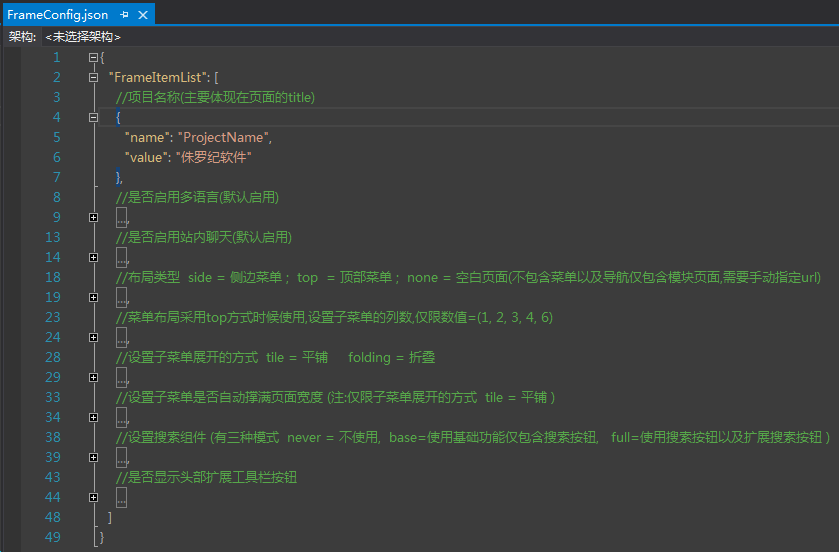
由于本框架采用的是基于Abp的实现，因此所有的重写以及扩展使用都可以通过abp官方api得到解释，此处针对目前部分已经完成的功能扩展做解释。

1. **配置信息设置**

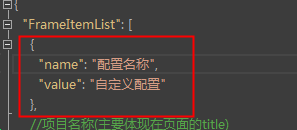
框架配置信息主要是针对整个系统运行做的一些设置信息，保存方式基本上是Key，Value方式。

例如配置文件FrameConfig.Json文件

该配置文件的配置项调整后直接影响的就是启动页面的一些变化，具体释义在配置文件有详细介绍，主要是针对页面布局的或本地化。



注：项目开发者还可以继续在配置文件上添加属于自己项目的配置信息添加方式如图：



程序中获取配置信息有两种

C# 代码

SettingManager.GetSettingValueForApplicationAsync(“配置名称”)

Js 代码

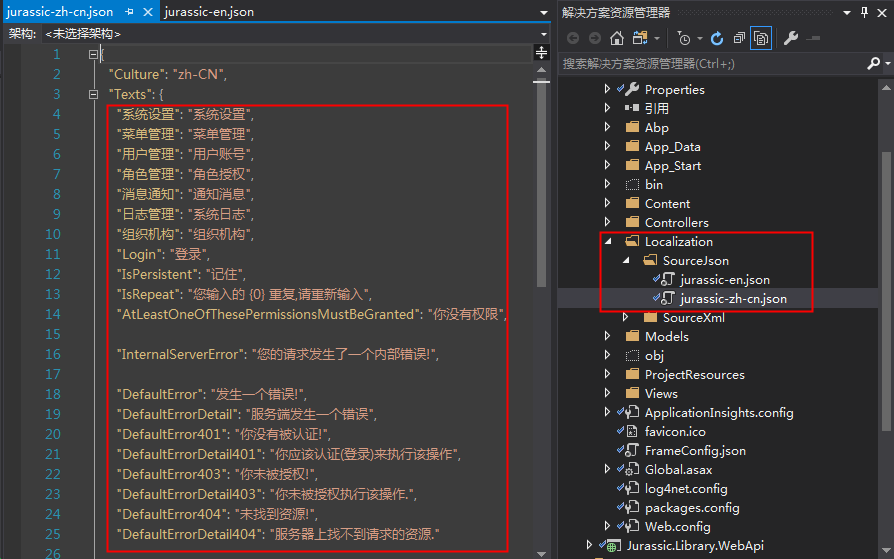
abp.setting.get(“配置名称”）

具体更多关于配置信息的使用方式可以参考

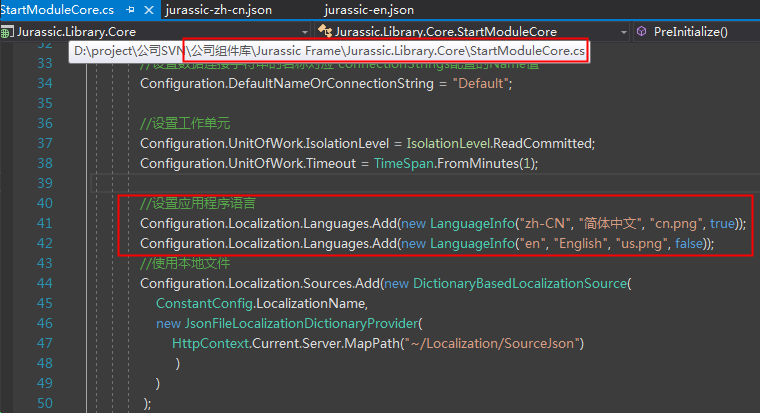
<https://github.com/ABPFrameWorkGroup/AbpDocument2Chinese/blob/master/Markdown/Abp/2.5ABP%E5%85%AC%E5%85%B1%E7%BB%93%E6%9E%84-%E8%AE%BE%E7%BD%AE%E7%AE%A1%E7%90%86.md>

1. **本地化处理**

框架已经提供一种本地化多语言的方案,该方案支持中,英两种语言,需要配置相应的K-V值,具体配置文件如图：

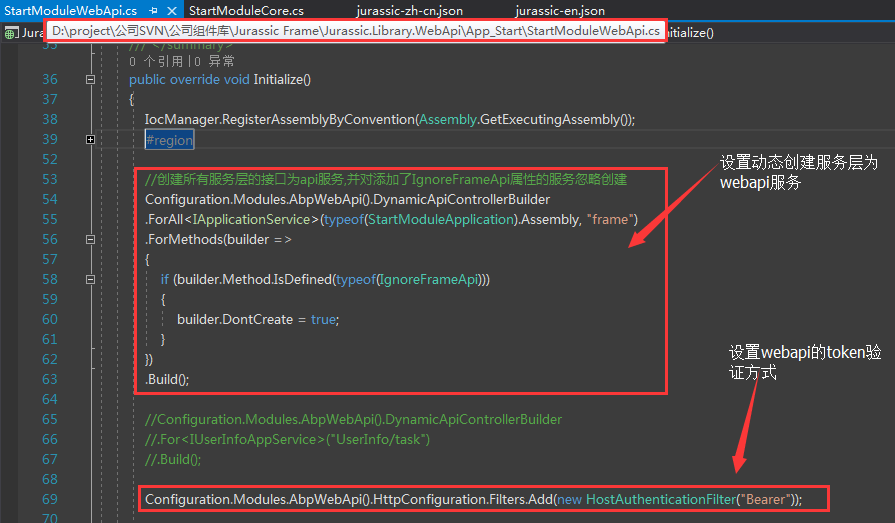


**注：**改配置信息目前未支持数据库保存，因此保存方式是本地json文件，如需添加第三种语言目前可以通过添加新的语言项目方式完成，并且新增一个新的json文件即可，json文件名称的后缀需要与代码中的参数字符串相同，例如zh-cn，添加方式如图：

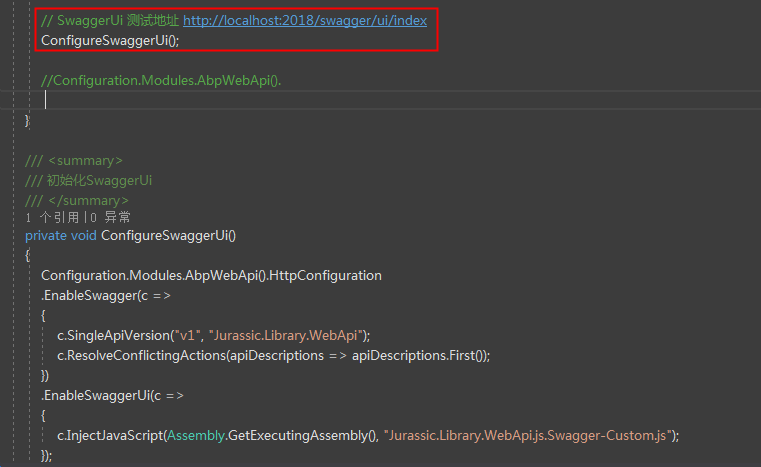


1. **动态api的创建**

具体配置信息时 **Jurassic.Library.WebApi**程序集中，如图：



注: 并且绑定了一个用于测试webapi的ui组件,输入对应项目的启动地址即可,具体地址参考图中红框部分



1. **自定义基类的使用**

Abp开发的模式中分为 web、webapi、服务层，因此每一个业务层都会有相应的父类作为继承，目的是为了方便常用接口的调用，而框架在此基础上添加了一层继承便于后续扩展，因此在这些业务层建立业务类的时候可以继承如下这些类

**Web 需要继承 FrameExtAbpController**

**WebApi 需要继承 FrameExtAbpApiController**

**服务层 需要继承 FrameExtApplicationService**

注1：这些类均继承了abp 的父类对象，因此使用时候后与直接使用abp父类对象无差异。

注2：继承了框架自带的这些父类将可以调用重写了的session对象，AbpSession 对象是在Abp的基础上进行了重写，添加登录用户的一些信息便于项目中随时获取。

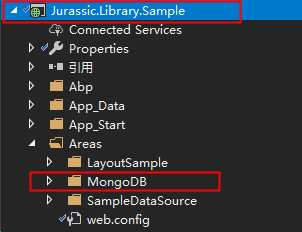
1. **关于MongoDb数据持久化的扩展**

首先需要引用Jurassic.Library.MongoDB程序集dll，以及mongoDB所属的驱动dll。

然后与数据库的交互操作与以往数据仓储开发类似都需要继承对应IRepository接口，此处进行处理可以直接继承MongoDBRepositoryBase，具体使用如图：

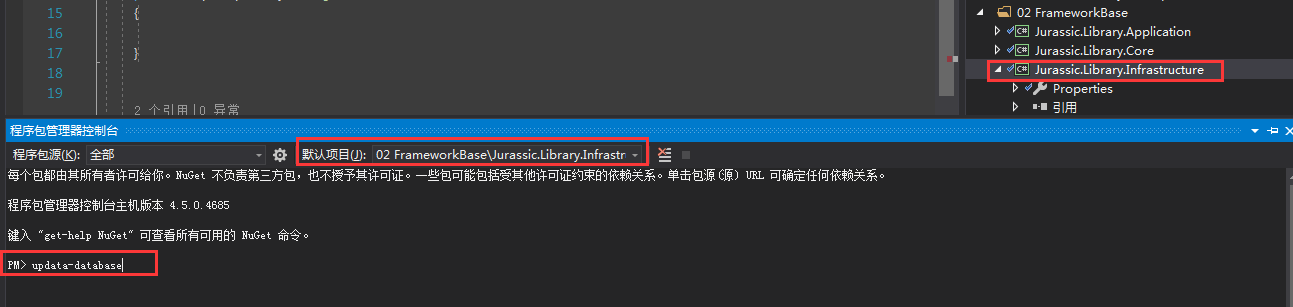


注：由于MongoDB并非结构化数据库方式存储，因此也可能存在直接存储json对象的方式，无需通过对象方式进行交互数据，因此同时也可以直接通过MongoDB原生驱动的方法进行数据持久化。详细示例参考：



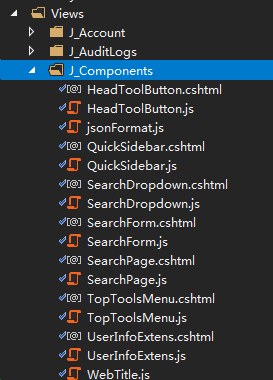
1. 初始项目数据库的生成

首先配置好项目webconfig数据连接地址然后通过” Jurassic.Library.Infrastructure”程序集完成数据库的动态生成,具体生成方法如图,打开**程序包管理控制台,**如图设置.执行 updata-database

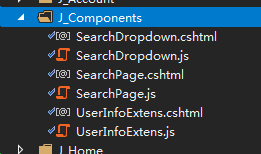


1. 框架的组件重写

框架布局采用组件模块化，因此所有模块以及组件都可以进行重写，例如大部分组件都写在视图目录的J\_Components文件夹下。



如需在各自项目中重写只需要在各自的项目的视图中添加J\_Components目录，然后新增相同的视图名称以及js 文件即可，具体实现就会按照项目该目录下的视图文件进行渲染。

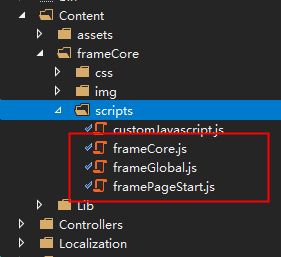


示例：我们需要重写原有的登录页面的logo文字信息，因此我们需要在项目视图（Views）目录下新建J\_Components目录然后，新建WebTitle.js（因为该组件只有js文件没有视图文件所以不需要）然后修改部分文字描述即可在项目启动时候默认采用新增加的组件覆盖原有框架里面的方案。



1. 框架相关Js

框架核心js包含如下几个js文件



FrameCore.js 框架核心js框架,主要包含路由以及自定标签等(不推荐把项目js加入到该js文件)

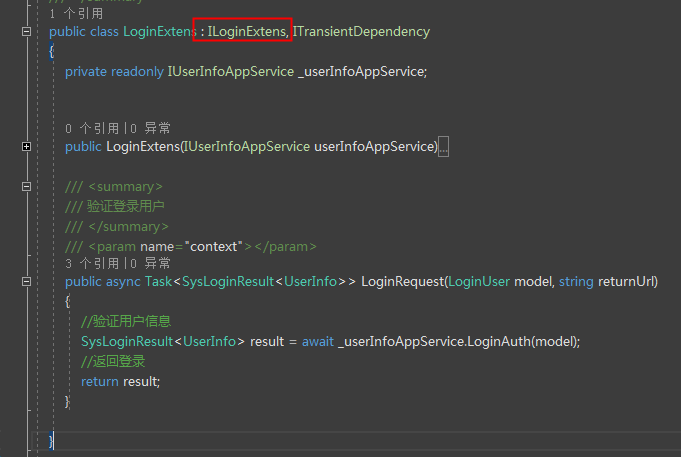
FrameGlobal.js 框架公共js内库,包含常用转换函数以及一些js工具类的函数,具体使用详见js注释

framePageStart.js 框架模板页面的js文件,主要提供给系统主界面的js数据业务支持。

1. 登录验证的重写

对于部分项目登录的验证可能会采用不同的方式情况，系统支持对登录账号密码的验证的重写，只需要在项目中实现ILoginExtens接口，并实现LoginRequest即可，参数包含登录用户的账号及密码和返回路径。

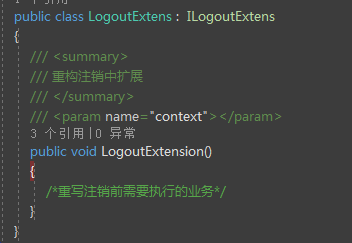
注：返回对象必须包含登录用户基本信息。



1. 用户注销的扩展

框架提供了关系用户注销前的扩展方法，同样只需要实现接口ILogoutExtens即可。

注：该方法只是在注销前主要做的事情，并不影响注销用户。

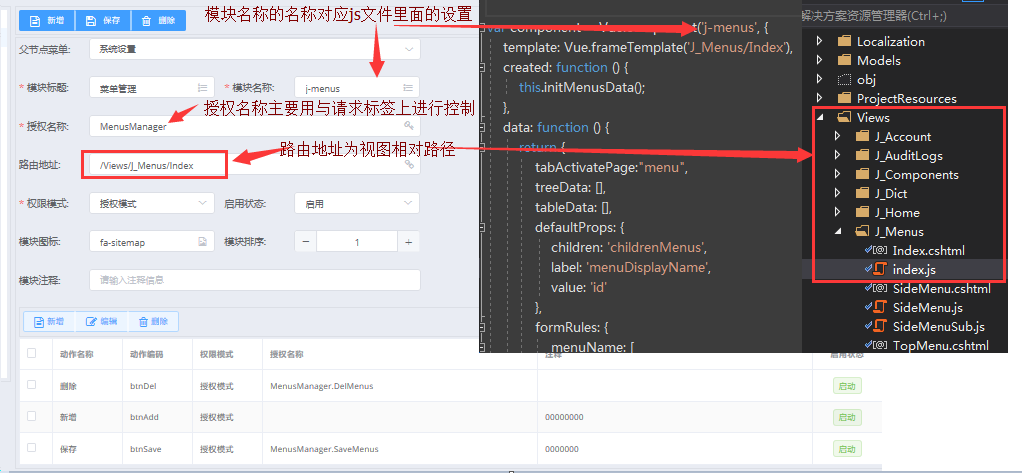


1. 续

# 框架基础功能介绍

1. **模块管理**

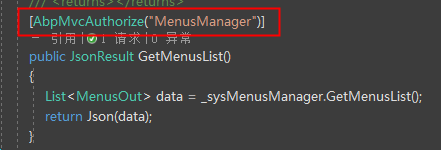
模块的配置与视图对应关系



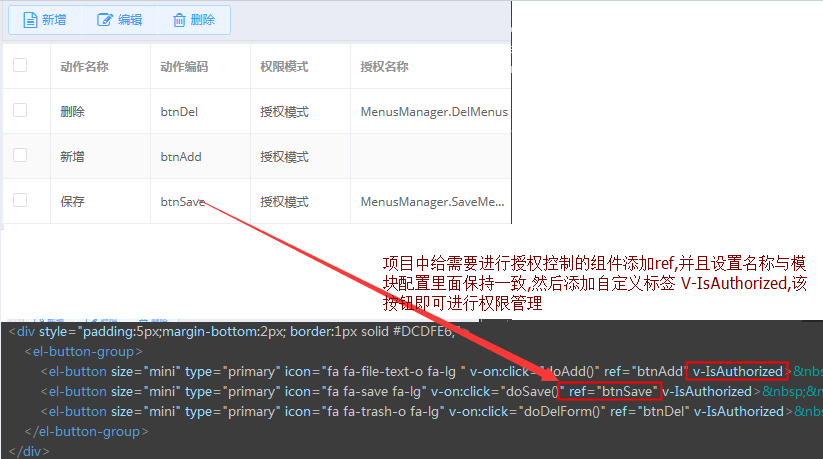
模块名称 => js文件中组件名称

路由地址 => views视图下视图文件的相对路径,不需要输入后缀名称(建议视图名称与js名称一致)

授权名称 => 主要用于后台代码控制权限时候标签的名称例如请求上的标签:



关于模块按钮以及组件级别的控制需要首先配置模块的子项信息,具体配置与代码的关系如图:



注1:未添加ref="btnAdd"或 v-IsAuthorized 的组件将不会触发权限控制.

注2:授权名称的使用通模块授权方式一样属于请求上的标签名称。

注3：权限模式分，授权模式（有权限管理）、登录模式（只有登录用户可以）、开放模式（所有人可以使用），该设置同时适用于模块与按钮。

1. **用户管理**

用户管理主要是对系统登录用户的维护.包含基本的新增,编辑功能,以及密码重置

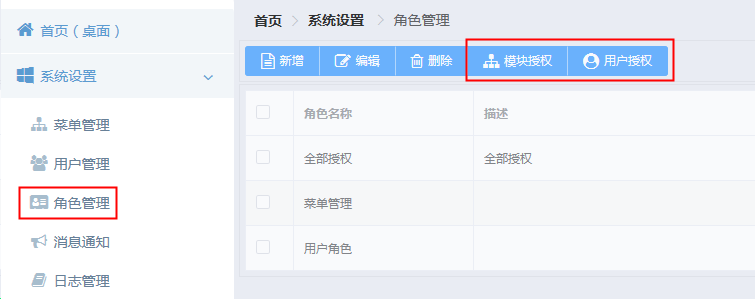


注1:删除为逻辑删除

注2:重置密码后新密码是888888

1. **角色管理**

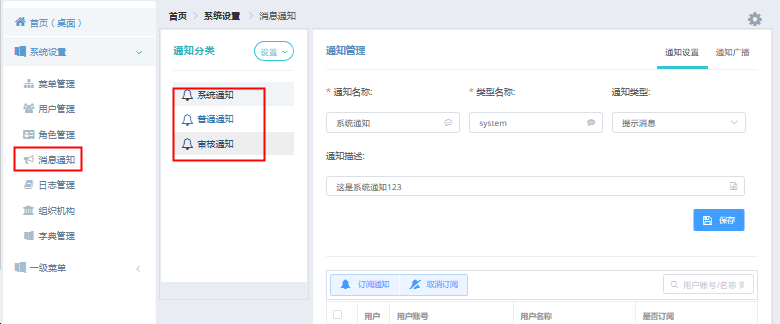
角色管理主要是设置角色与模块,角色与用户之间的关系,以及角色基本信息



注1:角色删除为物理删除,并且会删除该角色所关联的模块与用户关系。

1. **消息通知**

消息通知主要是设置消息的类型以及所订阅的用户，并且可以进行站内消息群发。



用户方接受消息可以通过如图菜单进行管理。





注：未订阅的消息类型用户是无法收到消息推送的。

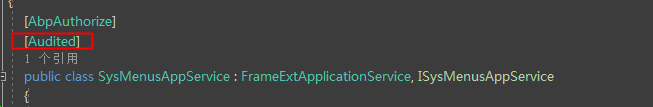
1. **日志管理**

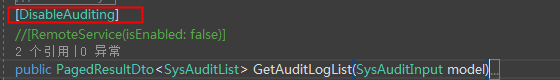
日志管理是追踪系统运行过程中所有的请求以及请求参数的信息，并且详细显示



注1：给类或者请求添加标签即可监听该请求或者该类下面所有请求的日志，使用方式如图：

给类添加标签，该类下面所有请求都会记录日志，





注2：如果设置标签“DisableAuditing” 将不会监听类或请求的日志信息。

1. **组织架构**

组织架构主要是系统建立业务关系的模块,主要根据项目来进行维护.



注：机构类型可以根据节点区分为部门或公司，主要目的是业务使用过程中明确各个节点级别。

1. **字典管理**

主要对一些常用标准数据的维护，例如性别、类型等等这些字典类型的数据。



注：通常**字典编码** 表示combox类型组件的code值，**字典名称**表示combox类型组件的Text值 ，而**字典值**作为预留的扩展使用。

# 续