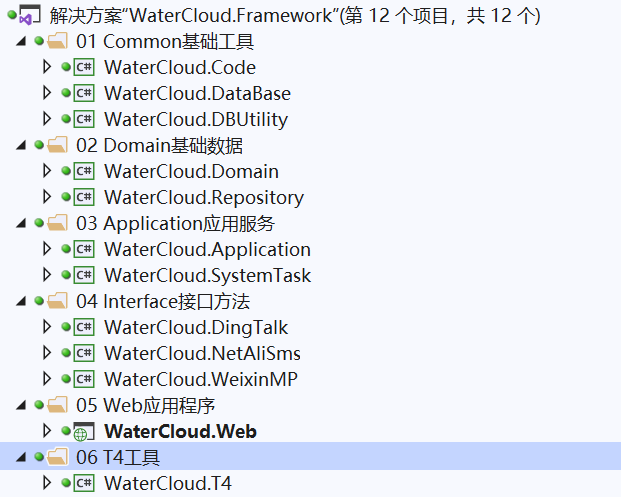
# WaterCloud开发框架

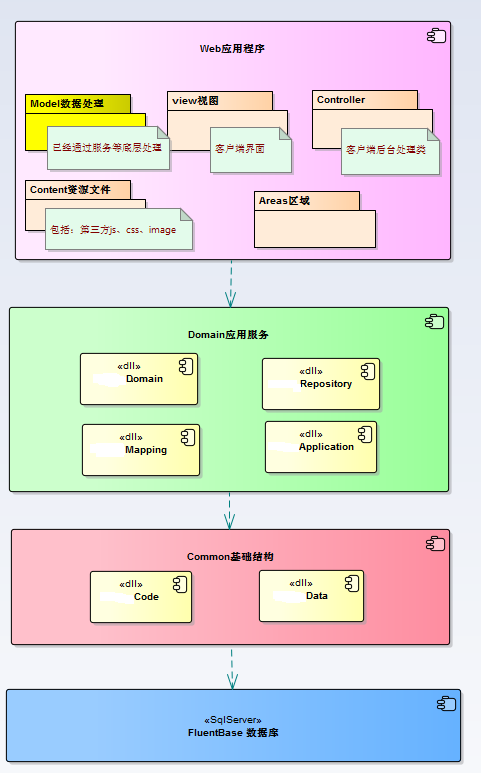
项目地址：<https://gitee.com/qian_wei_hong/WaterCloud>

## 一．解决方案简介：



1. WaterCloud.Code 底层核心类（存放工具类和底层代码，可编绎成dll提供）。
2. WaterCloud.Data 数据层（数据库工具栏、ORM工厂类，可编绎成dll提供）。
3. WaterCloud.DBUtility数据库工具（可编绎成dll提供）。
4. WaterCloud.Domain 领域层（对象类、接口）。
5. WaterCloud.Repository 数据访问（具体实现）。
6. WaterCloud.Application 应用（有点类似业务逻辑层） 。
7. WaterCloud.SystemTask 定时器。
8. WaterCloud.Web Web应用程序。
9. WaterCloud.T4 。
10. 接口略。

可以根据下图对各层进一步理解（取消Mappingying数据库映射）：



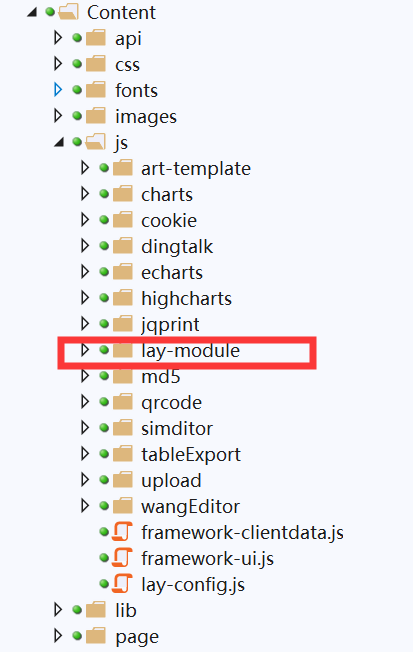
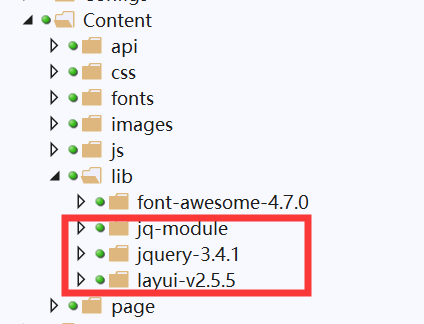
## 二．框架主要运用技术：

### 1、前端技术（引用参考Shared/\_Index.cshtml）

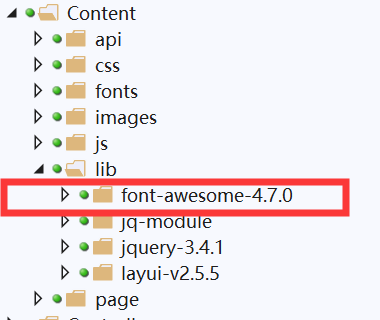
layui官网地址：<https://www.layui.com/doc/>;

layuimini官网地址：<http://layuimini.99php.cn/>;

#### (1) js框架：jquery-3.4.1、LayUI、LayUI mini（开源）。



#### 图标：Font Awesome 4.7.0。



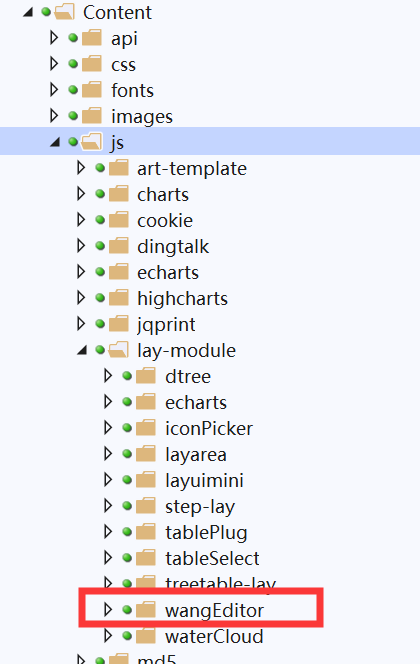
<http://www.fontawesome.com.cn/icons/bars/> 字体图标查询库



#### 客户端验证：LayUI verify。

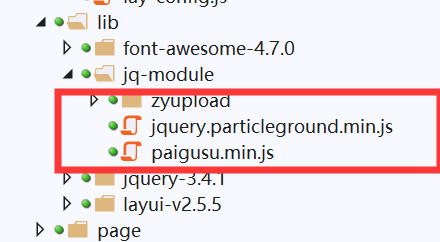
说明地址：[https://www.layui.com/doc/modules/form.html#verify](https://www.layui.com/doc/modules/form.html" \l "verify)

#### 富文本编辑器：开源wangEditor。



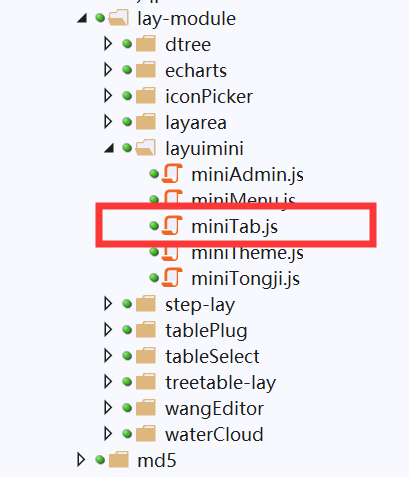
地址：<https://www.kancloud.cn/wangfupeng/wangeditor3/332599>

#### 上传文件：开源zyupload。

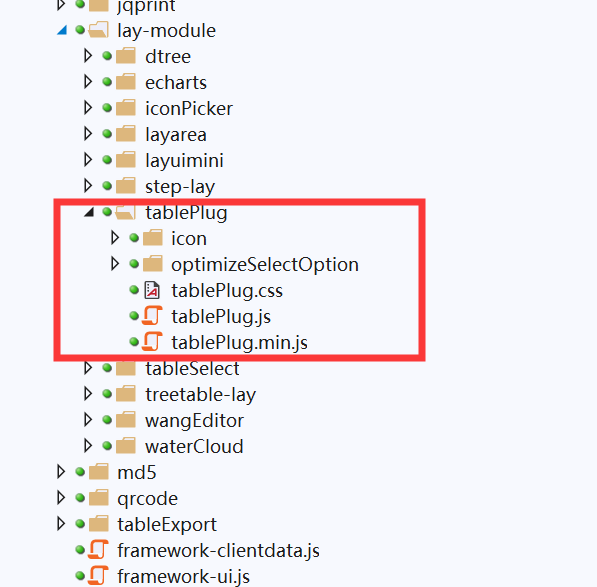


地址：<http://www.jq22.com/jquery-info14213>

#### 动态页签：LayUI mini miniTab。



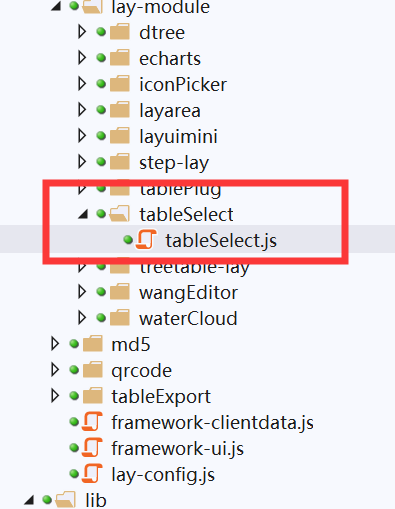
#### 数据表格：LayUI table、LayUI 开源 TalbePlug。



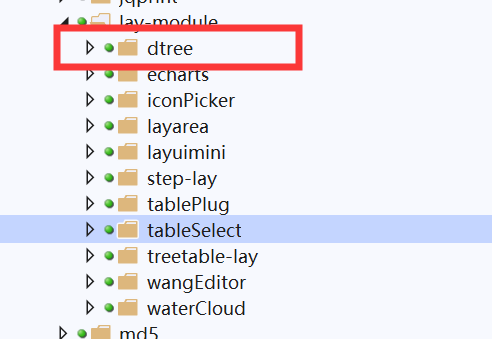
地址：

<https://sun_zoro.gitee.io/layuitableplug/testTableCheckboxDisabled>

#### 下拉选择框：LayUI select、LayUI 开源 TalbePlug(optimizeSelectOption)。

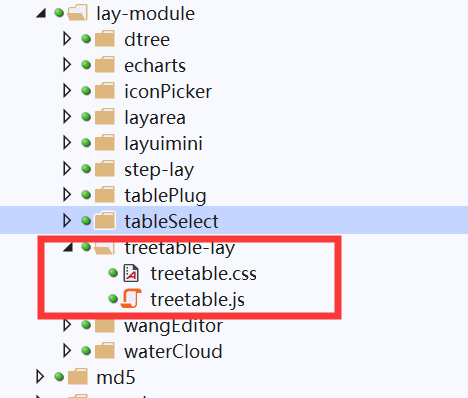


#### 树结构控件：LayUI 开源 dtree。



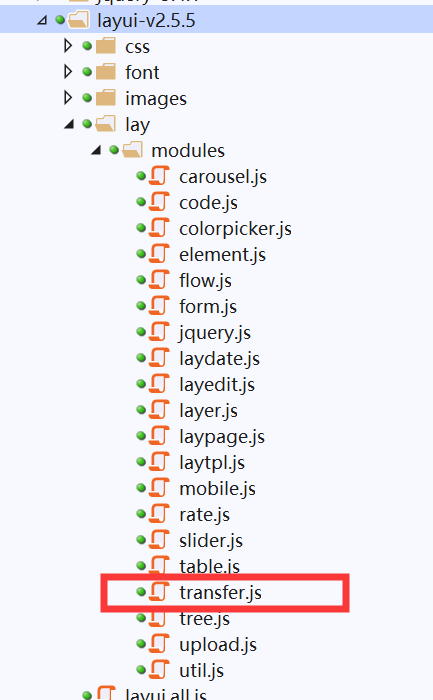
地址：<http://www.wisdomelon.com/DTreeHelper/>

#### 树状表格：LayUI 开源 treetable-lay。



地址：<https://gitee.com/whvse/treetable-lay/tree/master/1.x>

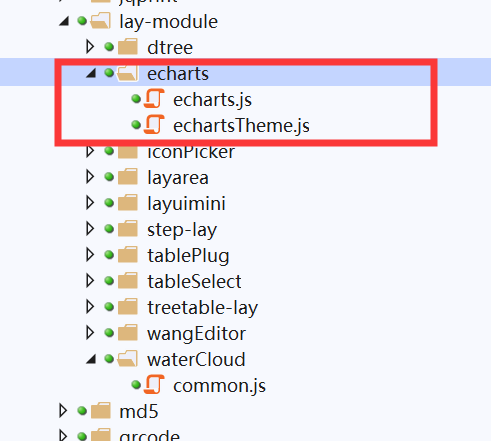
#### 穿梭框：LayUI transfer。



地址：<https://www.layui.com/doc/modules/transfer.html>

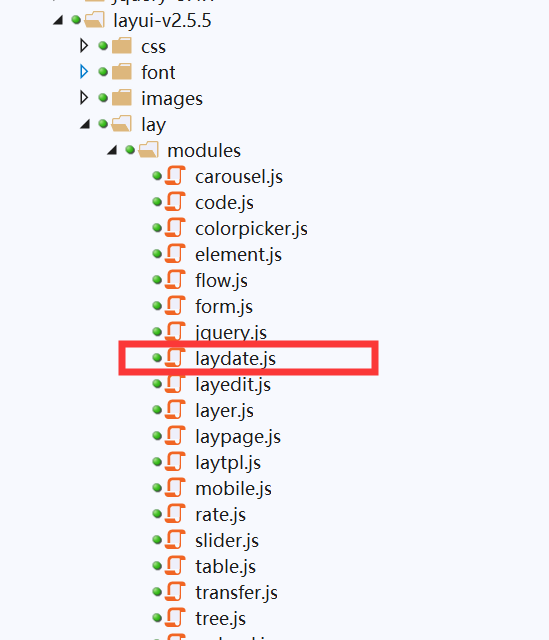
#### (12) 页面布局：LayUI、LayUI mini。

#### (13) 图表插件：echarts



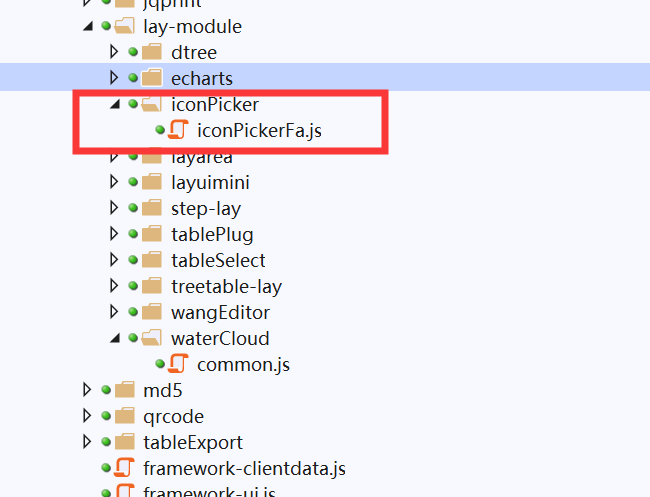
地址：<http://echarts.baidu.com/examples.html> （echarts官网图表）

#### 日期控件：LayUI laydate



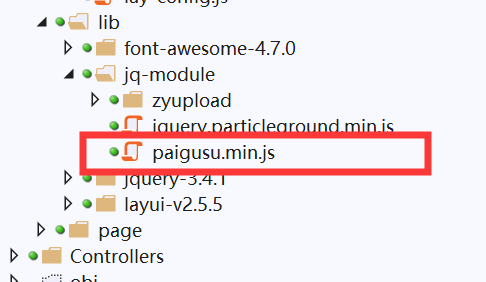
地址：<https://www.layui.com/doc/modules/laydate.html>

#### 图标选择：LayUI 开源 IconPicker



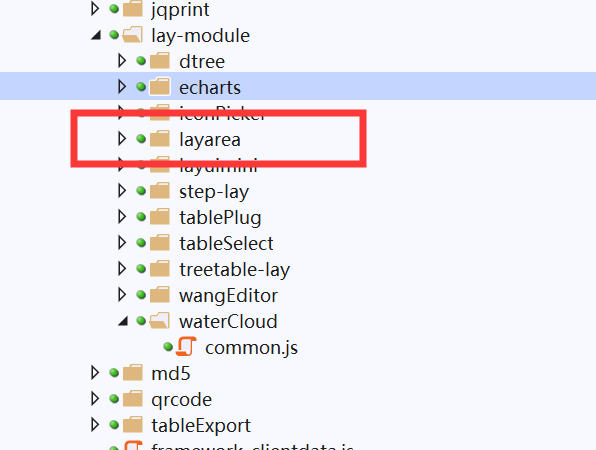
地址：<https://gitee.com/wujiawei0926/iconpicker>

#### 颜色选择：paigusu



地址：<http://www.jq22.com/jquery-info20030>

#### 省市区选择：LayUI 开源 layarea



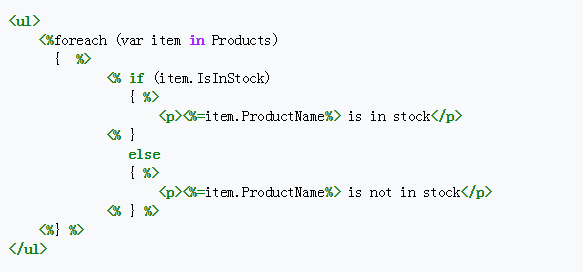
地址：<https://github.com/fesiong/layarea>

#### MVC View

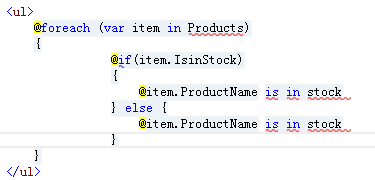
<https://zh.wikipedia.org/wiki/User:%E5%B0%8F%E6%9C%B1/%E6%B2%99%E7%9B%92/ASP.NET_MVC>

<http://www.jianshu.com/p/5f6156cacc76>

##### mvc 1.0-2.0版本试图以.aspx方式，如下写法



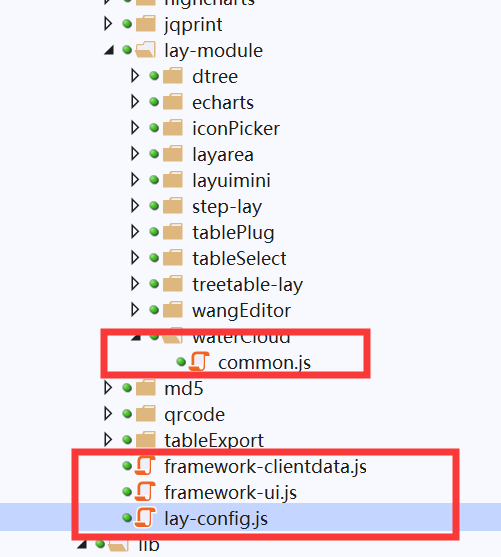
##### MVC v3.0 开始，试图添加了一种更简洁的写法Razor，如下写法：



##### 允许部分视图—可以在多个界面共用



#### 最重要（common.js、framework-ui.js、framework-clientdata.js、lay-config.js）



lay-config.js：layui模块化设置文件必须引用。

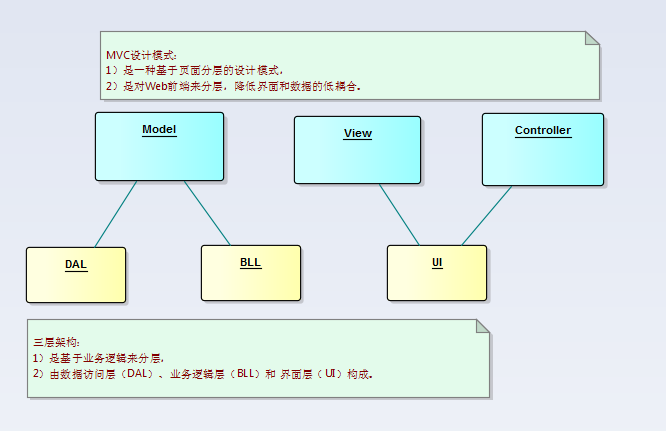
framework-ui.js ：jquery的一些方法。

common.js：封装了各种插件的方法供界面使用。

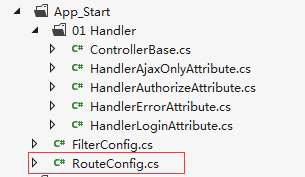
framework-clientdata.js：初次调用后端数据

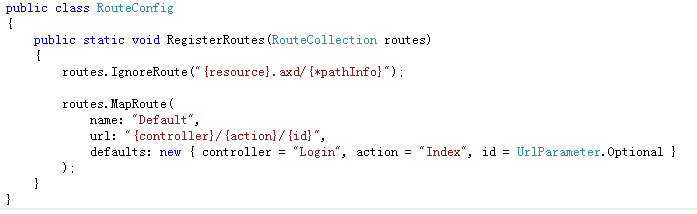
### 2、后端技术

#### （1）核心框架：ASP.NET MVC5



##### Mvc路由



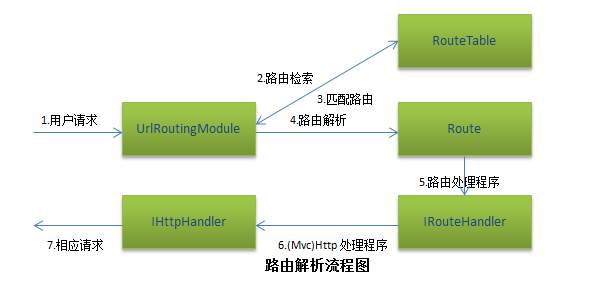


1. // 路由名称
2. // 带有参数的url
3. // 参数默认值 (UrlParameter.Optional-可选的意思) );

参考：<http://www.cnblogs.com/zeusro/p/RouteConfig.html>







如何在传统的Web站点中使用路由(一般情况下用户将传统站点转化为MVC站点的项目迁移过渡)。

主要包含以下两个步骤：

      1.常见实现IRouteHandler接口的WebFormRouteHandler类，返回实现IHttpHandler接口的实例化对象(实际上任何一个Page都是一个IHttpHandler实例对象)。

public class WebFormRouteHandler : IRouteHandler

{

public string VirtualPath { get; private set; }

//初始化虚拟路径

public WebFormRouteHandler(string virtualPath)

{

this.VirtualPath = virtualPath;

}

public IHttpHandler GetHttpHandler(RequestContext requestContext)

{

//创建实例化的页面对象

var page = BuildManager.CreateInstanceFromVirtualPath(VirtualPath, typeof(Page)) as IHttpHandler;

return page;

}

}

   2. 配置全局应用程序类(Global.asax)，实现路由到传统Web Form的映射。

//映射传统的web站点

routes.Add("show", new Route("blog/show/{author}", new WebFormRouteHandler("~/pages/show.aspx")));

routes.Add("compose", new Route("blog/compose/{year}/{month}/{day}", new WebFormRouteHandler("~/pages/compose.aspx")));

然后在default.aspx中添加两个链接：

<div>

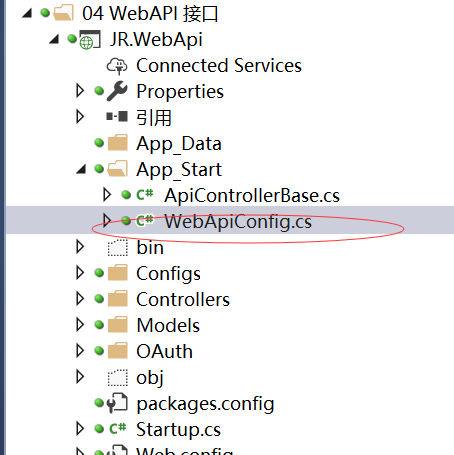
<a href="blog/show/miracle">blog/show/miracle</a>

<a href="blog/compose/2012/5/28">blog/compose/2012/5/28</a>

</div>

#### ~~（2）核心框架：WEB API~~

##### WEB API路由：

启动类：

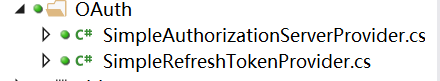


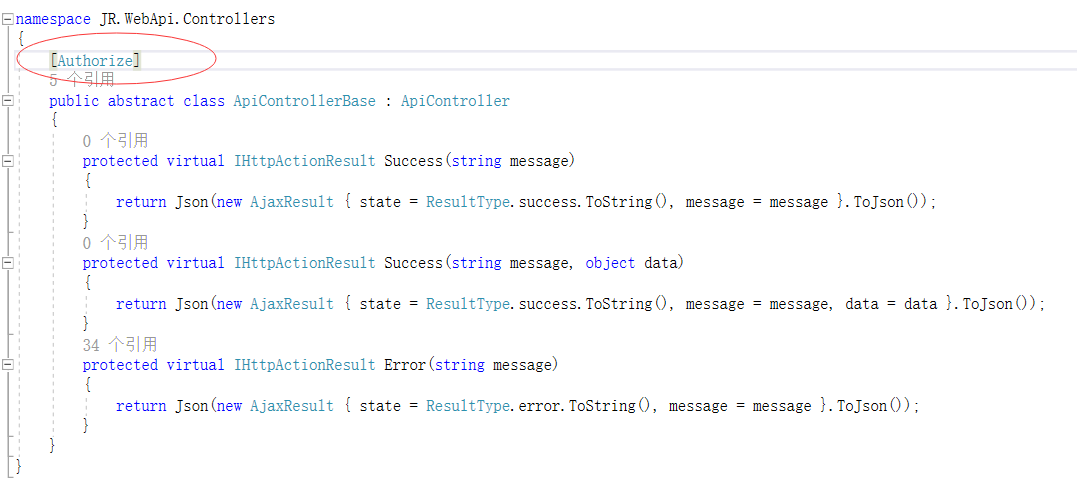
授权验证OAuth2.0（密码验证）

1、通过：

<http://192.168.1.17:51270/grant_type=password&username=username&password=password>获取Token

2、所有接口请求headers添加Authorization:bearer Token。







#### （3）安全支持：过滤器、Sql注入、请求伪造 (MVC特性类)

##### AuthorizeAttribute 类 多用于权限管理

我们可以重写AuthorizeAttribute达到自定义的权限管理，HandlerLoginAttribute

mvc中可以为Controller或Action添加定制特性实现登录验证【HandlerLogin】

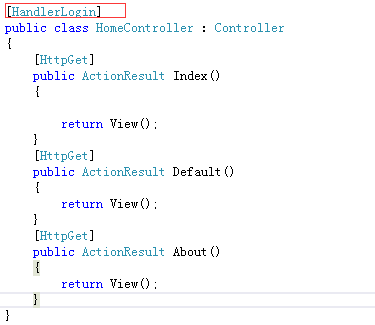
1. 每个action执行前都会先执行OnActionExecuting方法；（）
2. 2.FCL提供了多种方式来检测特性的存在，比如IsDefined、GetCustomAttributes方法等，IsDefined方法仅仅是判断目标有没有应用指定特性，而GetCustomAttributes方法会构造指定特性的新实例。
3. 继承AuthorizeAttribute登录特性，重写OnAuthorization方法，只需要给Controller或者Action给上标签即可。

参考：<http://www.cnblogs.com/qk2014/p/4804631.html>

<http://www.jianshu.com/p/5f6156cacc76>

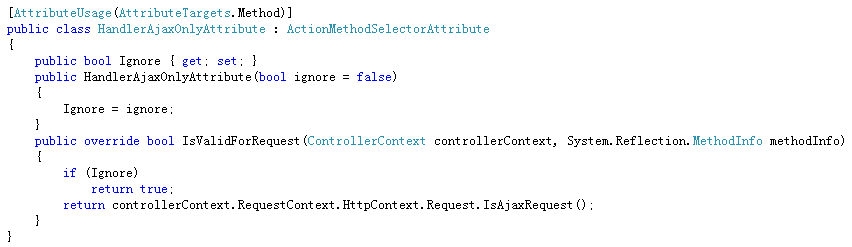
##### 使用：WaterCloud框架中使用第三种方式进行登录验证





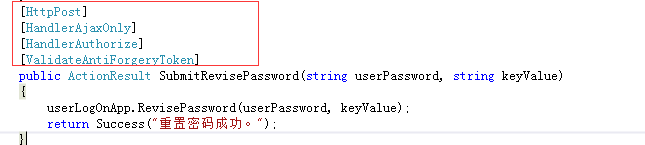
在Home 首页加载的时候，判断是否已经有登录信息，

给Conrtrller添加【HandlerLogin】特性标签进行登录验证，对应的Controller下所有的action都会执行登录信息验证。

给某个特定的action添加【HandlerAjaxOnly】特性标签，对此action进行Ajax调用。



一个Action可以有多个特性。

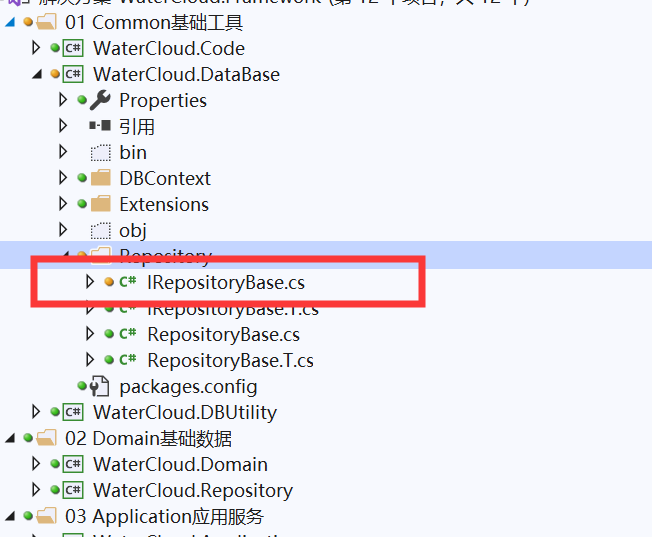


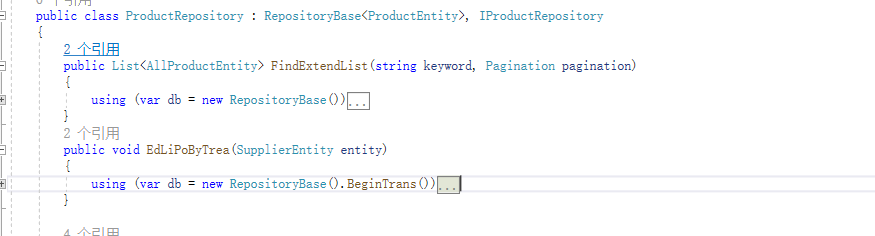
#### （4）持久层框架：Chloe.ORM

参考：<http://www.52chloe.com/Wiki/Document>

1、大部分操作都跟EF一样，因为框架封装过，增删改查都一样。

2、这里面带备注：





3、不使用事务（复杂查询）： using (var db = new RepositoryBase())

{

var query=db.Query<User>()

.InnerJoin<City>((user, city) => user.CityId == city.Id)

.InnerJoin<Province>((user,city,province)=>

city.ProvinceId == province.Id);

}

dbContext就是我们的db,其他都一样。

4、使用事务（复杂操作）： using (var db = new RepositoryBase().BeginTrans())

{

各种操作

db.Commit();

}

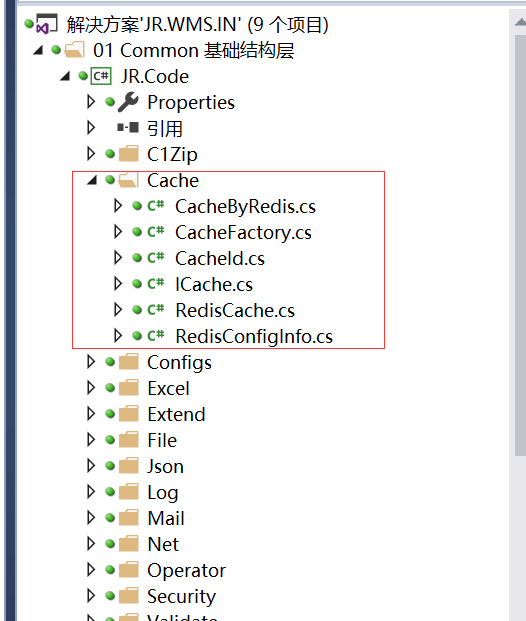
官网地址：<http://www.52chloe.com/Wiki/Document/3324809512287731712>

#### （5）服务端验证：实体模型验证、自己封装Validator

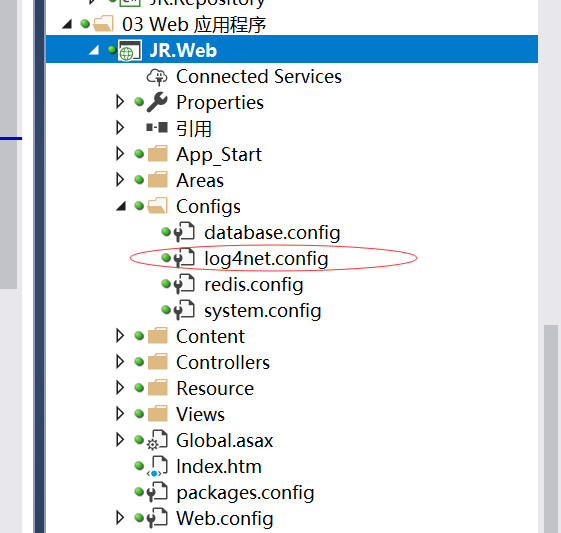
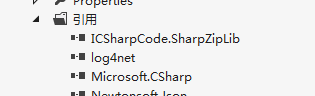


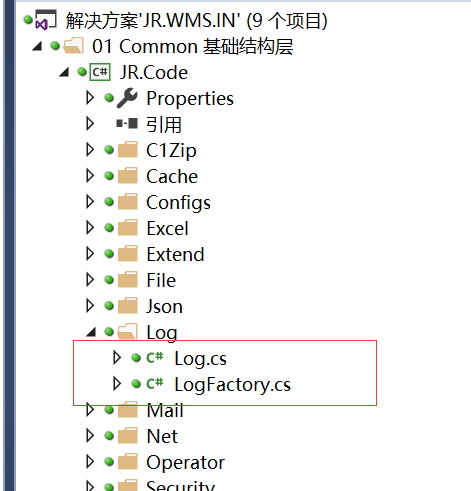
#### 缓存框架：微软自带Cache、~~Redis~~

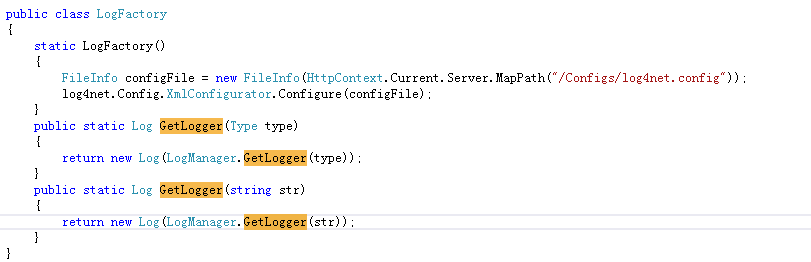
Redis等有空加上

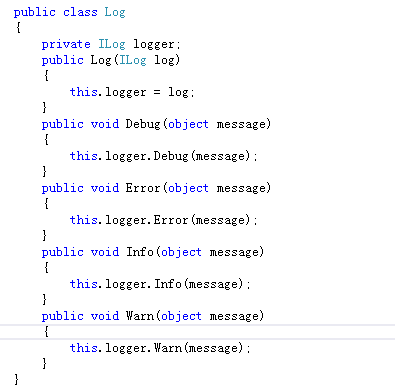


#### （7）日志管理：Log4net、登录日志、操作日志

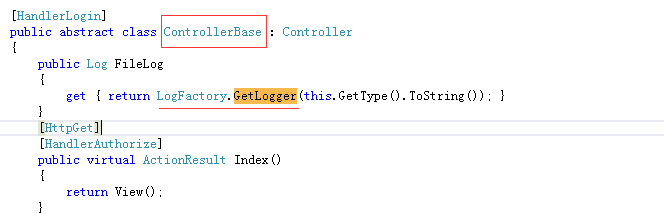
 

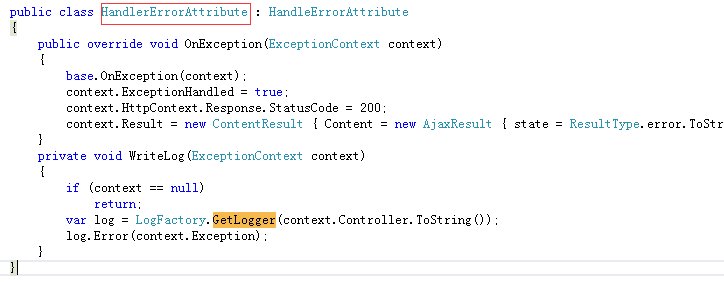






WaterCloud中调用：





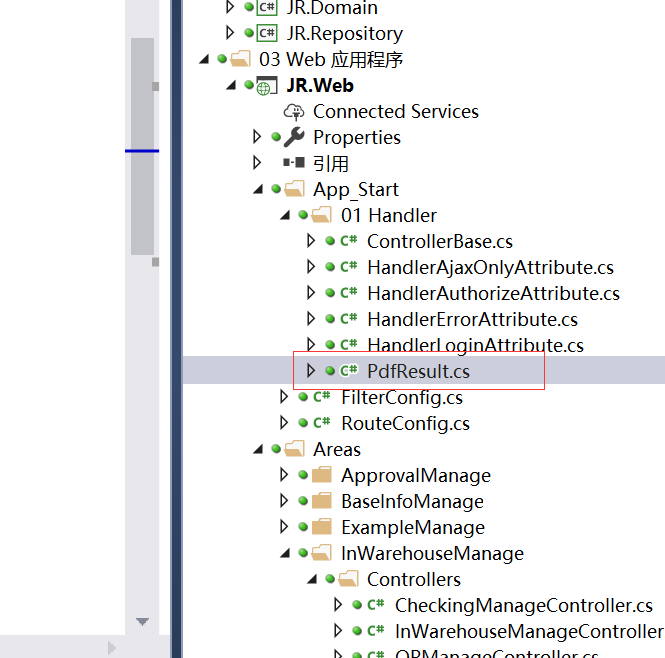
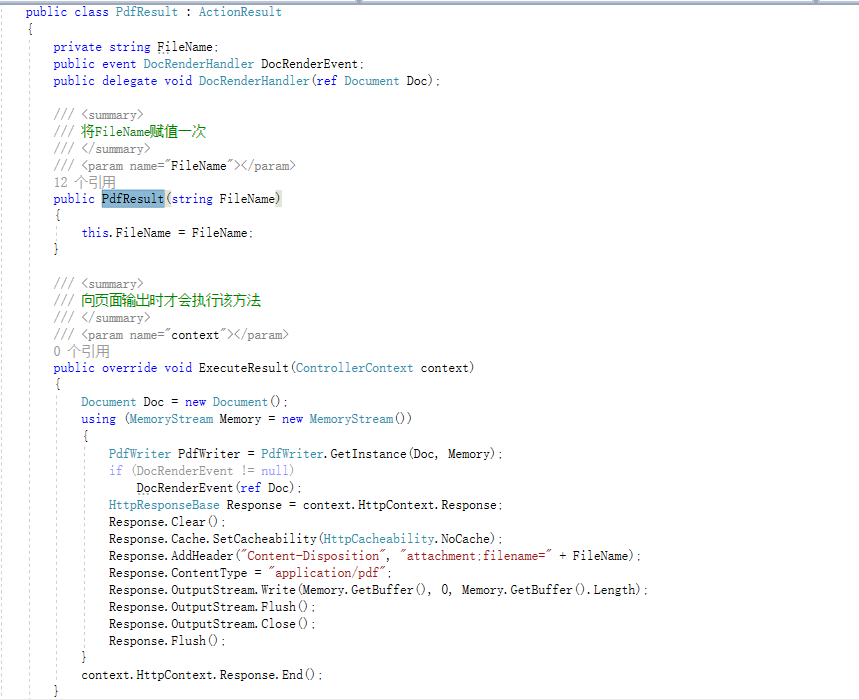
HandlerErrorAttribute 统一处理Error错误日志。（属于txt文本日志）

#### （8）工具类：

##### NPOI、（Word,Excle操作）可以用Apose



##### PdfResult（ITextSharp操作，pdf文档导出）

##### Newtonsoft.Json、（Json格式转换序列化）



##### 验证码



##### 公共类库

