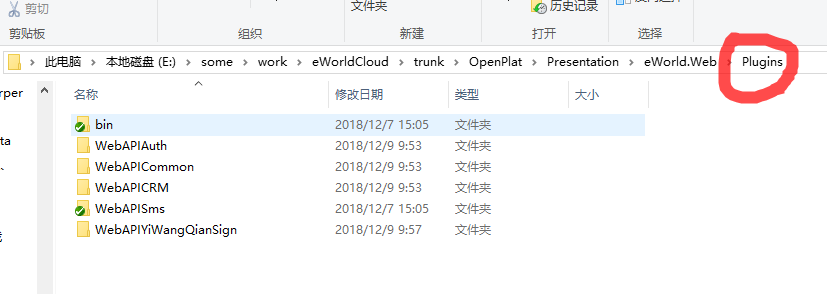
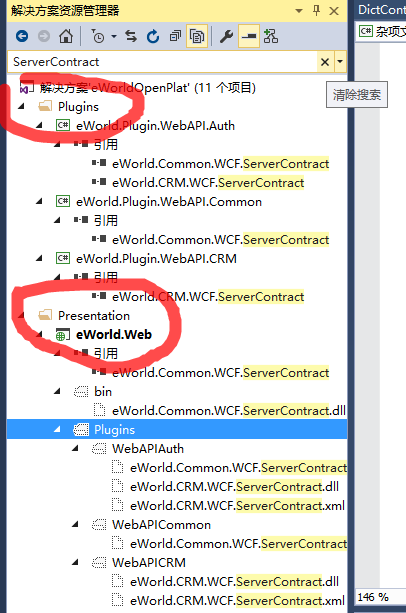
探索一些

1.



彪哥这里的接口，Plugins，似乎最后还都是包在web里面的，虽然在vs中看着是并列一般，只有也解释了为什么这些web api应用只是类库，而不是可以发布的web api应用。

不对，仔细看了下，只是最终编译后，生成在了这唯一的一个web应用当中。



2.

似乎是动态引用的，在变的。刚刚弹出来，让我重新加载。说明crm那边修改了



3.

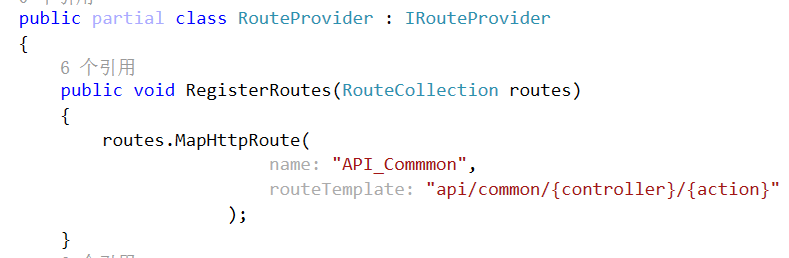
可参照网上找的autofac api注入：

api的注入：

api的实现：

builder.RegisterApiControllers(typeFinder.GetAssemblies().ToArray());

各个接口的地址：



4.

访问获取一级科室的接口

<http://localhost:7000/api/common/Dict/Get_Plat_SubjectFirst>

（1）

namespace eWorld.Web.Framework.Mvc

{

/// <summary>

/// Base nopCommerce model

/// </summary>

public partial class BaseNopModel

{

public BaseNopModel()

{

this.CustomProperties = new Dictionary<string, object>();



}

public virtual void BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)

{

}

/// <summary>

/// 使用这个属性来存储任何自定义值的模型。

/// </summary>

public Dictionary<string, object> CustomProperties { get; set; }

}

（2）

namespace eWorld.Web.Controllers

{

public class AuthorizationController : Controller

{

private readonly IInstallationLocalizationService \_locService;

public AuthorizationController(IInstallationLocalizationService locService)

{

this.\_locService = locService;

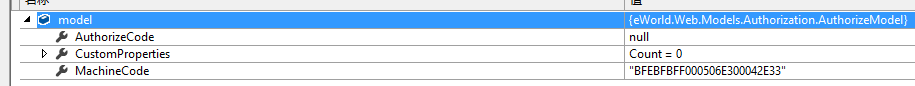
}

public ActionResult Index()

{

var model = new AuthorizeModel();

model.MachineCode = AuthorizationCode.getMNum();



return View(model);

}

这里的getMNum()方法寻找：

/// <summary>

/// 生成机器码

/// </summary>

/// <returns></returns>

public static string getMNum()

{

string strNum = getCpu() + GetDiskVolumeSerialNumber();//获得24位Cpu和硬盘序列号

string strMNum = strNum.Substring(0, 24);//从生成的字符串中取出前24个字符做为机器码

return strMNum;

}

（3）

namespace eWorld.Web.Framework

{

public class NopValidatorFactory : AttributedValidatorFactory

{

//private readonly InstanceCache \_cache = new InstanceCache();

public override IValidator GetValidator(Type type)

{

查找Type：



if (type != null)

{

var attribute = (ValidatorAttribute)Attribute.GetCustomAttribute(type, typeof(ValidatorAttribute));

结果为null



if ((attribute != null) && (attribute.ValidatorType != null))

{

//验证可以依赖于某些客户的特殊设置（如工作语言）

//这就是为什么我们不缓存验证器

//var instance = \_cache.GetOrCreateInstance(attribute.ValidatorType,

// x => EngineContext.Current.ContainerManager.ResolveUnregistered(x));

var instance = EngineContext.Current.ContainerManager.ResolveUnregistered(attribute.ValidatorType);

return instance as IValidator;

}

}

return null;

}

}

}

原始接口：

namespace FluentValidation

{

/// <summary>

/// Gets validators for a particular type.

///

/// </summary>

public interface IValidatorFactory

{

/// <summary>

/// Gets the validator for the specified type.

///

/// </summary>

IValidator<T> GetValidator<T>();

/// <summary>

/// Gets the validator for the specified type.

///

/// </summary>

IValidator GetValidator(Type type);

}

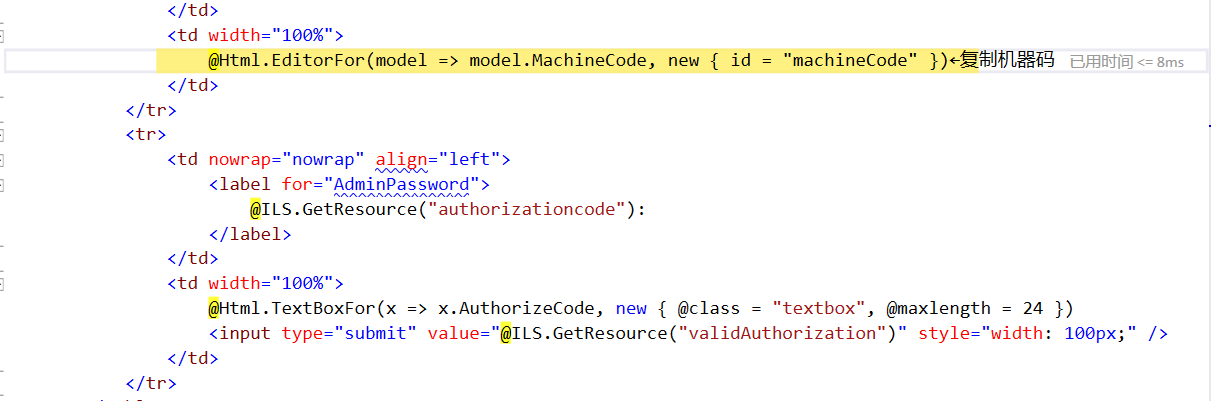
}

（4）



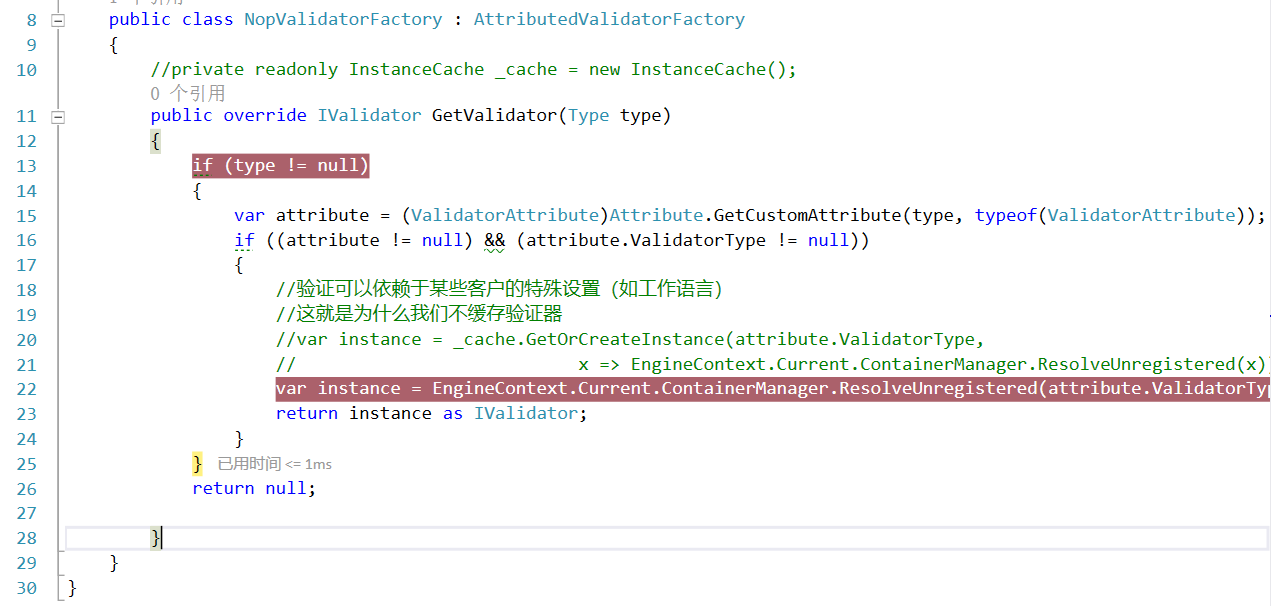
（5）

转跳授权的Index页面：



（6）

又继续跳回来。



（7）

继续跳到这个页面：



（8）

最后跳回授权的首页



5.

6.

7.

8.