**IApplicationModelConvention 接口**

public interface IApplicationModelConvention

{

    void Apply(ApplicationModel application);

}

这个接口就是主要是用来自定义一些 MVC 约定的一些东西的， Apply这个方法传入一个ApplicationModel对象，我们可以利用这个对象来修改我们需要的东西

**ApplicationModel 对象**

public class ApplicationModel : IPropertyModel, IFilterModel, IApiExplorerModel

{

    public ApiExplorerModel ApiExplorer { get; set; }

    public IList<ControllerModel> Controllers { get; }

    public IList<IFilterMetadata> Filters { get; }

    public IDictionary<object, object> Properties { get; }

}

ApiExplorerModel：主要是配置默认MVC Api Explorer的一些东西，包括Api的描述信息，组信息，可见性等。

ControllerModel：主要是 Controller默认约定相关的了，这个里面东西就比较多了，就不一一介绍了，我们等下就要配置里面的一个东西。

IFilterMetadata ：空接口，主要起到标记的作用。

**示例：添加全局路由统一前缀**

没有那么多废话了，直接上代码，要说的话全在代码里：

//定义个类 RouteConvention，来实现 IApplicationModelConvention 接口

public class RouteConvention : IApplicationModelConvention {

    //接口的Apply方法

    public void Apply (ApplicationModel application) {

        //遍历所有的 Controller

        foreach (var controller in application.Controllers) {

            // 遍历所有方法

            foreach (var action in controller.Actions) {

                // 遍历所有选择器

                foreach (var selector in action.Selectors) {

                    // 如果没有路由特性则添加一个路由特性

                    if (selector.AttributeRouteModel == null) {

                        selector.AttributeRouteModel = new AttributeRouteModel (

                            new RouteAttribute (

                                $"api/{controllerModel.ControllerName}/{action.ActionName}"

                            )

                        );

                    }

                }

            }

        }

    }

}

最后，在 Startup.cs 文件中，添加上面的扩展方法就可以了。

public class Startup {

    public Startup (IHostingEnvironment env) {

        //...

    }

    public void ConfigureServices (IServiceCollection services) {

        //...

        services.AddMvc (opt => {

            // 路由参数在此处仍然是有效的，比如添加一个版本号

            opt.Conventions.Insert (0, new RouteConvention ());

        });

    }

}