# 理论知识

编辑记录

第一版 LCTR 2020年7月22日

第二版 LCTR 2020年12月12日

目录

[理论知识 1](#_Toc58839470)

*[基础概念](#_Toc58839471)* [2](#_Toc58839471)

*[跨平台](#_Toc58839472)* [3](#_Toc58839472)

*[环境](#_Toc58839473)* [4](#_Toc58839473)

[](#_Toc58839474) **[开发工具包](#_Toc58839474)** [4](#_Toc58839474)

[](#_Toc58839475) **[运行时](#_Toc58839475)** [4](#_Toc58839475)

[](#_Toc58839476) **[微软官方下载页面](#_Toc58839476)** [4](#_Toc58839476)

*[规范](#_Toc58839477)* [5](#_Toc58839477)

[](#_Toc58839478) **[命名规范](#_Toc58839478)** [5](#_Toc58839478)

*[项目文件结构](#_Toc58839479)* [6](#_Toc58839479)

[](#_Toc58839480) **[配置文件](#_Toc58839480)** [6](#_Toc58839480)

[](#_Toc58839481) **[项目结构](#_Toc58839481)** [6](#_Toc58839481)

[](#_Toc58839482) **[项目文件](#_Toc58839482)** [6](#_Toc58839482)

*[项目开发说明](#_Toc58839483)* [8](#_Toc58839483)

[](#_Toc58839484) **[常用NuGet包](#_Toc58839484)** [8](#_Toc58839484)

[](#_Toc58839485) **[常用NuGet包的使用方式](#_Toc58839485)** [9](#_Toc58839485)

## ***基础概念***

* **服务定位模式**（**Service Locator Pattern**）
* **依赖注入模式**（**Dependency Injection Pattern**）
* **工厂模式**（**Factory Pattern**）
* **面向切面编程**（**AOP**）
* **参考资料**
  1. [控制容器的反转和依赖注入模式](https://martinfowler.com/articles/injection.html)
  2. [设计模式之工厂模式（factory pattern）](https://www.cnblogs.com/yssjun/p/11102162.html)
  3. [.Net中的AOP系列之《间接调用——拦截方法》](https://www.cnblogs.com/farb/p/MethodInterception.html)

## ***跨平台***

* **Windows**
* **Linux**
* **macOS**

## ***环境***

* 开发工具包
  + SDK（包括运行时）
* 运行时
  + ASP.NET Core Runtime
    - 运行网站和服务器应用
  + [Hosting Bundle](https://dotnet.microsoft.com/download/dotnet-core/thank-you/runtime-aspnetcore-3.1.6-windows-hosting-bundle-installer) (windows)
    - 包括. .NET Core Runtime
    - 以及IIS支持
  + .NET Core Runtime
    - 只能运行控制台应用
  + Desktop Runtime (windows)
    - 包含.NET Core Runtime
* **[微软官方下载页面](https://dotnet.microsoft.com/download/dotnet-core)**

## ***规范***

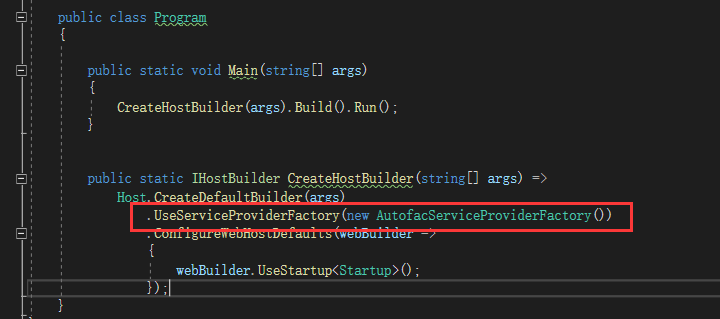
* **命名规范**
  + **大驼峰**
    - **类名**
    - **属性**
    - **字段**
    - **方法**
  + **小驼峰**
    - **变量**
    - **参数**

## ***项目文件结构***

* **配置文件**
  + **VS相关**
    - 应用启动配置（通过VS启动） \Properties\launchSettings.json
    - 发布配置 \Properties\PublishProfiles\\*.pubxml
    - Docker部署命令 \Dockerfile
  + **应用配置** \appsettings.json
* **项目结构**
  + **控制台应用**
  + **业务逻辑类库**
  + **实体模型类库**
  + **业务模型类库**
* **项目文件**
  + **常用**
    - 入口 \Program.cs
    - 启动设置 \Startup.cs
    - 组件配置类 \Configura\\*Configure.cs
    - 控制器 \Controllers\\*Controller.cs
    - 拦截器 \Filter\\*Filter.cs
    - 中间件 \Middleware\\*Middleware.cs
  + **附加**
    - 项目结构数据 \\*.xml
    - 静态文件资源 \wwwroot\\*.\*
    - 使用IIS时上传此文件会使应用停止运行 \app\_offline.htm
    - SOAP
      * Client
        + 接口类 \Services \Client\I\*Service.cs
      * Server
        + 接口类 \Services\Server\I\*Service.cs
        + 实现类 \Services\Server\\*Service.cs
    - 类库
      * 业务逻辑
        + 接口类 \IBusiness\I\*Business.cs
        + 实现类 \Business\\*Business.cs
        + 工具 \Util\\*.cs
        + 日志组件 \Logger\Logger.cs
      * 实体模型
        + 假如数据表为Base\_Area \Base\Base\_Area.cs
      * 业务模型
        + 假如数据表为Base\_Area \Base\AreaDTO.cs
        + 和某表相关的类也放在相关文件夹中 \Base\AreaType.cs

## ***项目开发说明***

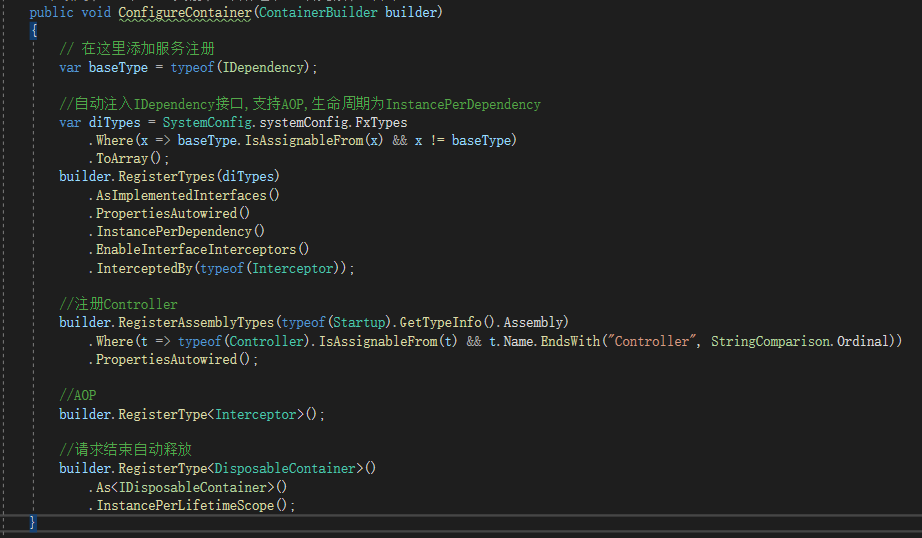
* **常用NuGet包**
  + **[Library.Container](#Library_Container" \o "查看使用方式)**
    - Autofac
      * [官方文档](https://autofaccn.readthedocs.io/zh/latest/)
    - IOC容器
    - AOP拦截类
    - 过滤器
  + **[Swashbuckle.AspNetCore](#Swashbuckle_AspNetCore" \o "查看使用方式)**
    - 接口文档
  + **[Library.OpenApi](#Library_OpenApi" \o "查看使用方式)**
    - 扩展接口框架的功能
    - 一般也会和swagger结合使用
  + **[Library.DataMapping](#Library_DataMapping" \o "查看使用方式)**
    - 提供数据映射扩展方法
  + **[Library.FreeSql](#Library_FreeSql" \o "查看使用方式)**
    - 通过依赖注入的方式使用国产的第三方ORM
      * 支持多个常见的数据库
      * 支持CodeFirest模式，可以自动同步实体到数据库
      * [官方文档](https://www.cnblogs.com/FreeSql/p/11531300.html)
  + **[Library.Elasticsearch](#Library_Elasticsearch" \o "查看使用方式)**
    - 通过依赖注入的方式使用Elasticsearch
      * 使用java开发的基于Lucene的搜索服务器
      * [快速了解](https://mp.weixin.qq.com/s/dbBtGkFAENv1lbgzqfd4cA)
      * [官网文档](https://www.elastic.co/guide/en/elasticsearch/reference/7.x/getting-started.html)
  + **Library. Extension**
    - 包含一些常用的拓展功能
  + **Library.Cache**
    - 缓存拓展功能
  + **Library.Http**
    - 模拟http请求
  + **Library.Log**
    - 日志组件
  + **Library.IdTool**
    - 雪花ID生成器
  + **Library.Soap**
    - 在.Net Core上实现Soap协议
      * 服务端
      * 客户端
* **常用NuGet包的使用方式**
  + **Library.Container**
    - 在**Program.cs**中使用Autofac替换.Net Core默认的服务实现工厂



* + - 在**Startup.cs**的**Configure**方法中获取Autofac服务容器



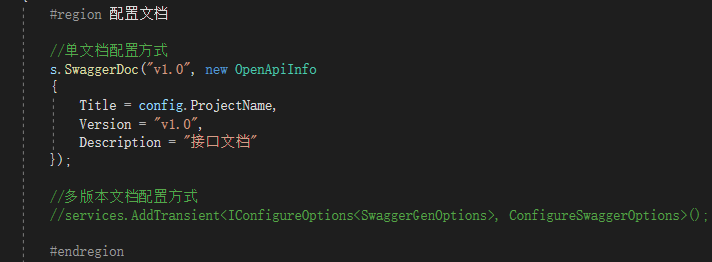
* + - 在**Startup.cs**的**ConfigureContainer方法中注册控制器和APO拦截类**



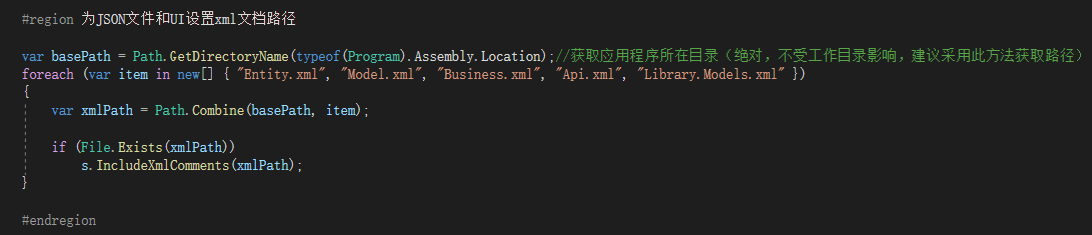
* + - 获取服务（例如IOperator）



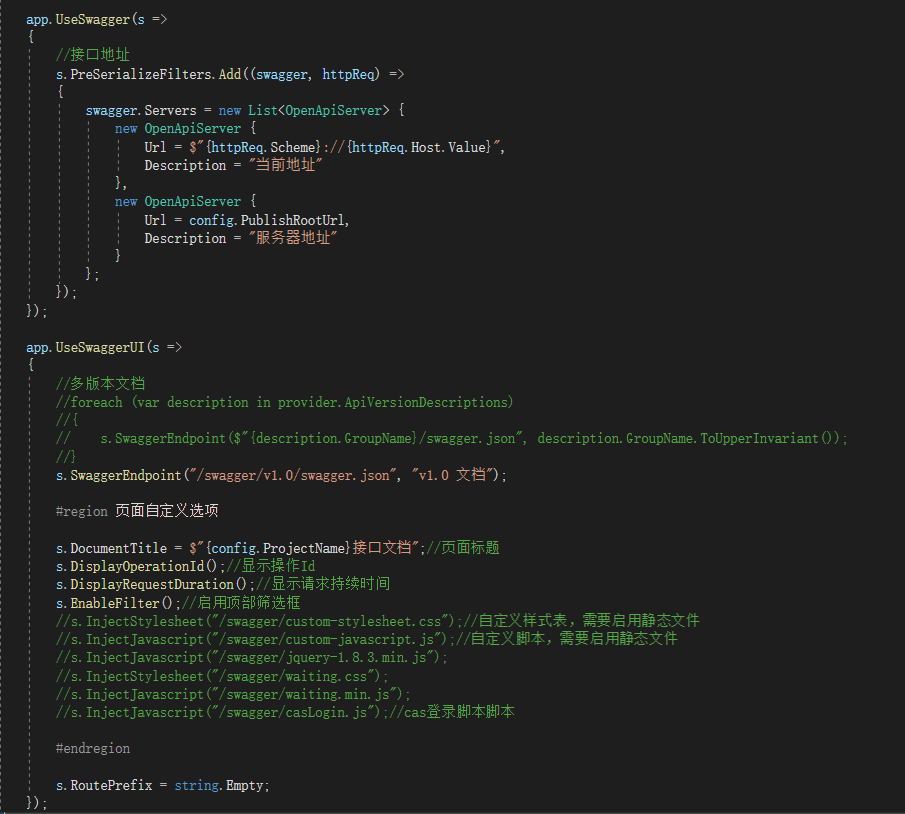
* + **Swashbuckle.AspNetCore**
    - **基础配置**
      * 在**Startup.cs**的**ConfigureServices方法中**
        + **配置文档**



* + - * + **配置xml文档**



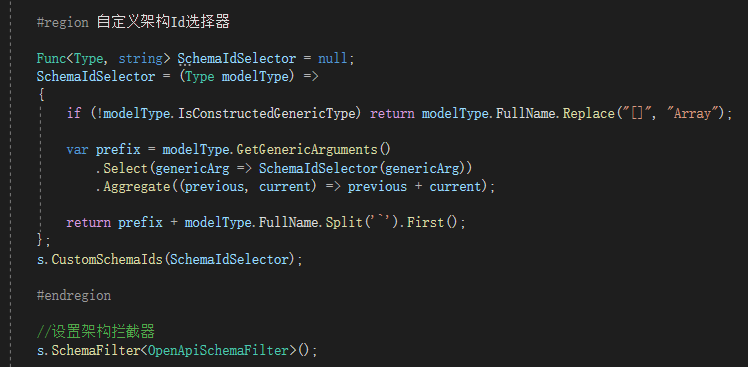
* + - * **在Startup.cs**的**Configure方法中配置UI**



* + - **附加配置**
      * 在**Startup.cs**的**ConfigureServices**方法中
        + **配置多版本**



* + - * + **配置框架自定义架构（需要Library.OpenApi）**



* + - * + **启用注解**



* + **Library.OpenApi**
    - **和swagger搭配使用**
      * 为**实体类**的属性添加**接口架构**特性(注解)



* + - * 在**控制器**方法中添加特性



* + - **控制Json序列化**
      * 在**业务模型**中添加**主标签**特性



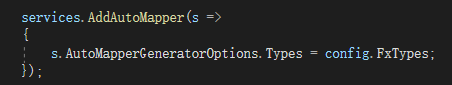
* + - * 在**继承的实体类**中添加**附属标签**特性



* + - * 使用专属的扩展方法得到**筛选**后的数据



* + **Library.DataMapping**
    - **基础配置**
      * 在**Startup.cs**的**ConfigureServices方法中**



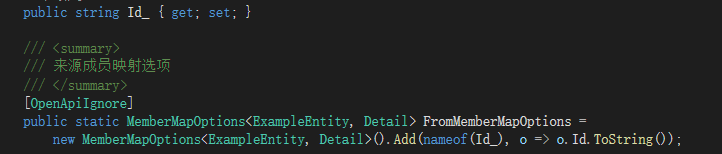
* + - * + **FxTypes为要注册的类型(一般是业务模型)的集合**
      * 在**实体类**中添加特性
        + 映射来源



ExampleEntity为指定的**实体**

在需要开启自定义成员映射时，第二个参数设置为true

当需要设置自定义成员映射选项时



ExampleEntity为指定的**实体类**，Detail为指定的**业务模型**

Add方法可以链式调用，**第一个参数不可重复**

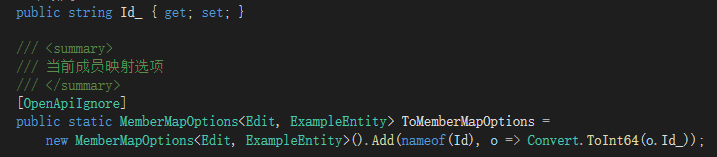
* + - * + 映射目标



ExampleEntity为指定的**实体**

在需要开启自定义成员映射时，第二个参数设置为true

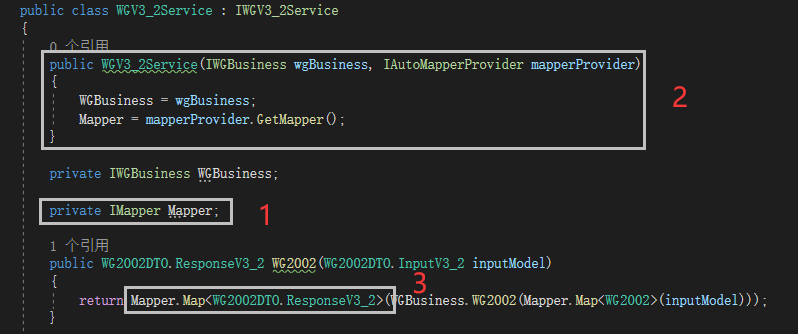
当需要设置自定义成员映射选项时



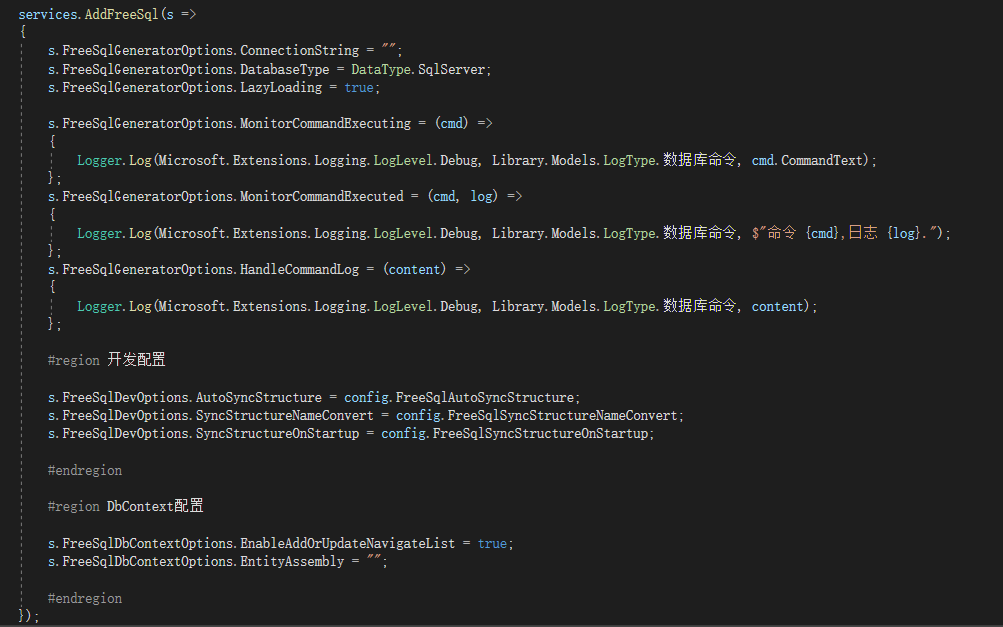
ExampleEntity为指定的**实体类**，Edit为指定的**业务模型**

Add方法可以链式调用，**第一个参数不可重复**

* + - 调用方式



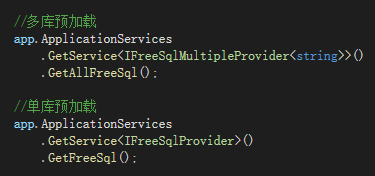
* + - * 1. 定义一个类型为**IMapper**的字段；
        2. 在**构造函数**中注入**IAutoMapperProvider**；
        3. 调用**Map**方法；
  + **Library.FreeSql**
    - 基础配置
      * 在**Startup.cs**的**ConfigureServices方法中**
        1. **单库**



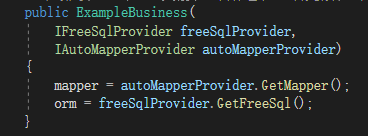
* + - * 1. **多库**



* + - * + 具体配置信息看注释即可
      * **如果要预热（自动同步实体）一下，在Startup.cs**的**Configure方法中**



* + - 调用方式



* + - * 1. 在**构造函数**中注入

单库**IFreeSqlProvider**；

多库**IFreeSqlMultipleProvider<TKey>**;(TKey为库标识的数据类型)

* + - * 1. 调用方法获取需要的对象；
    - 扩展方法
      * 返回动态类型的数据 **ISelect.ToDynamic**
      * 获取数据并检查是否为空 **IBaseRepository.GetAndCheckNull**
      * 运行事务  **IFreeSql.RunTransaction**
    - 关于**CodeFirst**
      * 设置**FreeSqlDevOptions.AutoSyncStructure**属性为true
      * 如需要在应用启动时同步所有实体，需要将**FreeSqlDevOptions.SyncStructureOnStartup设置为true**
      * **实体类**添加特**Table特性**



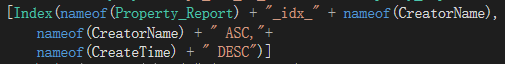
* + - * **实体类**添加自定义**主键名称**（可选）



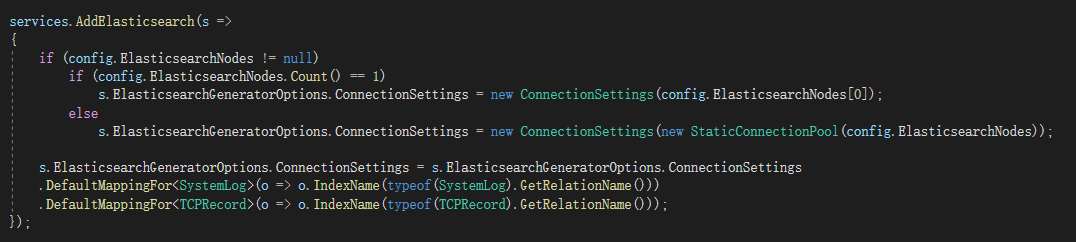
* + - * 实体类**主键**添加特性（其他属性可根据需要设置，比如指定数据类型）



* + - * 实体类**索引**添加特性



* + **Library.Elasticsearch**
    - 基础配置
      * 在**Startup.cs**的**ConfigureServices方法中**



* + - * + 需要使用**DefaultMappingFor**方法配置**默认映射**
      * 在**类**中添加**特性**



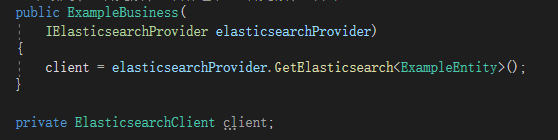
* + - * + **ElasticsearchIndiceExtension**特性提供了设置版本以及分库等一些拓展方法;
        + **ElasticsearchType**特性是必须的，设置**关系名称**和指定**主键**;
      * **在类的属性中添加特性**
* 指定**数据类型**



* **忽略**属性



* + - 使用方式



1. 在**构造函数**中注入**IElasticsearchProvider**；
2. 调用方法通过指定**实体类型**获取**Elasticsearch客户端**；