WTM (WalkingTec MVVM) Framework

WTM framework greatly improves development efficiency by structures conventional coding and automates repetitive coding.

In non-separation mode (eg. LayUI), it connects front-end UI with back-end code. You don't need to separate the front and back platform. You don't need two people to cooperate. WTM helps you to reduce the cost and shorten the R&D period.

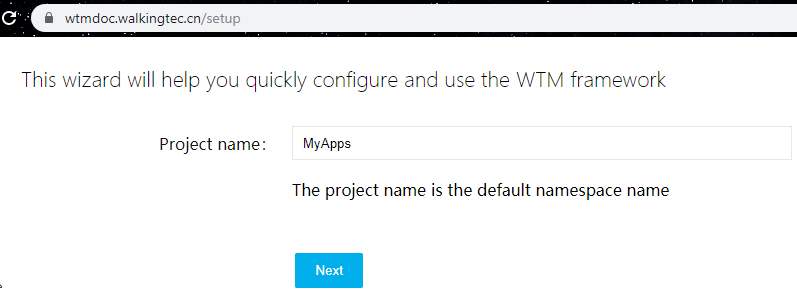
In the mode of front-end and back-end separation (eg. React+Ant Design or Vue+Element), both front-end and back-end code can also be generated at the same time by the code generators. It greatly reduces the communication cost of front-end and back-end personnel. In this way, ‘separation’ is no longer complex and expensive.

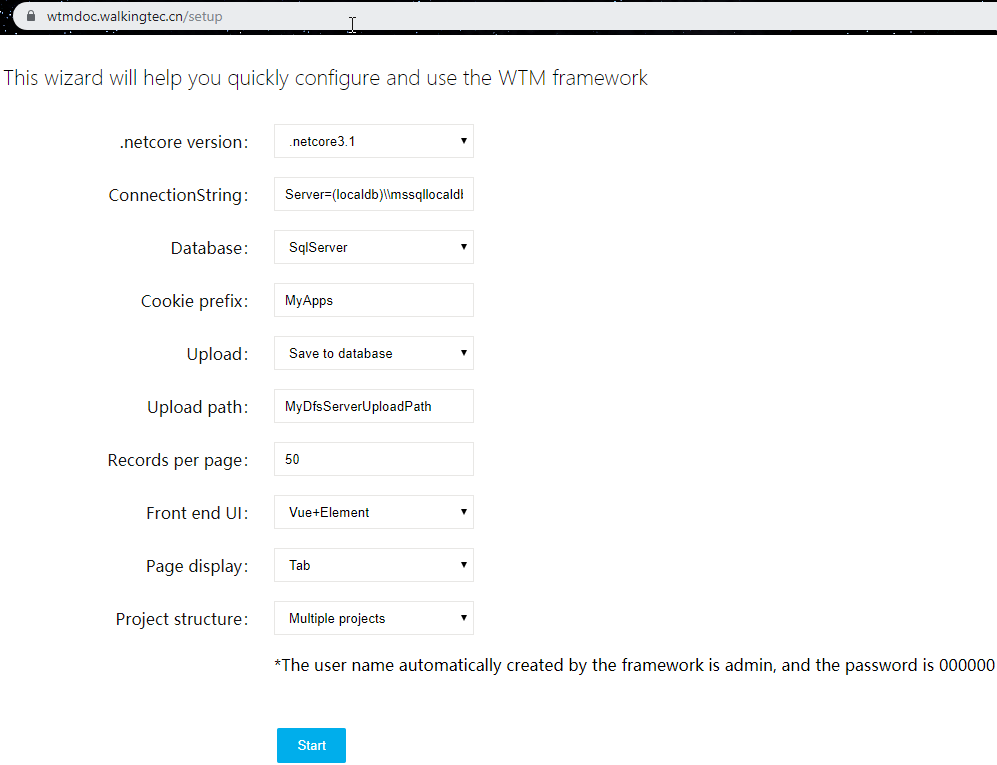
Documentation:

<https://wtmdoc.walkingtec.cn/>

Get Started:

<https://wtmdoc.walkingtec.cn/setup>



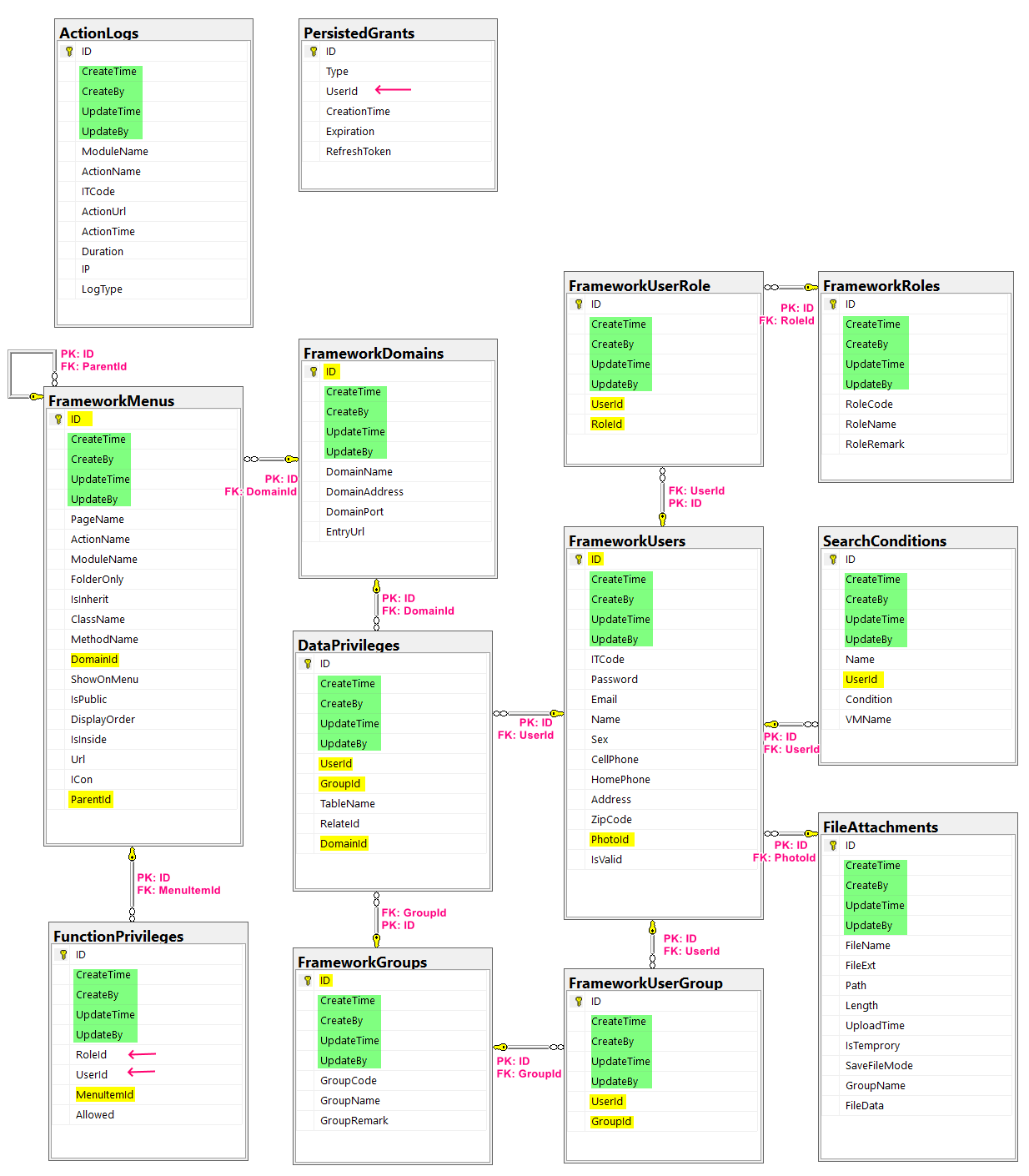


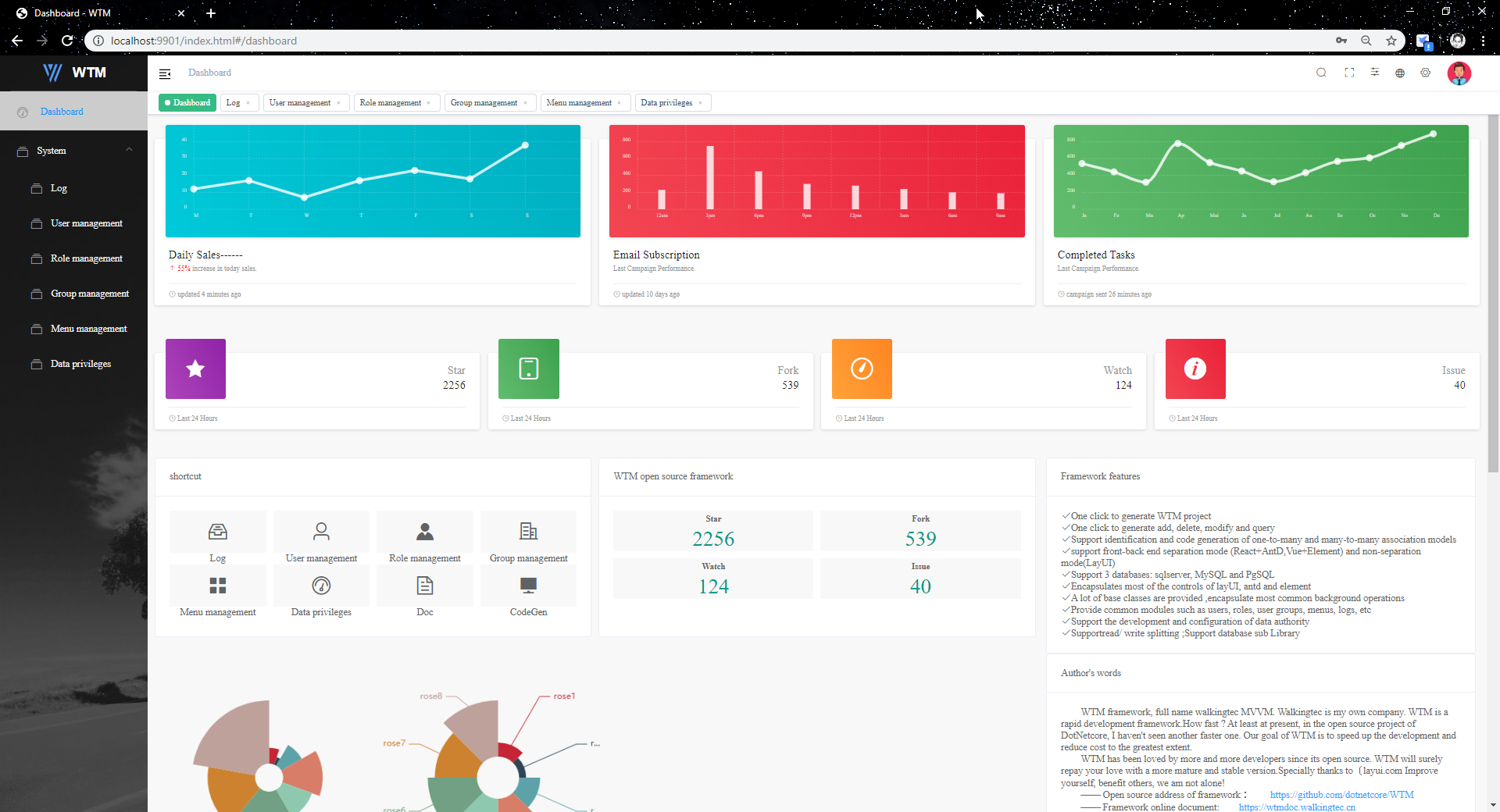
Download 4.7mb zip file

ConnectionStrings: Server=(localdb)\\mssqllocaldb;Database=MyApps\_db;Trusted\_Connection=True;

C:\Users\<username>\MyApps\_db.mdf

C:\Users\<username>\MyApps\_db\_log.ldf





**Add new model**

In MyApps.Model project: add new

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

using WalkingTec.Mvvm.Core;

namespace MyApps.Model

{

    public enum SchoolTypeEnum

    {

        [Display(Name = "Public")]

        PUB,

        [Display(Name = "Private")]

        PRI

    }

    public class School : BasePoco

    {

        [Display(Name = "School Code")]

        [Required(ErrorMessage = "{0} is required")]

        [RegularExpression("^[0-9]{3,3}$", ErrorMessage = "{0} must be three digits")]

        [StringLength(3)]

        public string SchoolCode { get; set; }

        [Display(Name = "School Name")]

        [StringLength(50, ErrorMessage = "{0} has maximun {1} characters")]

        [Required(ErrorMessage = "{0} is required")]

        public string SchoolName { get; set; }

        [Display(Name = "School Type")]

        [Required(ErrorMessage = "{0} is required")]

        public SchoolTypeEnum? SchoolType { get; set; }

        [Display(Name = "Remark")]

        public string Remark { get; set; }

    }

}

In MyApps.DataAccess Project: modify

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

using System;

using System.Collections.Generic;

using System.Linq;

using WalkingTec.Mvvm.Core;

using MyApps.Model;

namespace MyApps.DataAccess

{

    public class DataContext : FrameworkContext

    {

        public DataContext(CS cs)

             : base(cs)

        {

        }

        public DataContext(string cs, DBTypeEnum dbtype)

            : base(cs, dbtype)

        {

        }

        public DataContext(string cs, DBTypeEnum dbtype, string version=null)

             : base(cs, dbtype, version)

        {

        }

        public DbSet<School> Schools { get; set; }

    }

    /// <summary>

    /// DesignTimeFactory for EF Migration, use your full connection string,

    /// EF will find this class and use the connection defined here to run Add-Migration and Update-Database

    /// </summary>

    public class DataContextFactory : IDesignTimeDbContextFactory<DataContext>

    {

        public DataContext CreateDbContext(string[] args)

        {

            return new DataContext("Server=(localdb)\\mssqllocaldb;Database=MyApps\_db;Trusted\_Connection=True;", DBTypeEnum.SqlServer);

        }

    }

}

Migration:

When we create a new model and set database tables in the DataContext ,the framework does not automatically update the database.

Way 1: Delete database and redo

Delete the existing database, run the project again, and the framework will recreate the database according to the new model. it is obviously not suitable when there are some data in the later stage of the project.

Way 2: Use command line tools provided by EF

EF core provides some command-line tools for data synchronization, such as add migration and update.

EF The principle of EF core is to find the DbContext in the code through reflection and find the connection string to determine the database to operate.

However, the WTM framework supports SqlServer, MySql, PgSql and other databases. The configuration file determines which database to use. EF does not know where to find it.

In this case, we can use Design-time DbContext to manually specify the database to be synchronized. For details, please refer to Microsoft documents

<https://docs.microsoft.com/zh-cn/ef/core/miscellaneous/cli/dbcontext-creation>

**1. Install EF Core Design Package (VS)**  
Right click the MyApps.DataAccess project, select "Manage NuGet Packages", install

* Microsoft.EntityFrameworkCore.Design
* Microsoft.EntityFrameworkCore.Tools

**2. Add EF migration (terminal)**  
cd MyApps.DataAccess  
dotnet ef migrations add AddSchoolEntity

**3. Hide all existing tables (VS)**  
Comment out all existing tables and their indexes in <timestamp>\_AddSchoolEntity.cs under Migrations folder  
Shortcut: Ctrl+K+U (VS) Ctrl+/ (VS Code)

**4. Update database (terminal)**  
dotnet ef database update

**5. Check SQL Server (SSMS)**  
dbo.School table is created

Indeed, I think we better to manually add the Initial Migration **prior the first run**. so that I don't need to manually comment out existing tables in the AddSchoolEntity migrations.

dotnet ef migrations add InitialMigration

dotnet ef database update

dotnet ef migrations add AddSchoolEntity

dotnet ef database update

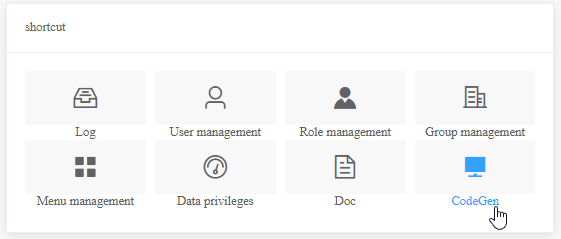
dotnet tool install --global dotnet-ef

dotnet tool update --global dotnet-ef

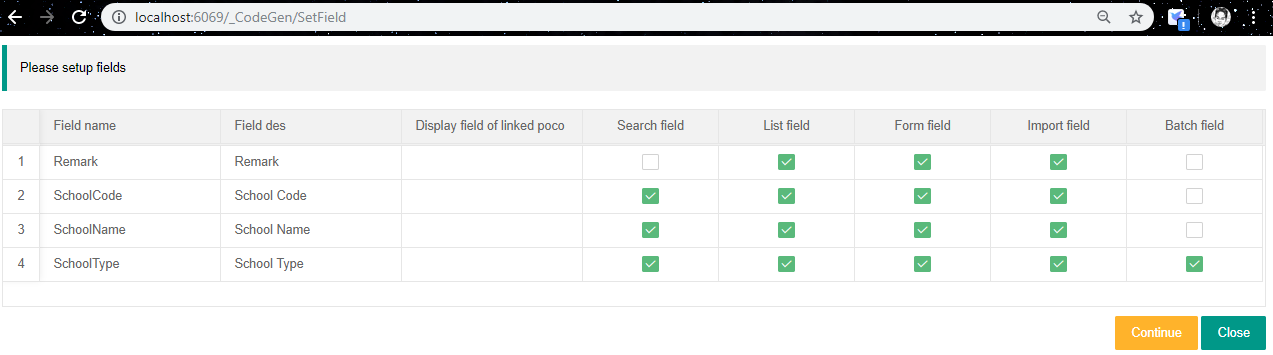
dotnet ef migrations remove -v

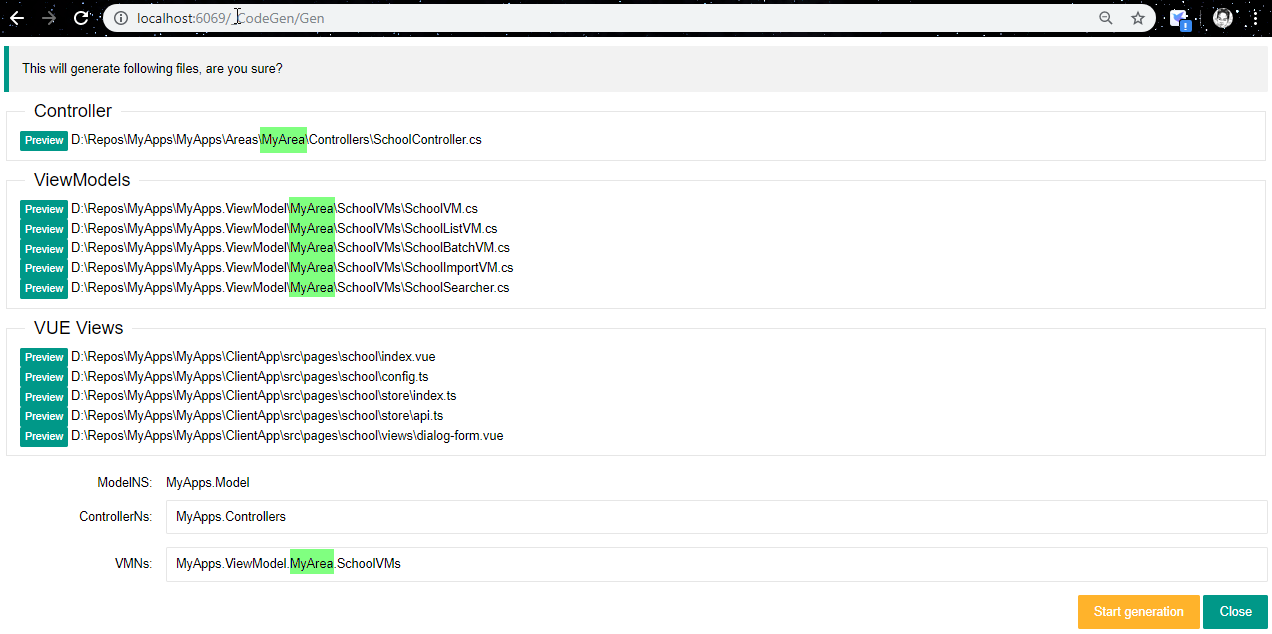
dotnet ef database drop

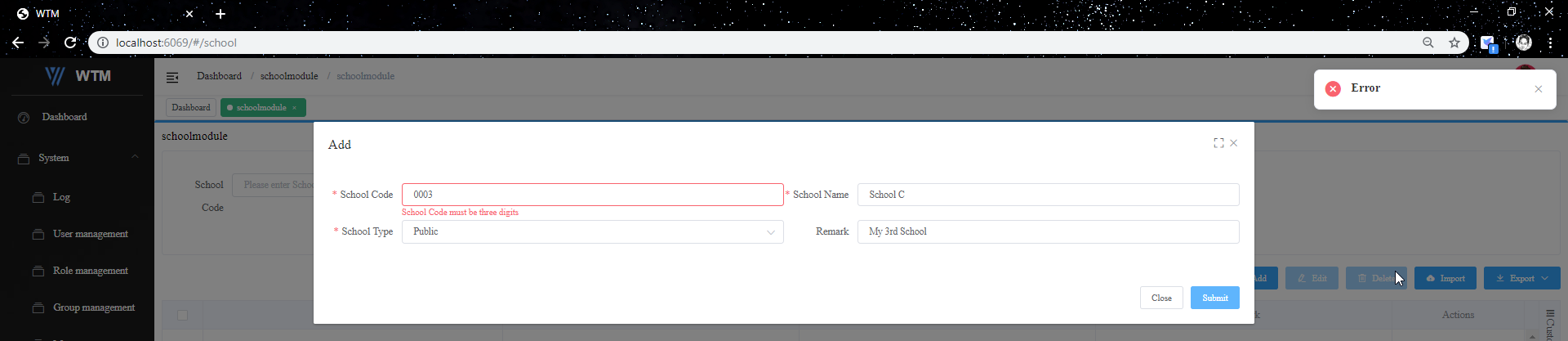
<https://www.entityframeworktutorial.net/efcore/entity-framework-core-migration.aspx>

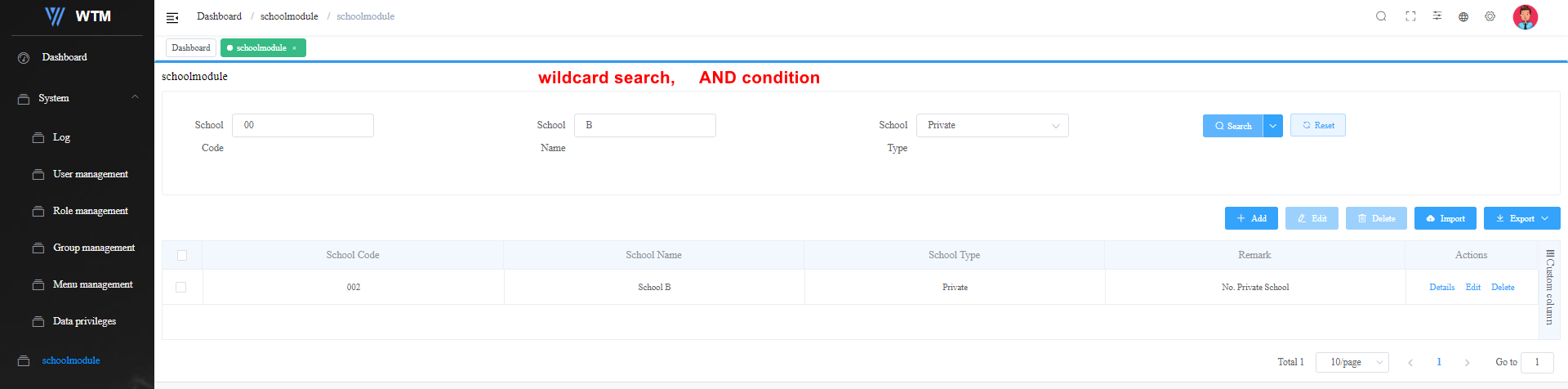


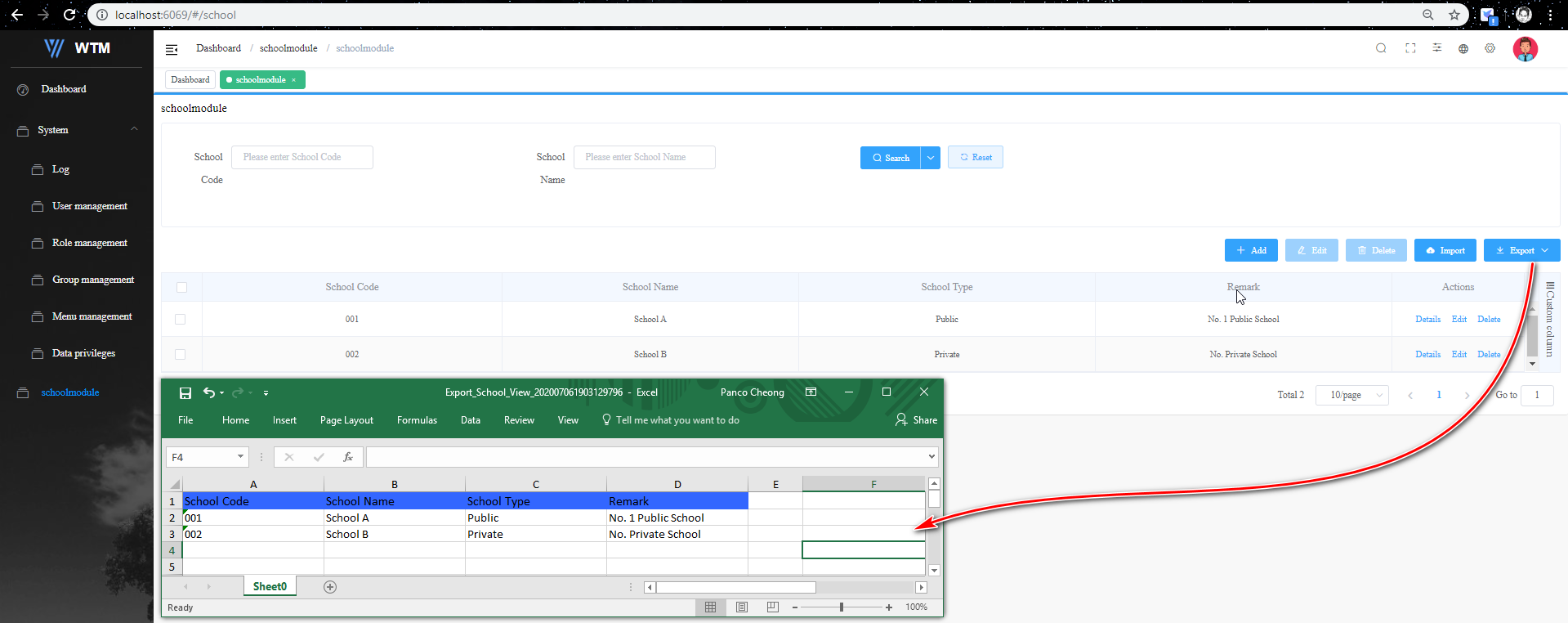


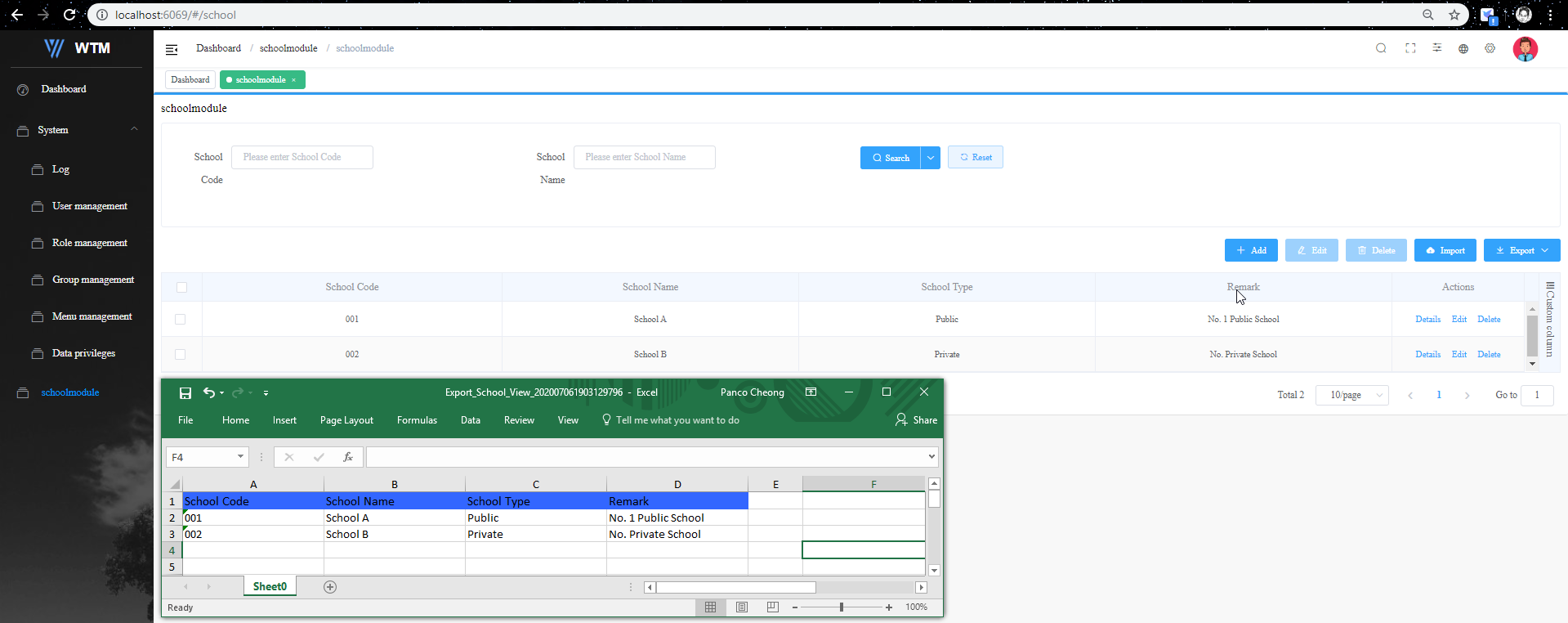


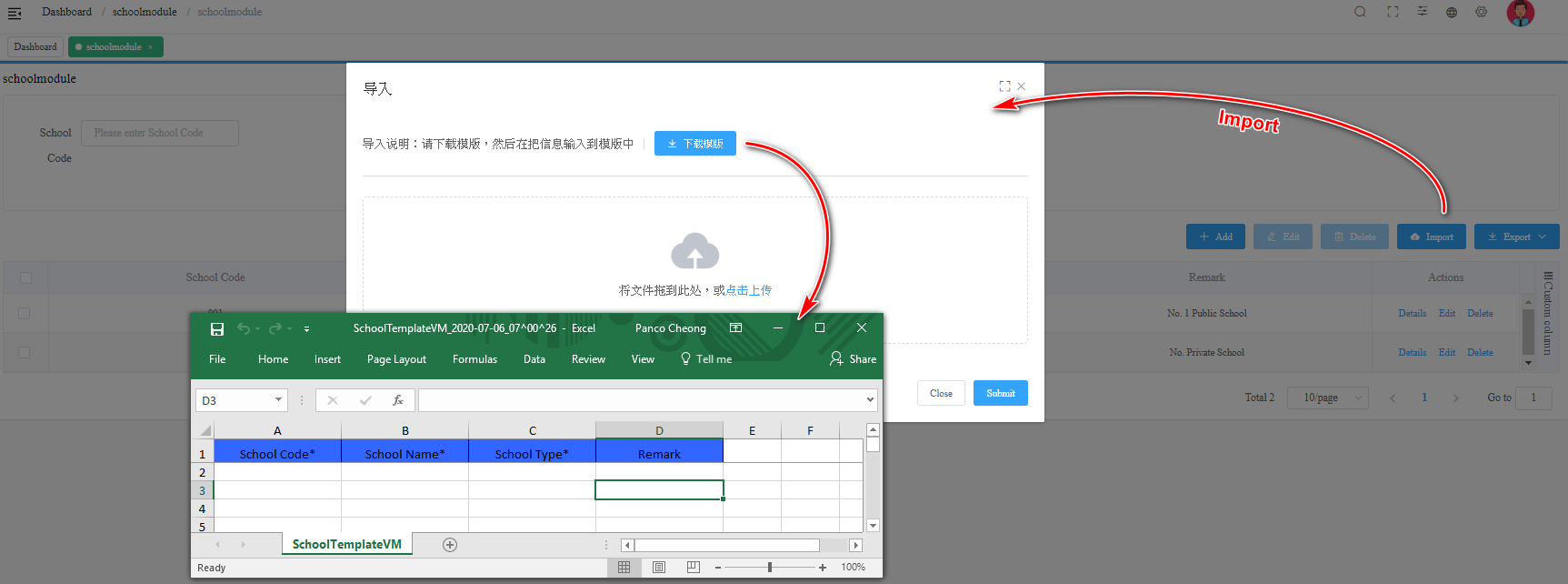










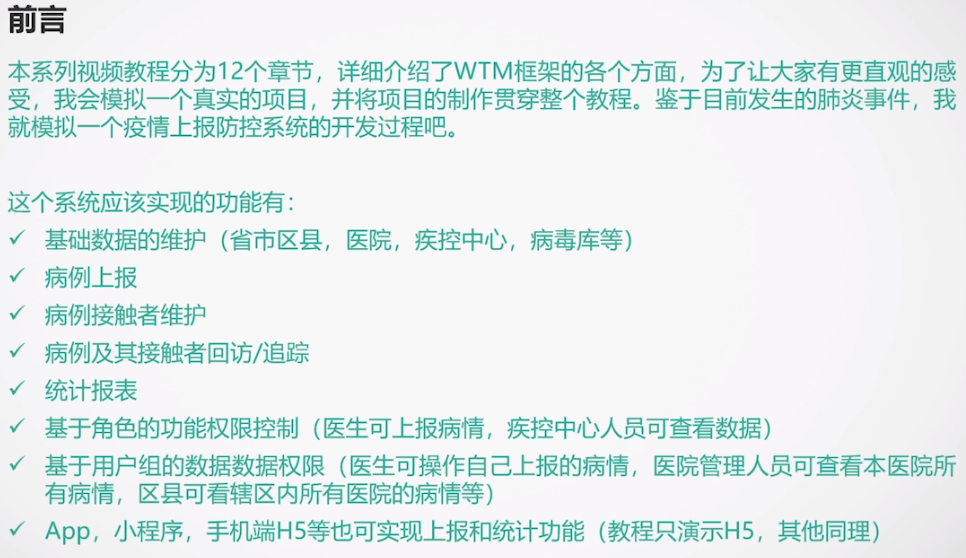


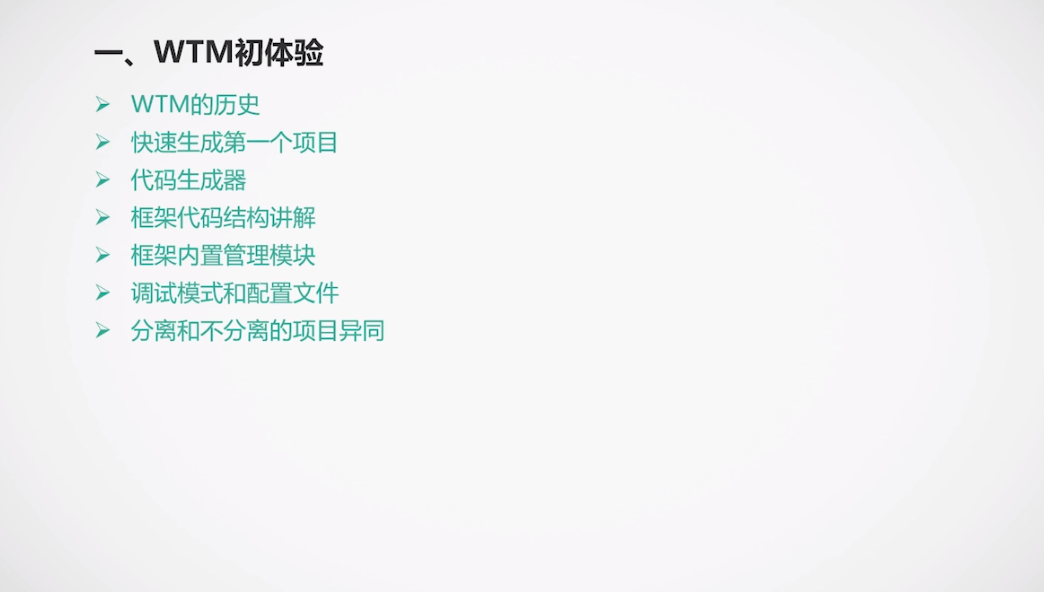
.net core 快速开发框架 WTM系列视频教程

<https://www.bilibili.com/video/av86527514/>



<https://wtmdoc.walkingtec.cn/>





Chapter 1 – Introduction

<https://www.bilibili.com/video/av86527514/?p=2>

WTM = WalkingTec MVVM (Model View View Model)

ERCS – Virus database sample app

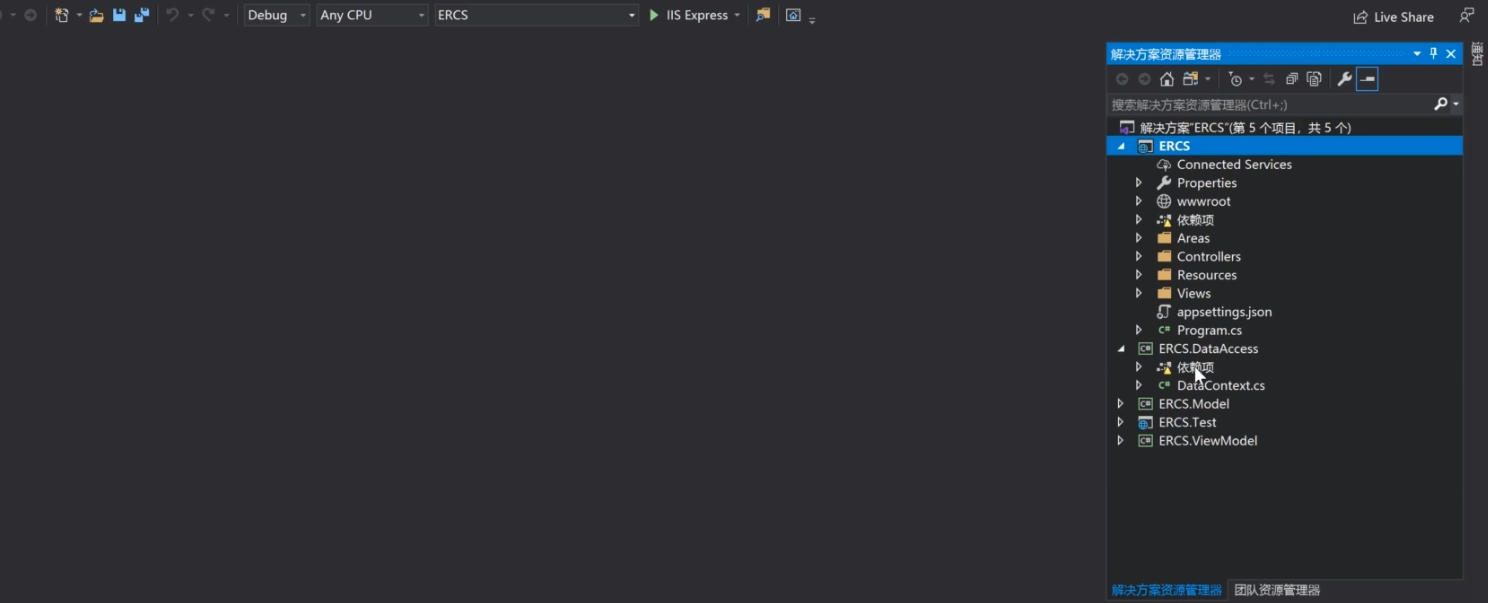


Not good on Oracle DB

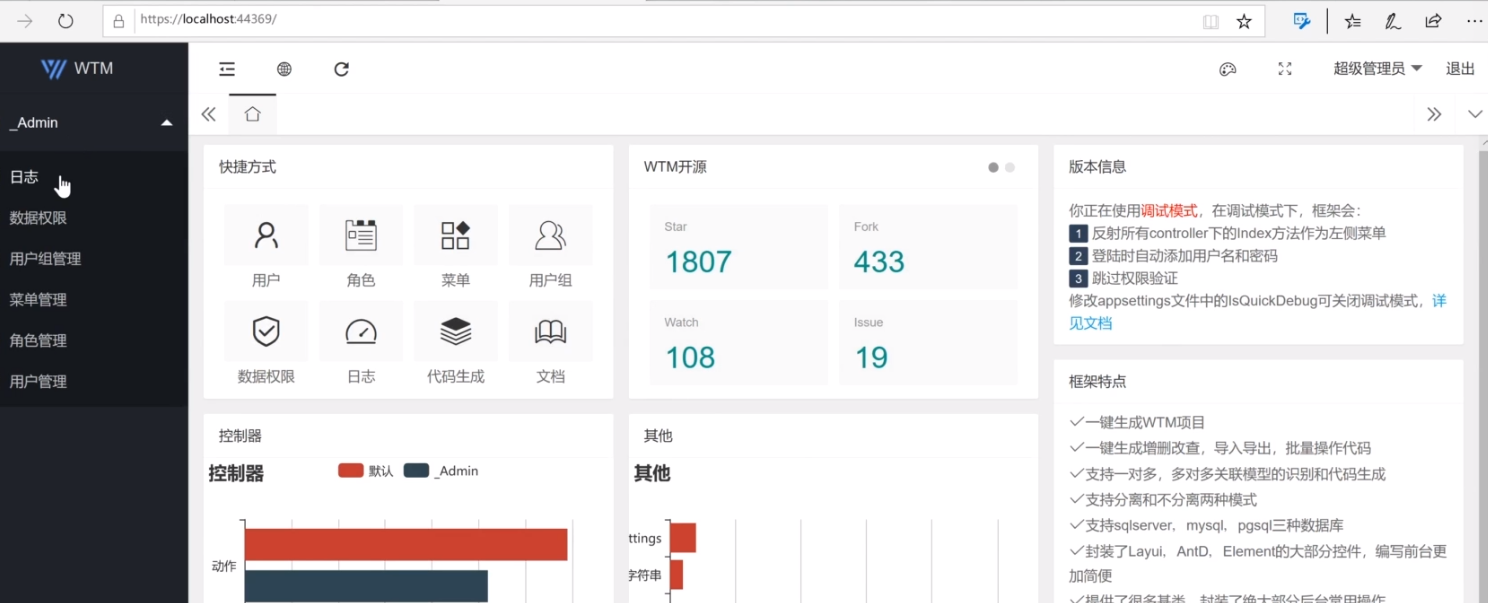
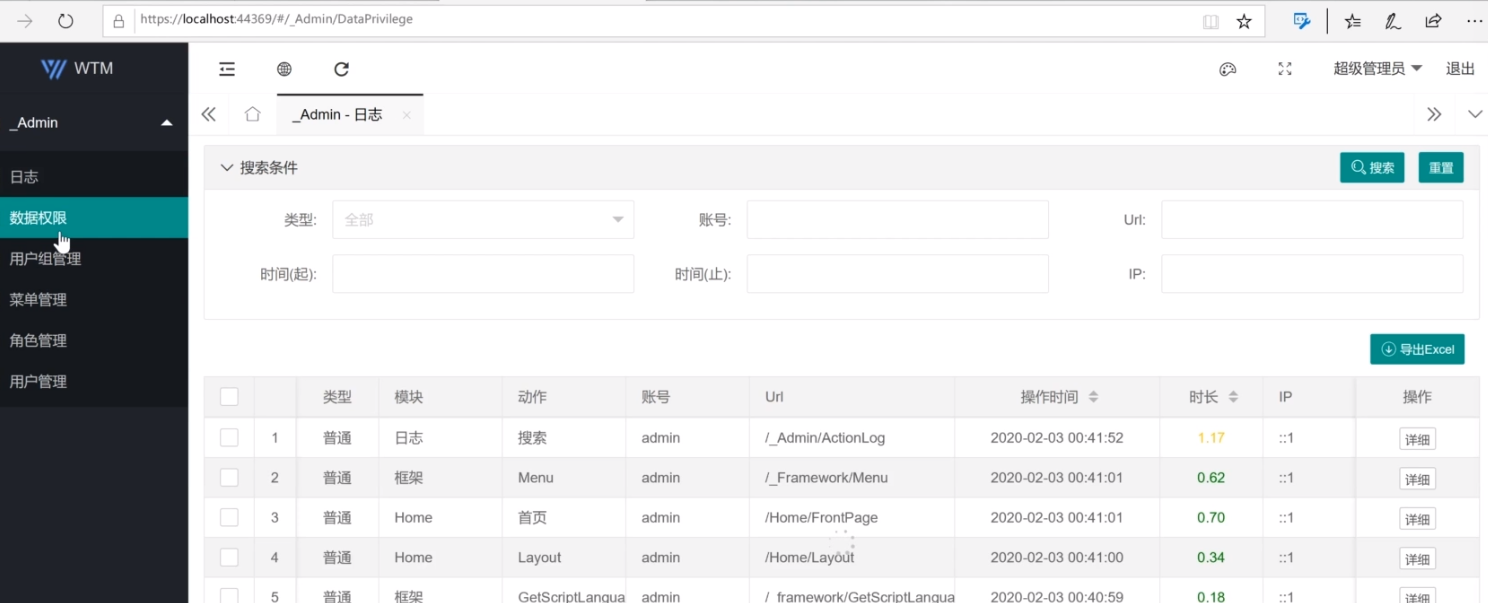
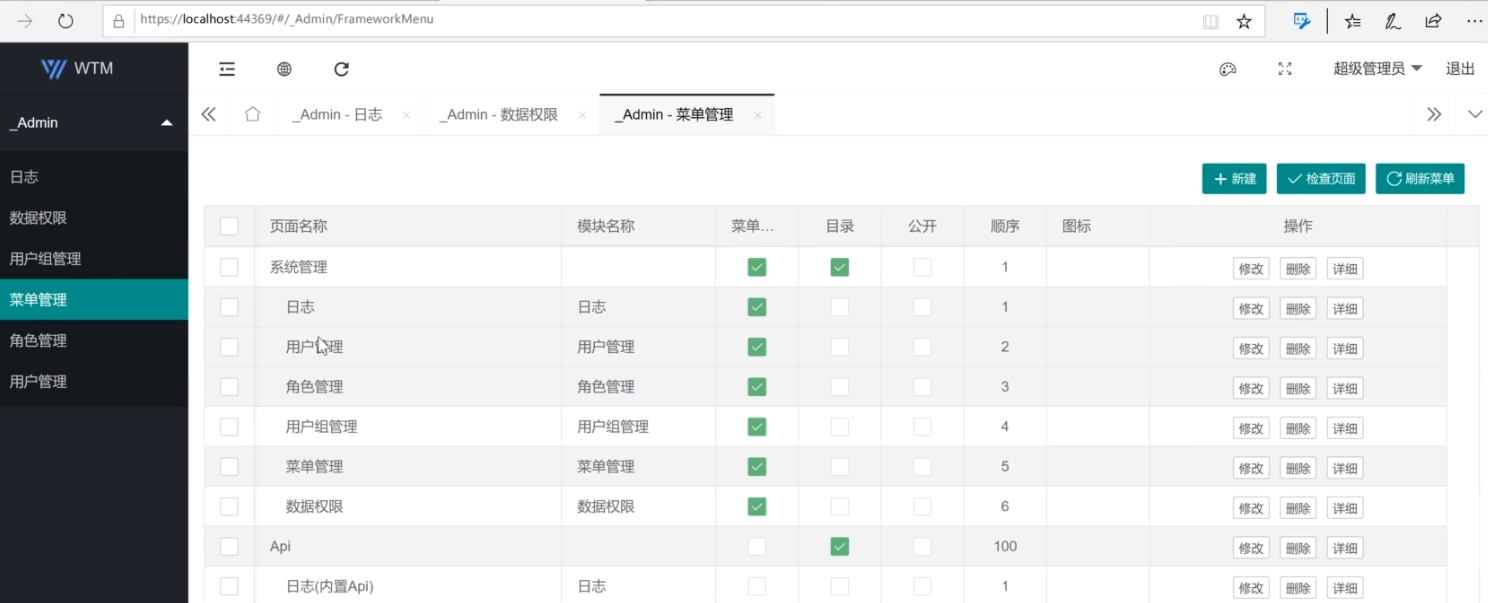
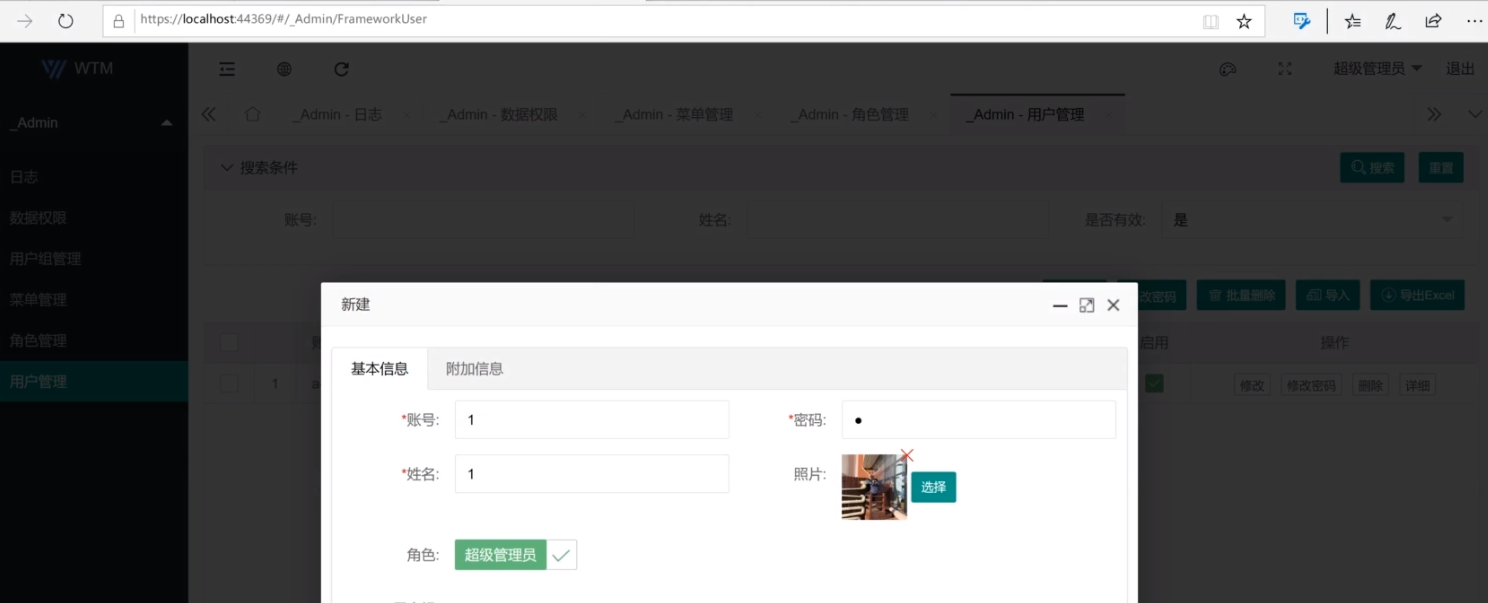
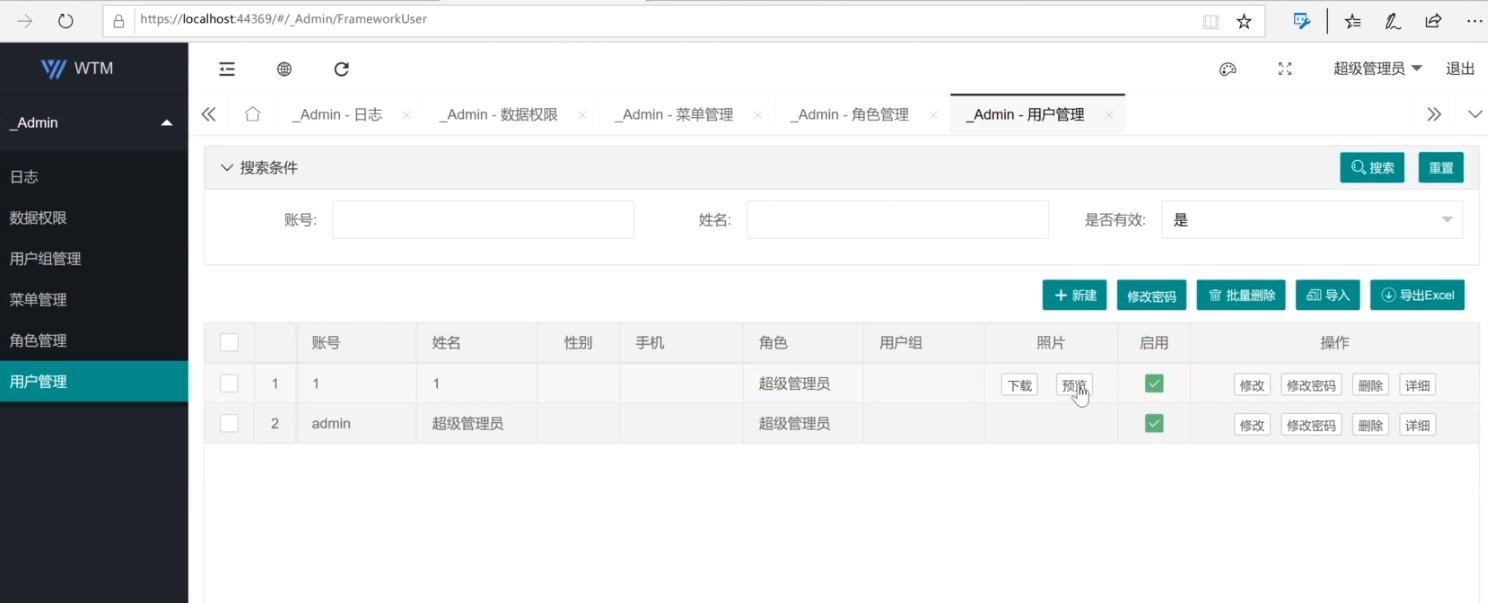
SQL Server 2012 and above

MySQL 5.7 and above





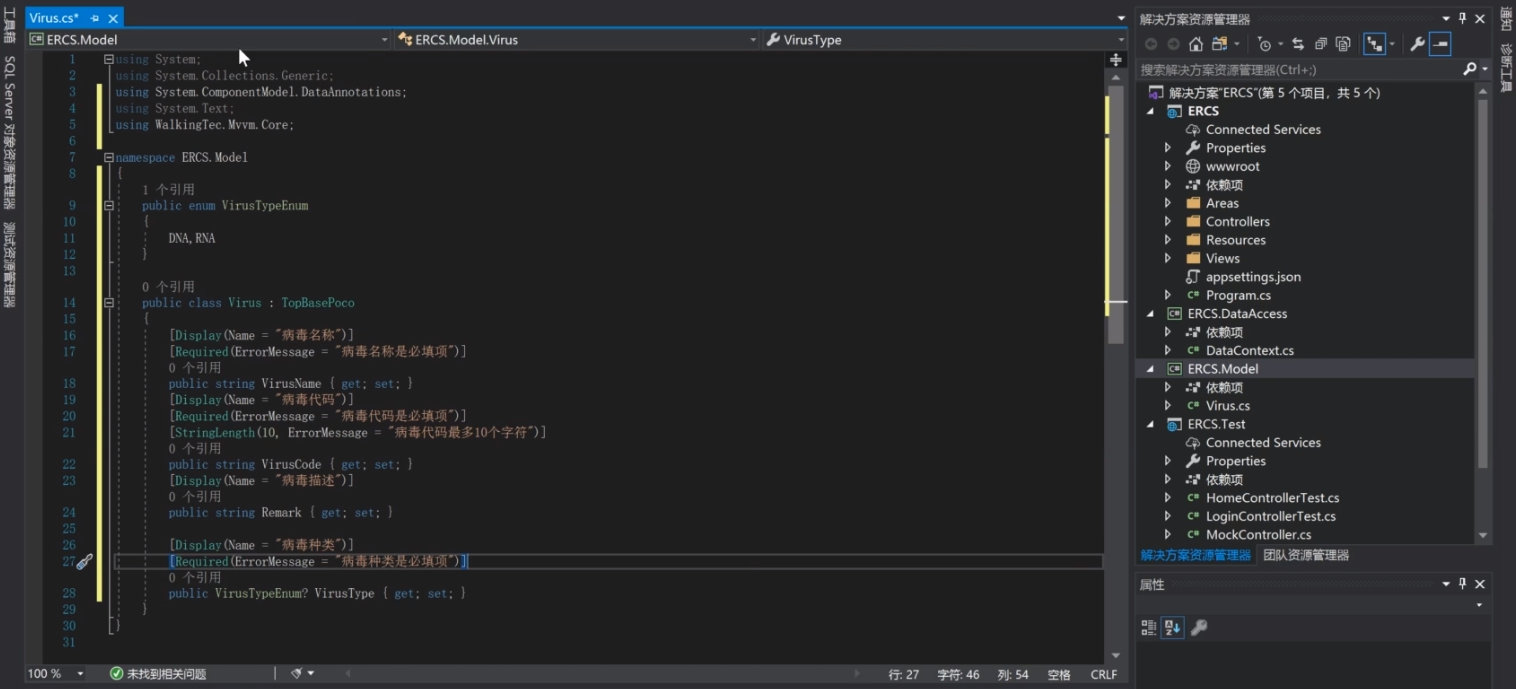
F5 to run

Use Code First

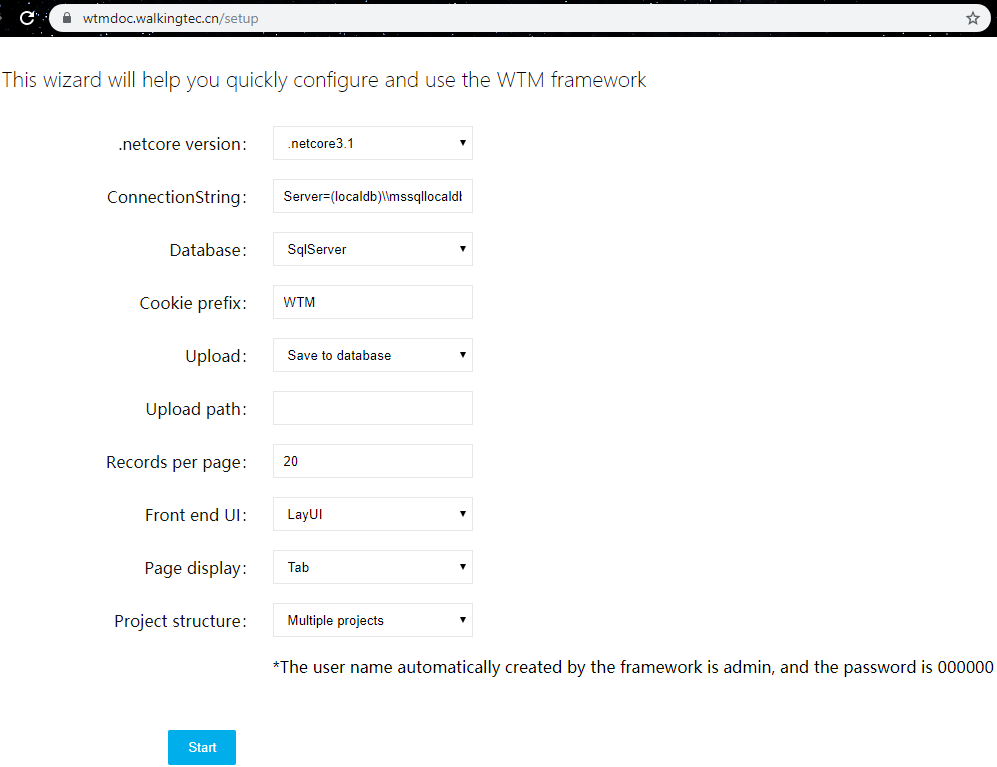
1. Define Virus.cs model in ERCS.Model

* Using WalkingTec.Mvvm.Core
* Inherit TopBasePoco
* Define Enum (if needed)
* Define attribute name (ie. Used by Label in UI)
* Define Validation Rule and Error Message
  + Enum is a required field by default.
  + Suffix ? for optional and then define required attribute to display customized error message



Exercise 1:

1. Generate ERCS



1. Create Virus.cs Model

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

using WalkingTec.Mvvm.Core;

namespace ERCS.Model

{

    public enum VirusTypeEnum

    {

        DNA, RNA

    }

    public class Virus : TopBasePoco

    {

        [Display(Name = "病毒名稱")]

        [Required(ErrorMessage = "病毒名稱是必填欄位")]

        public string VirusName { get; set; }

        [Display(Name = "病毒號碼")]

        [Required(ErrorMessage = "病毒號碼是必填欄位")]

        [StringLength(10, ErrorMessage = "病毒號碼最多10個字符")]

        public string VirusCode { get; set; }

        [Display(Name = "病毒描述")]

        public string Remark { get; set; }

        [Display(Name = "病毒種類")]

        [Required(ErrorMessage = "病毒種類是必填欄位")]

        public VirusTypeEnum? VirusType { get; set; } //Enum must have value by default

    }

}

1. Define DataContext.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

using System;

using System.Collections.Generic;

using System.Linq;

using WalkingTec.Mvvm.Core;

using ERCS.Model;

namespace ERCS.DataAccess

{

    public class DataContext : FrameworkContext

    {

        public DbSet<Virus> Viruses { get; set; }

        public DataContext(CS cs)

             : base(cs)

        {

        }

        public DataContext(string cs, DBTypeEnum dbtype, string version=null)

             : base(cs, dbtype, version)

        {

        }

    }

    /// <summary>

    /// DesignTimeFactory for EF Migration, use your full connection string,

    /// EF will find this class and use the connection defined here to run Add-Migration and Update-Database

    /// </summary>

    public class DataContextFactory : IDesignTimeDbContextFactory<DataContext>

    {

        public DataContext CreateDbContext(string[] args)

        {

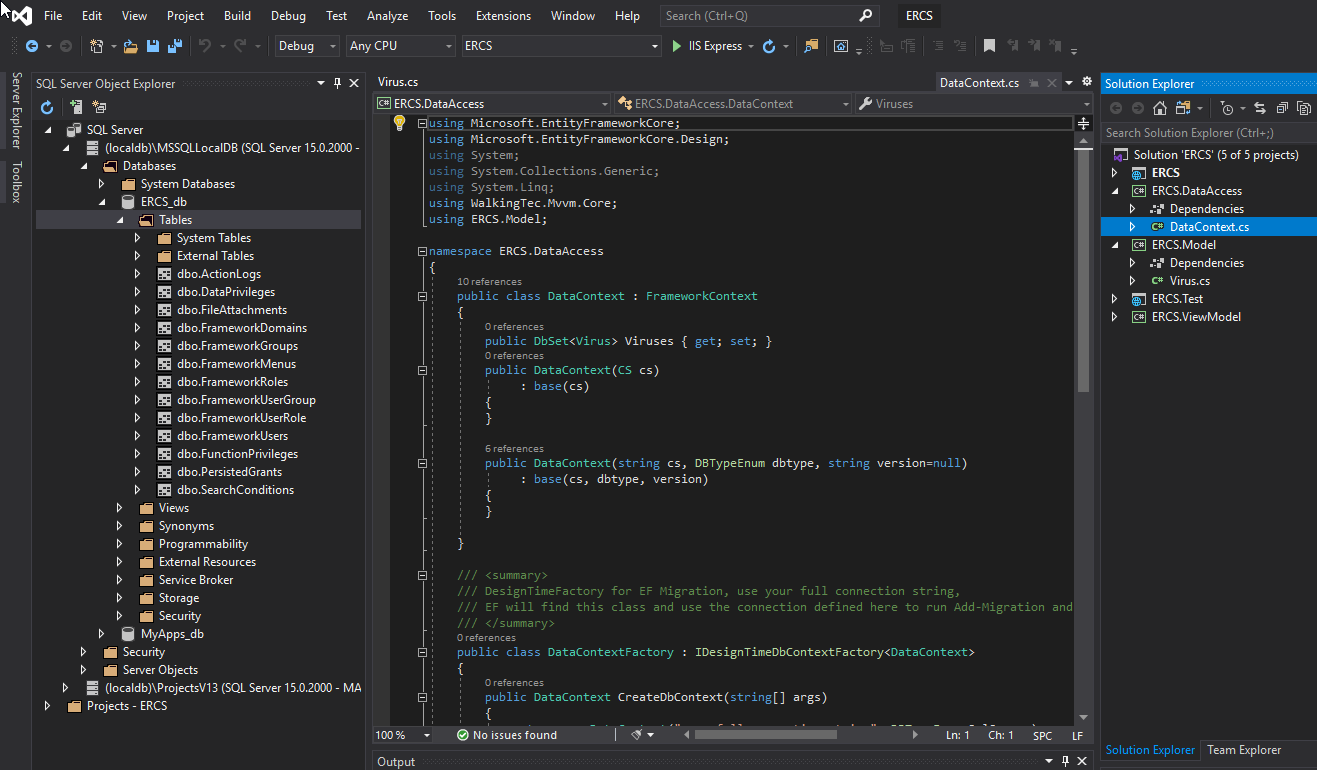
            return new DataContext("your full connection string", DBTypeEnum.SqlServer);

        }

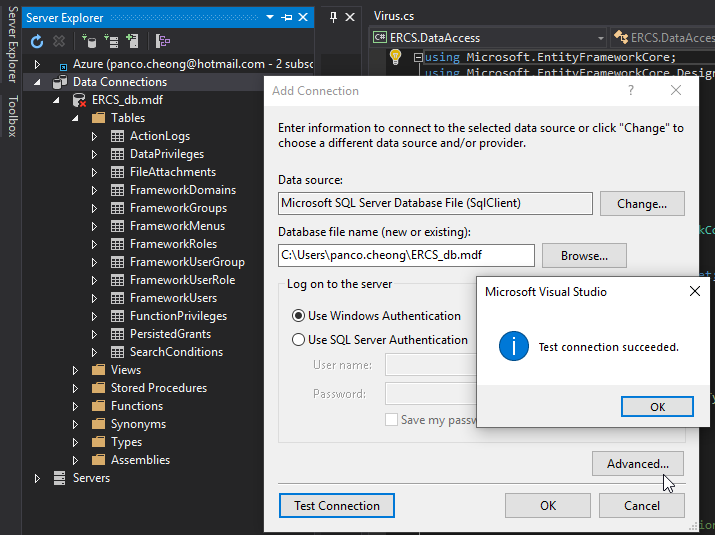
    }

}

SQL Server Object Explorer

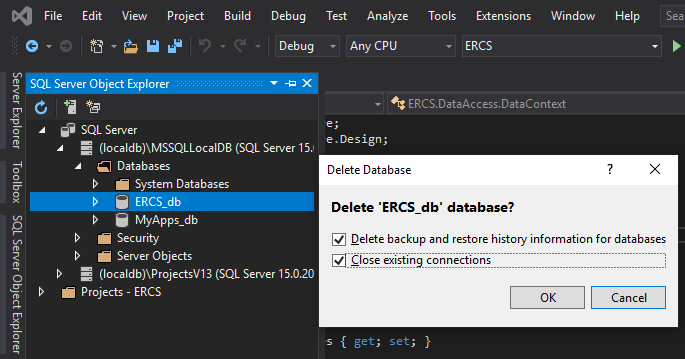


Server Explorer

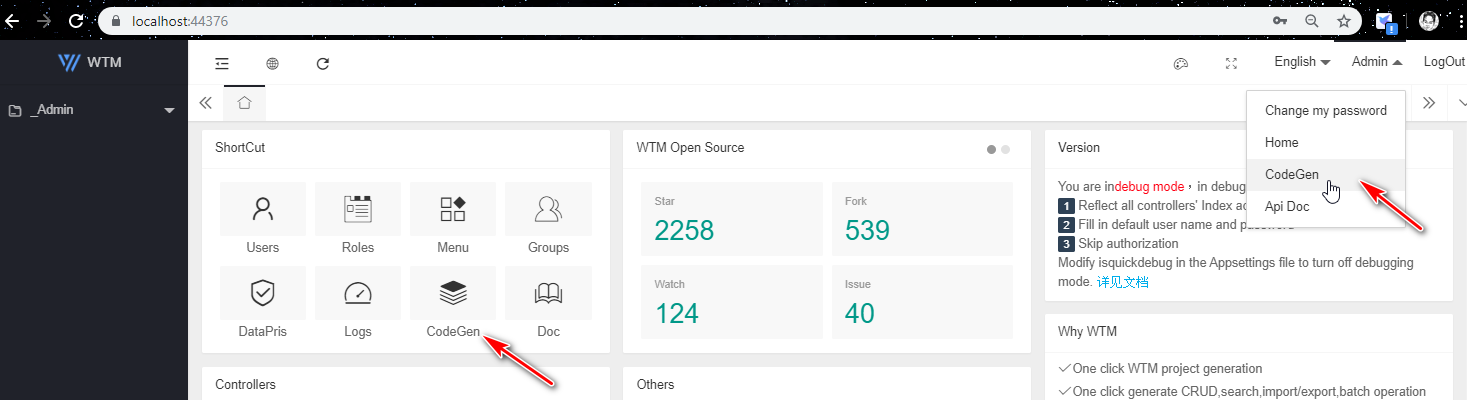


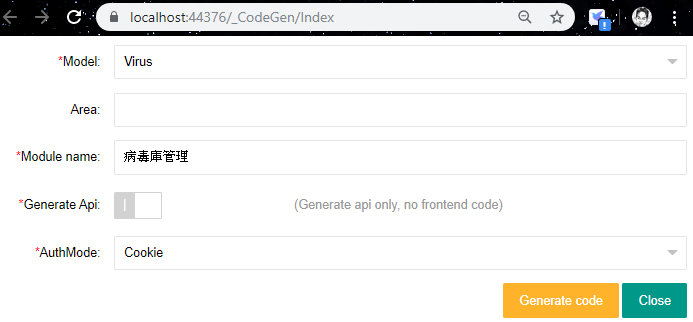
1. Delete database and re-run to re-generate the database, should able to see the dbo.Viruses table

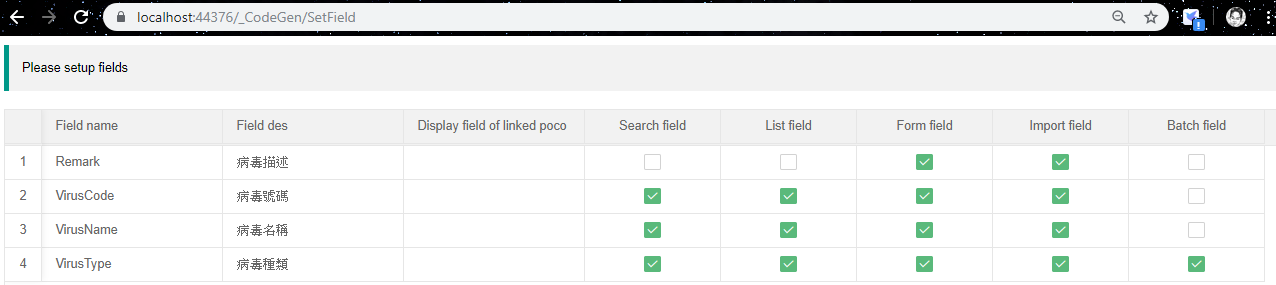
(use other way to add table without delete database in later chapter)



1. Generate code for Viruses







Field des: from the Display attribute of the code

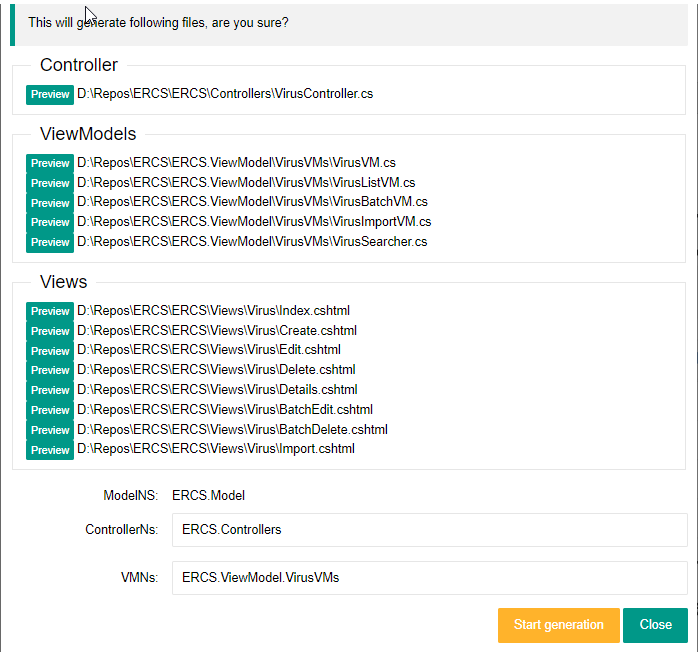
Search field: allow to search for the content in these fields (wild card search, AND condition for multiple criteria)

List field: display columns in List view for multiple items

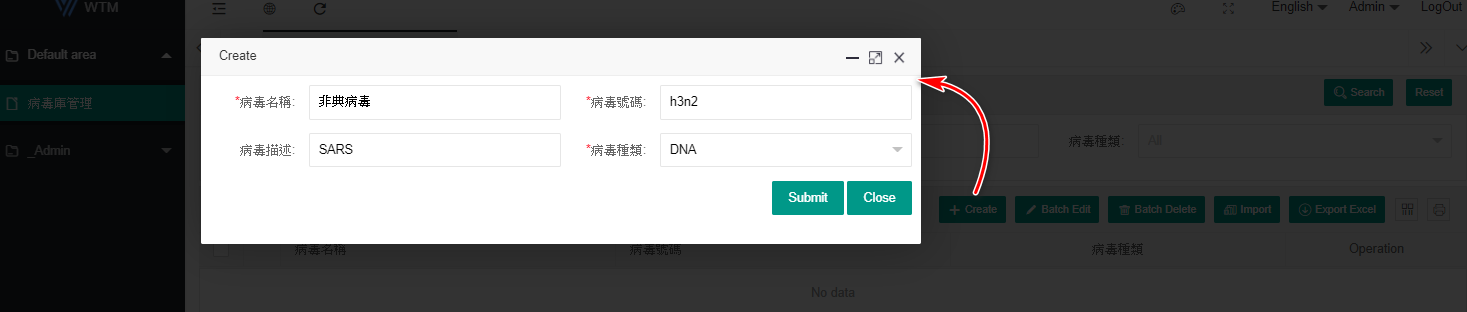
Form field: display fields in Form for individual item

Import field: allow to import from Excel file

Batch field: allow to be updated in batch mode

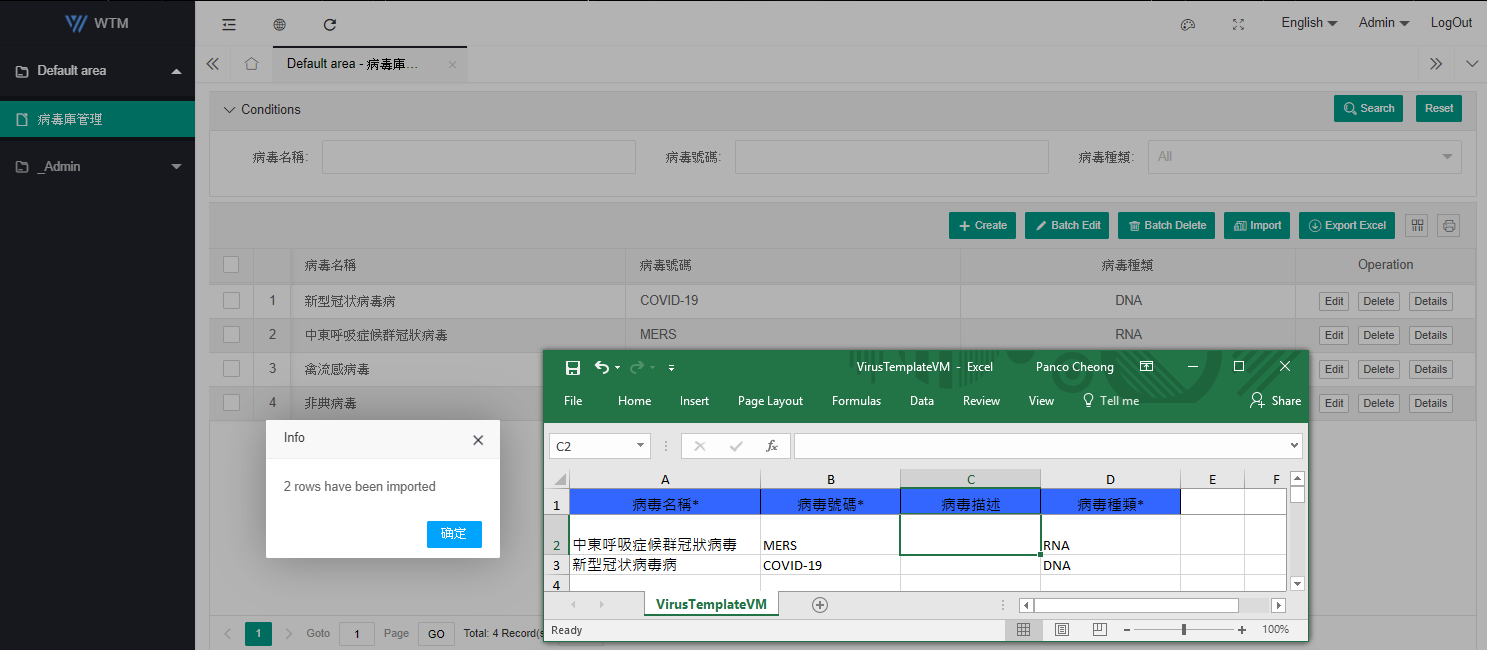


1. Stop debug, re-build the project and re-run
2. Create New Virus entry

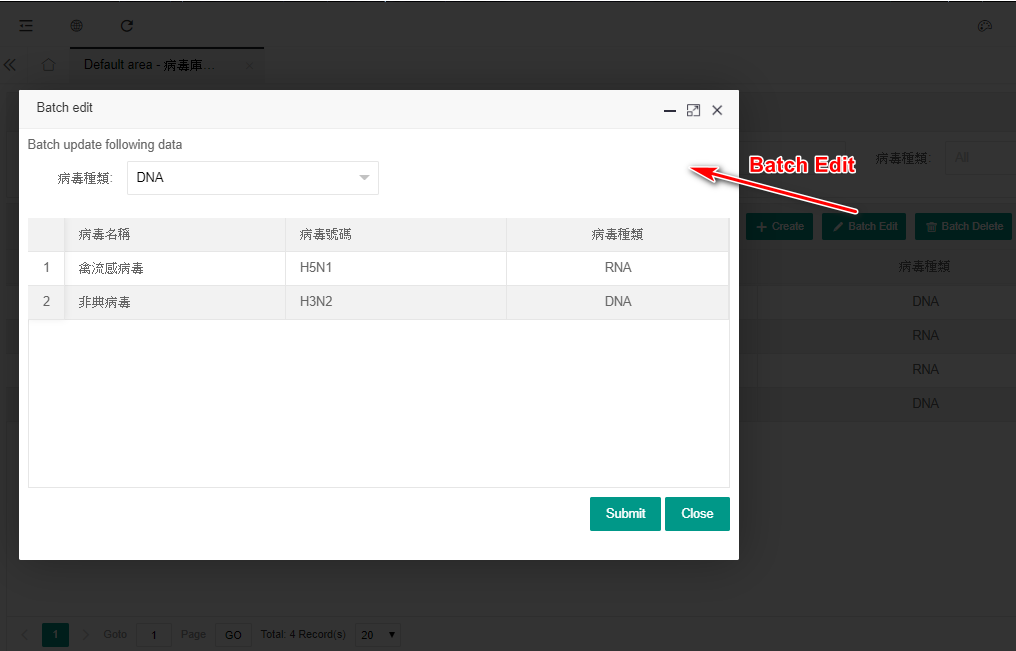




Download Template and Import



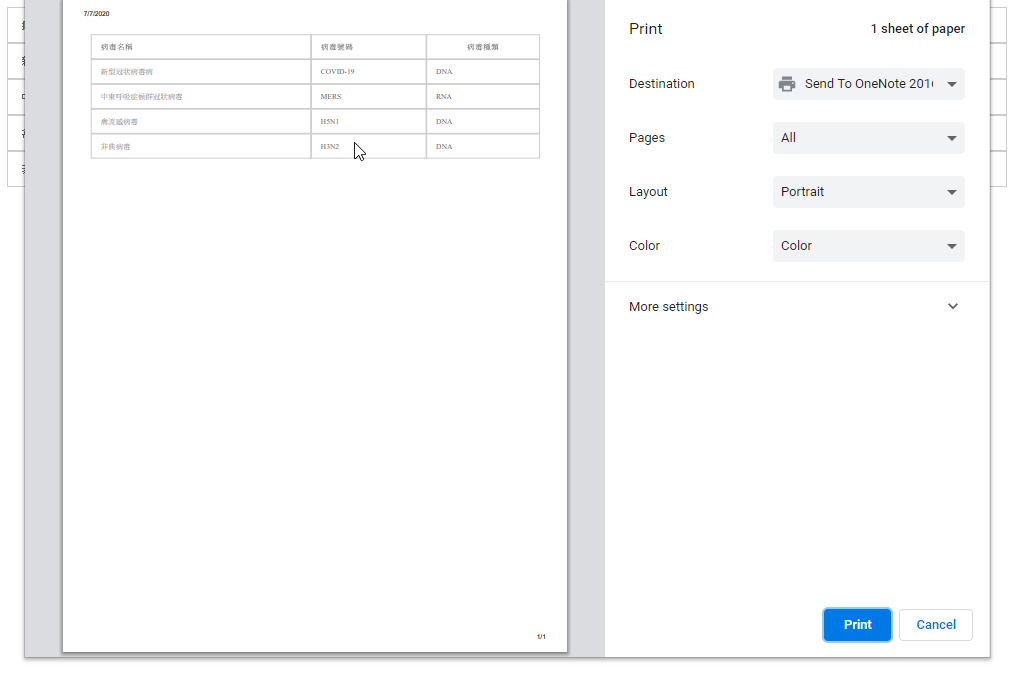
Batch Edit



Export the data

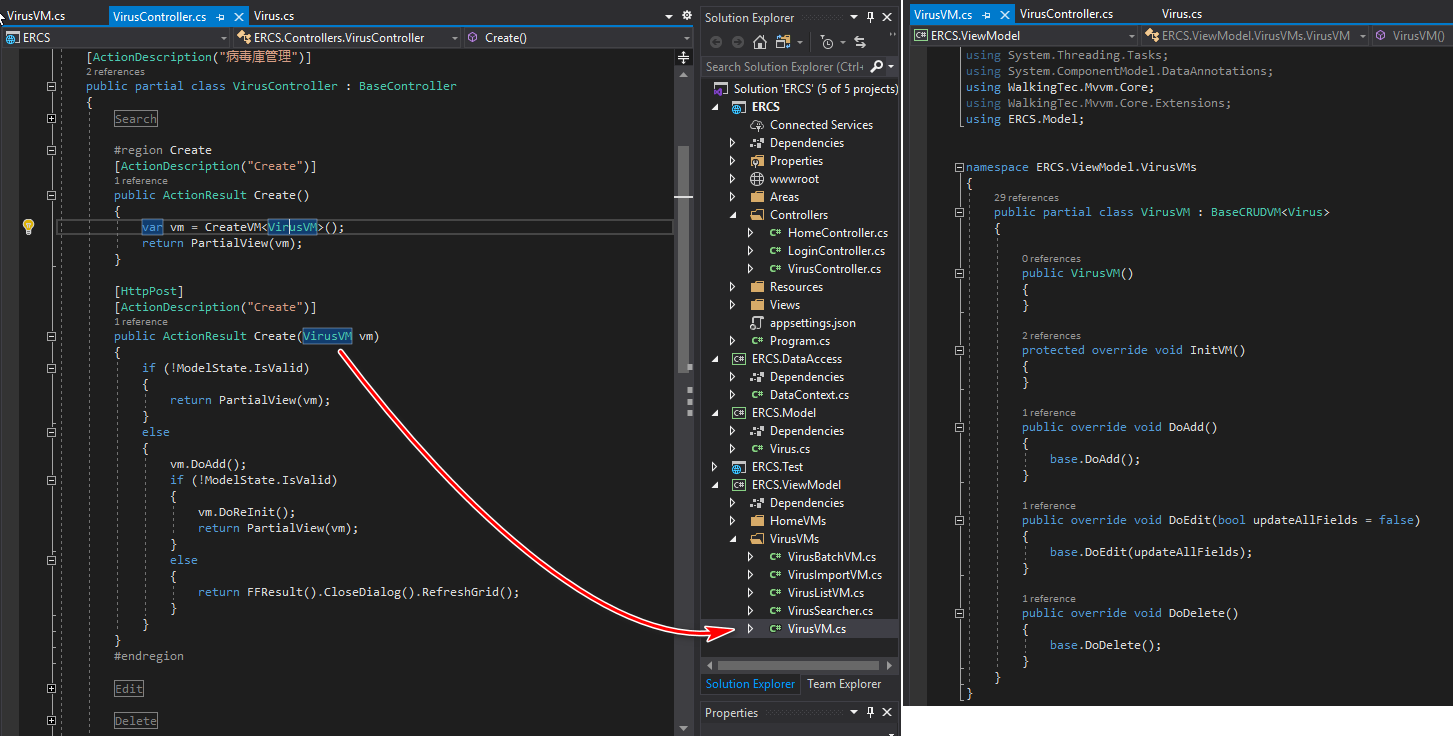
|  |  |  |
| --- | --- | --- |
| 病毒名稱 | 病毒號碼 | 病毒種類 |
| 新型冠状病毒病 | COVID-19 | DNA |
| 中東呼吸症候群冠狀病毒 | MERS | RNA |
| 禽流感病毒 | H5N1 | DNA |
| 非典病毒 | H3N2 | DNA |

Print and Filter Columns



HTTP GET – display the data to Web UI

HTTP POST – update the data from user input in Web UI

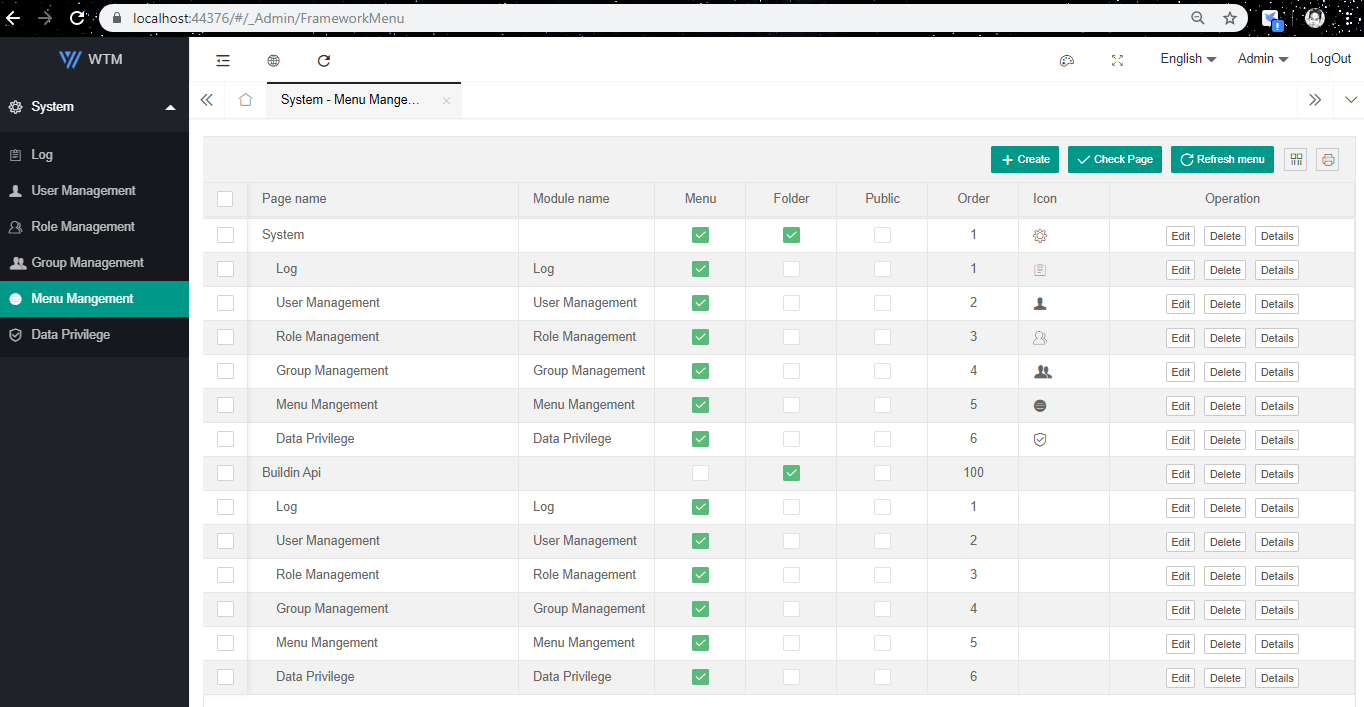


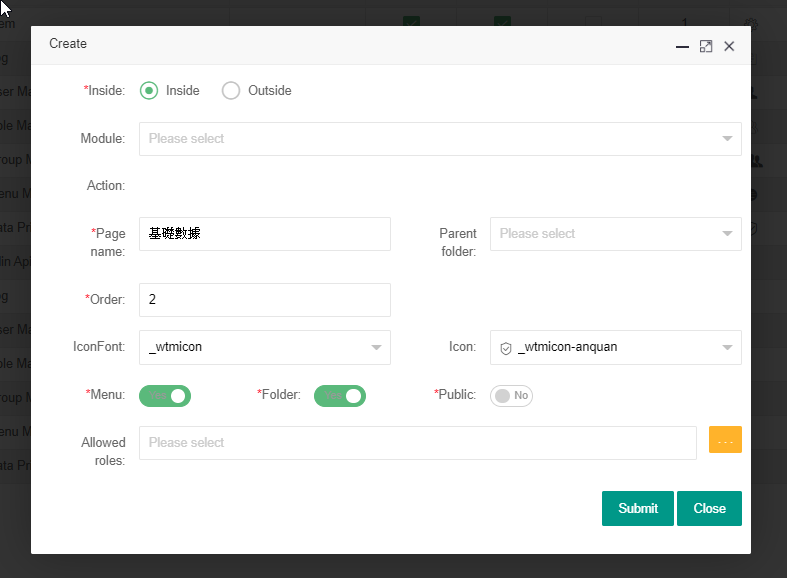
ERCS appsettings.json – configration

"IsQuickDebug": true, //is debug mode

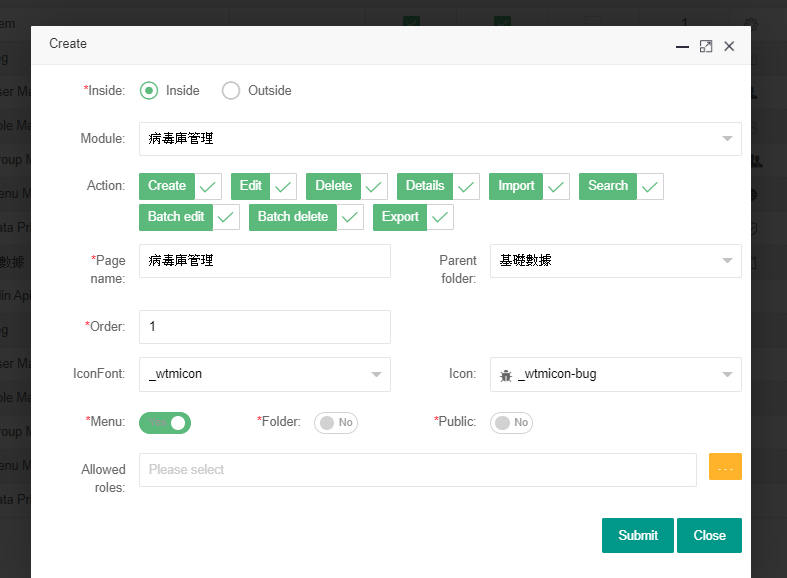
Set to False and re-run again

* Login box will not auto-fill
* The menu will not display all controllers functions, it only list those are being configured in menu management



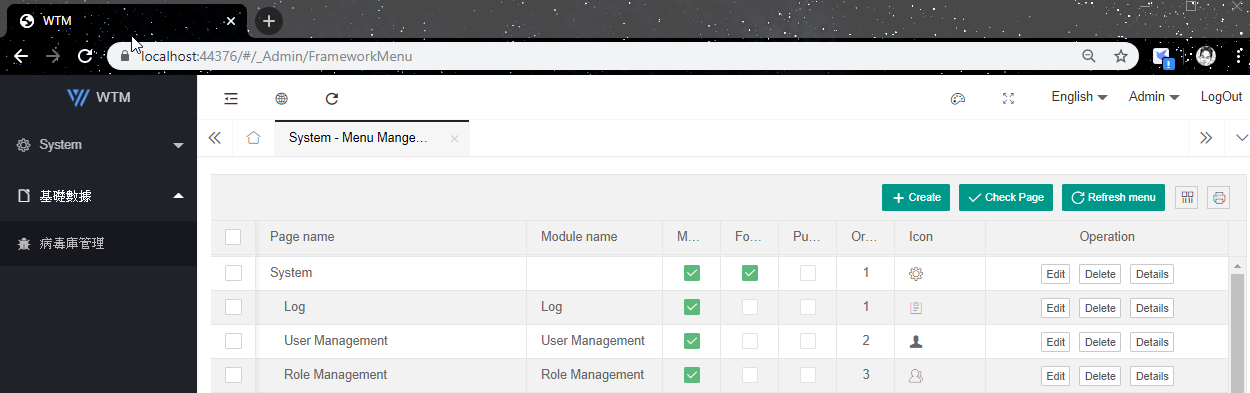


Admin Role is in allowed roles by default.



Click Refresh menu (its trigger to read the menu data from database again)

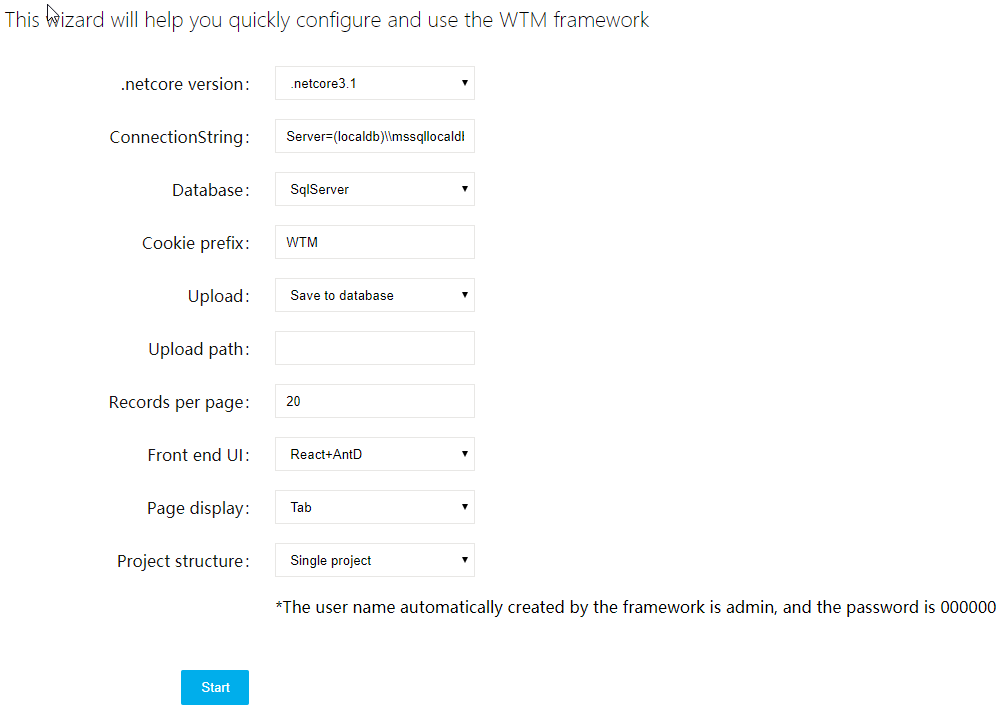
Then reload the page, you should see the menu item.



Separate vs non-Separate

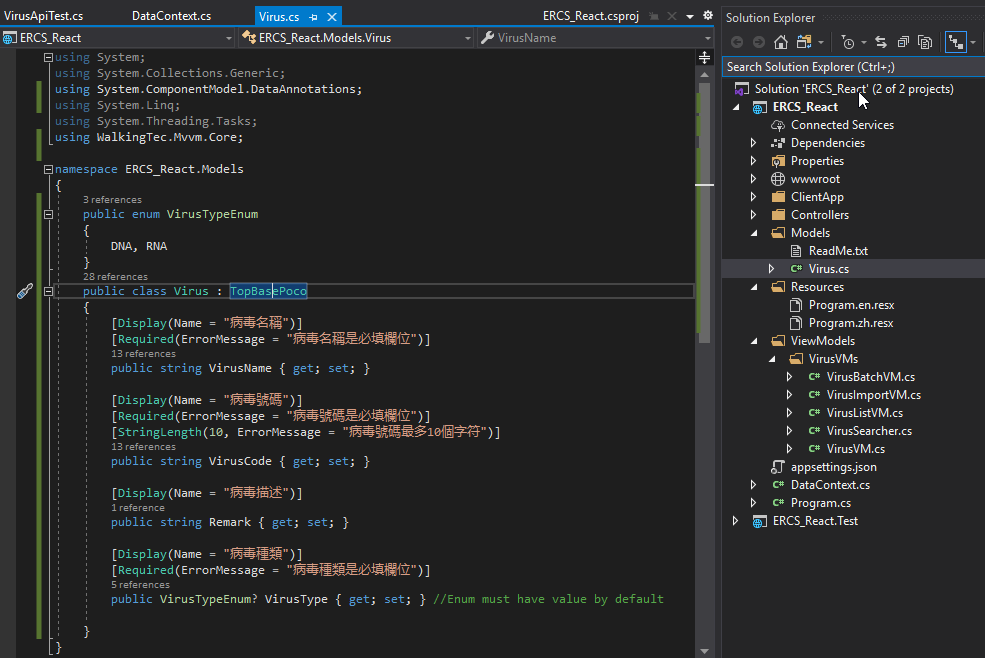
Non-Separate (LayUI): develop faster for small team

Separate (React + AntD or Vue + Element): all more team member to work on together, more work to do



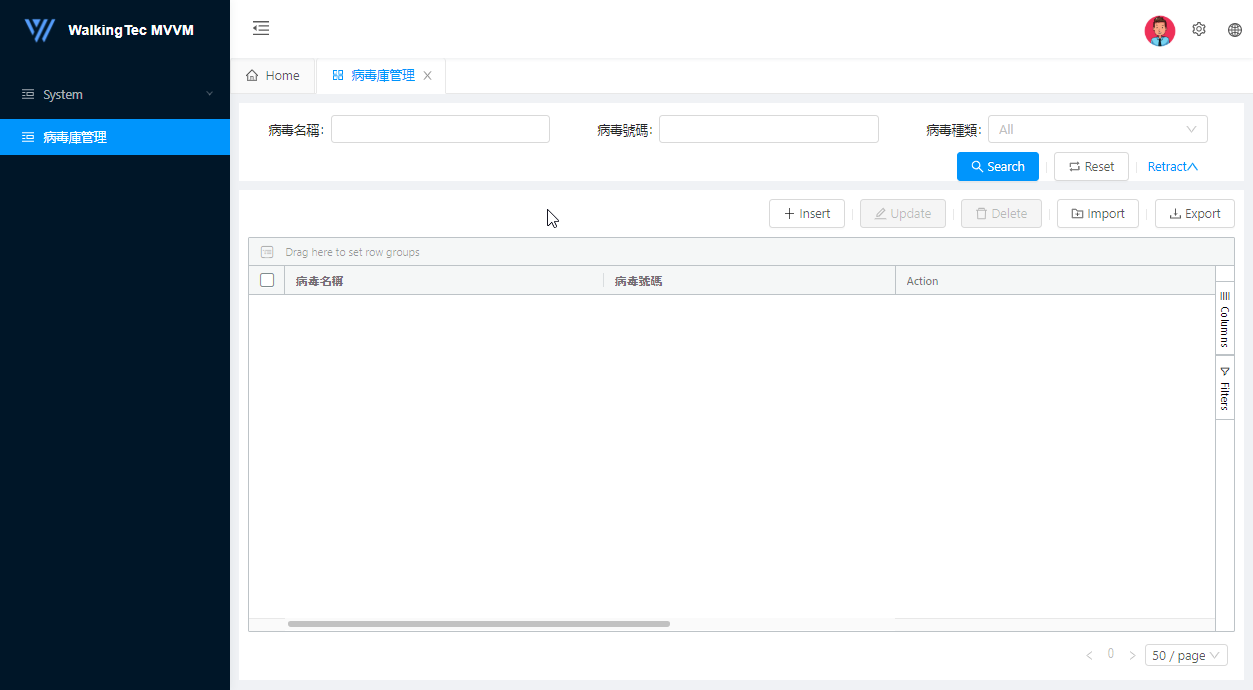
Project Name cannot use dash -, use underscore \_

Single Project will look like this

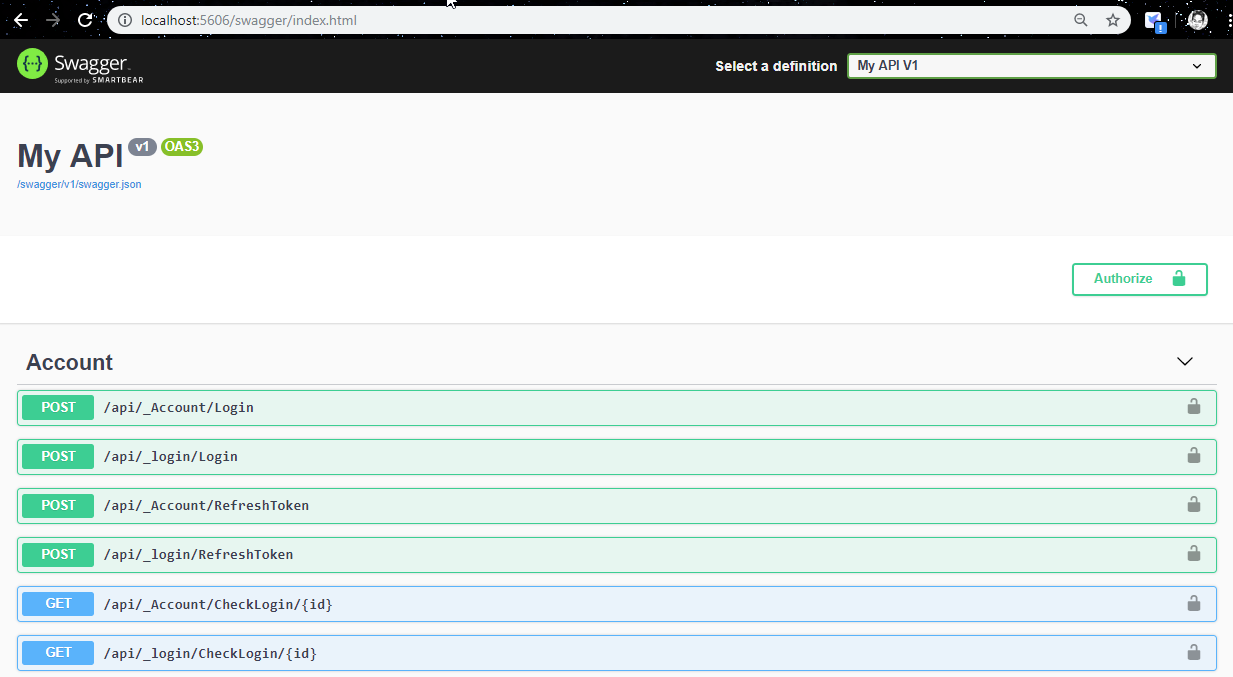


Controllers will use API based methods, still using the ViewModel to transfer data

All the HTML view are located in the ClientApp\src\pages folder (either React or Vue)



API Documentation



模拟一个肺炎疫情上报防控系统的开发过程,系统的功能有:

- 基础数据的维护(省市区县，医院，疾控中心，病毒库等)

- 病例上报

- 病例接触者维护

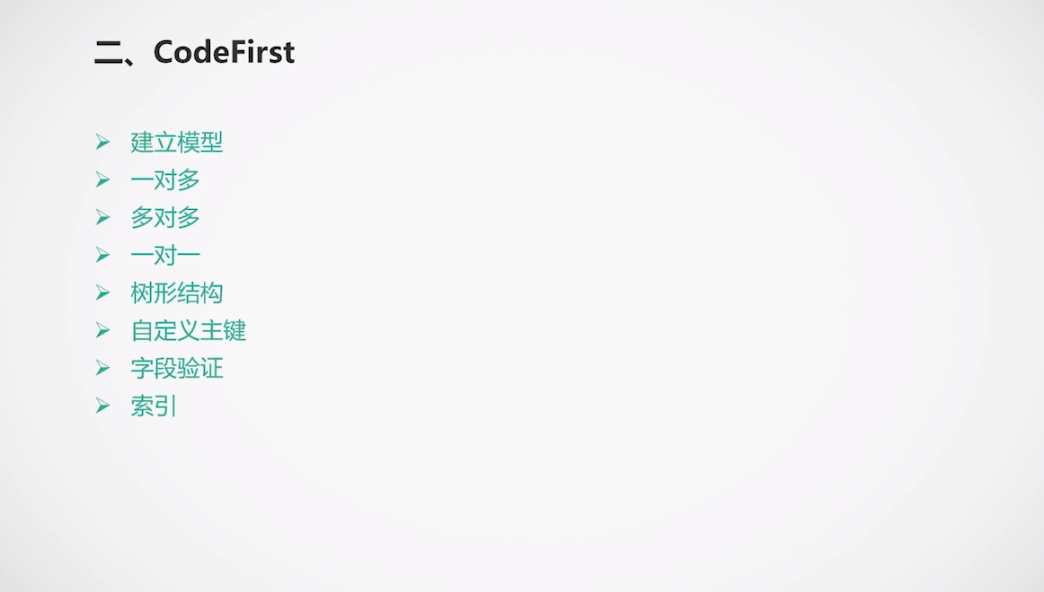
- 病例及其接触者回访/追踪

- 统计报表

- 基于角色的功能权限控制 (医生可上报病情，疾控中心人员可查看数据)

- 基于用户组的数据数据权限 (医生可操作自己上报的病情，医院管理人员可查看本医院所有病情，区县可看辖区内所有医院的病情等)

- App，小程序，手机端H5等也可实现上报和统计功能 (教程只演示H5，其他同理)



Code First – focus on coding, no need to define database first

<https://www.bilibili.com/video/av86527514/?p=3>

//Hospital.cs

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

using WalkingTec.Mvvm.Core;

namespace ERCS.Model

{

    public enum HospitalLevel

    {

        [Display(Name = "三級醫院")]    // can set attribute to Enum

        Class3,

        [Display(Name = "二級醫院")]

        Class2,

        [Display(Name = "一級醫院")]

        Class1

    }

    public class Hospital : TopBasePoco

    {

        [Display(Name = "醫院名稱")]

        public string Name { get; set; }

        [Display(Name = "醫院級別")]

        public HospitalLevel Level { get; set; }

        // Location and LocationId are tied together to create a table relationship

        // one-to-many relationship: one City to many Hospital

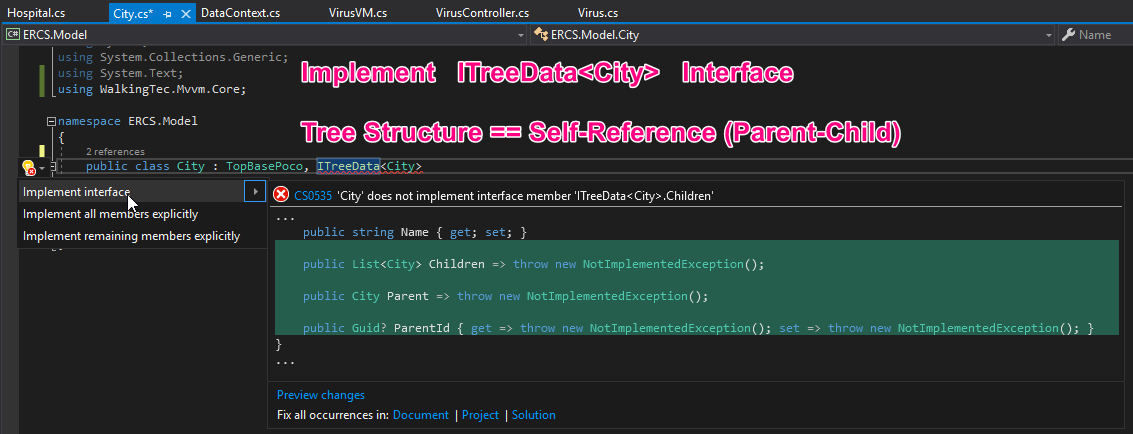
        public City Location { get; set; }      //create relationship with City

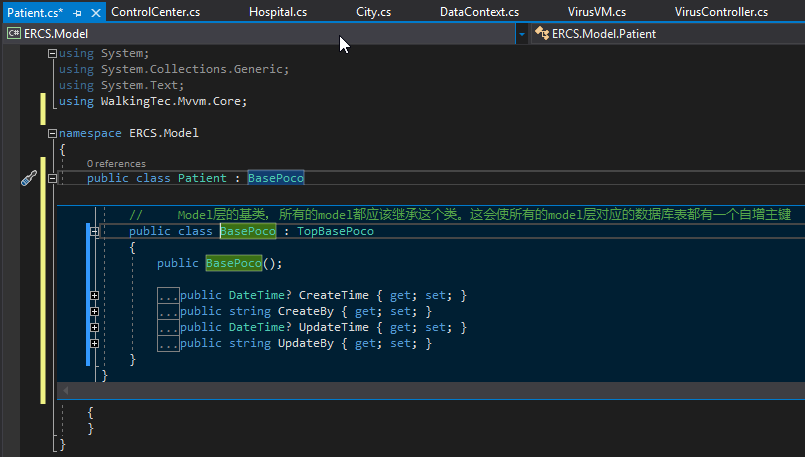
        [Display(Name = "醫院地點")]

        public Guid LocationId { get; set; }    //Foreign Key is LocationId

    }

}





// TopBasePoco <-- BasePoco <-- PresistPoco

// BasePoco has 4 more properties than TopBasePoco: CreateTime, CreateBy, UpdateTime, UpdateBy

// PersistPoco has 1 more property than BasePoco: IsValid (avoid physically delete record, just marked invalid)

模拟一个肺炎疫情上报防控系统的开发过程,系统的功能有:

- 基础数据的维护(省市区县，医院，疾控中心，病毒库等)

- 病例上报

- 病例接触者维护

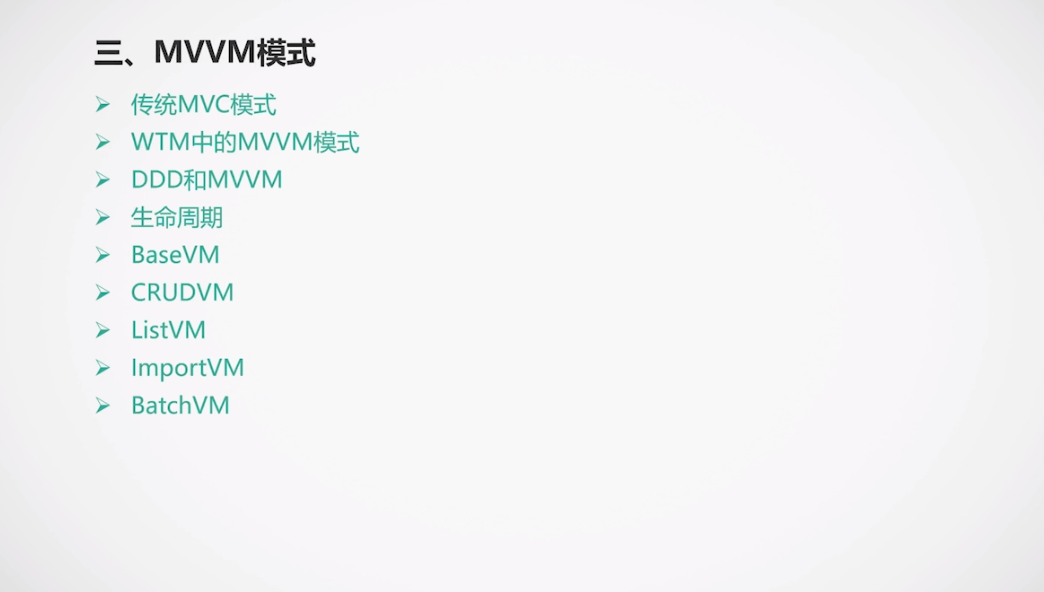
- 病例及其接触者回访/追踪

- 统计报表

- 基于角色的功能权限控制 (医生可上报病情，疾控中心人员可查看数据)

- 基于用户组的数据数据权限 (医生可操作自己上报的病情，医院管理人员可查看本医院所有病情，区县可看辖区内所有医院的病情等)

- App，小程序，手机端H5等也可实现上报和统计功能 (教程只演示H5，其他同理)



Domain Driven Development



