1.数据库上下文

Using Microsoft.EntityFrameworkCore;

Namespace RazorPagesContacts.Data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions options)

: base(options)

{

}

Public DbSet<实体类> Customers {get; set ;}

}

}

2. appsettings.json中写入的连接字符串

"ConnectionStrings": {

"DefaultConnection": "Data Source=WB19;Initial Catalog=MyWebSite;Integrated Security=True"

},

3.在Startup.cs对上下文进行依赖注入

public void Configure Services(IServiceCollection services)

{

services.AddDbContext<ApplicationContext>(optisons =>

optisons.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddMvc ();

}

4.控制器db声明使用

namespace RazorPagesContacts.Pages

{

Public class Create Model: Page Model

{

Private read-only AppDbContext \_db;

Public Create Model (AppDbContext db)

{

\_db = db;

}

Public sync Task<IActionResult> OnPostAsync ()

{

If (! ModelState.IsValid)

{

Return Page ();

}

\_db.Customers.Add (Customer);

await \_db.SaveChangesAsync ();//采用异步方式

Return RedirectToPage ("/Index");

}

}

}

5.使用form表单进行提交

参考：https://www.cnblogs.com/linfenghp/p/7140369.html

<**form** class="form-signin" role="form" id="a1" method="post" **asp-action**="Login2">

<h1>Login/登陆</h1>

<input type="text" class="form-control" name="user" placeholder="用户名" required autofocus />

<input type="password" class="form-control"name="password" placeholder="密码" required />

<button class="btn btn-lg btn-warning btn-block" id="submit" type="submit">登录</button>

<label class="checkbox"><input type="checkbox" value="remember-me"> 记住我</label>

</**form**>

6.控制器接收对应的值

public IActionResult Login2(string user , string password)

{

User us = new Models.User();

us.UserName = user;

us.Password = password;

return View();

}

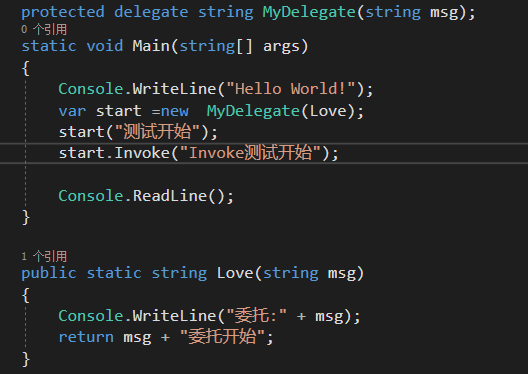
表单信息与文件同时上传。

<https://developer.mozilla.org/zh-CN/docs/Web/API/FormData/FormData>

文件上传以及导入导出。

<https://www.cnblogs.com/CreateMyself/p/6464199.html>

**委托**是声明一个方法类型，使用时new这个委托，再以参数的形式指明方法，就可以通过委托来使用方法简而言之：我委托一个人（指方法）去做一件事（执行这个方法）



**接口**是用于规范类中的方法

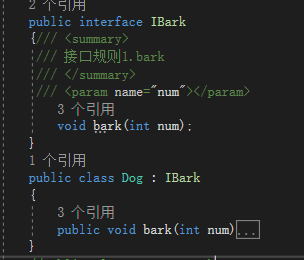
比如 public interface IBark接口中包含了 bark（）和eat（）两个方法

则继承该接口的类都必须拥有这两个方法。

使用，通过接口new继承的类，新的对象.出接口中指定的方法。

IBark bark =new Dog();

简而言之：电脑接口是USB（指接口要求的规范）的 你想用鼠标（使用接口中的方法）就要鼠标连接电脑 但你鼠标也要是USB的（类中必须包含接口所含有的方法）



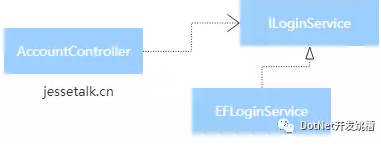
依赖

例如：AccountController这个控制器需要完成和用户相关的注册、登录 等事情。

但登陆需要EF和Idnetity来完成因此我们封装一个EFLoginService

为了体现原则，依赖于抽象而不是具体实现，所以我们给EFLoginService定义了一个接口，抽象了LoginService的行为

简而言之：比如上下文写好了方法，去注入IServiceCollection这个接口下对应的AddDbContext方法开启这个功能



反转把依赖的创建丢给其它人，自己只负责使用，其它人丢给你依赖的这个过程理解为注入。简而言之只管使用就ok了，具体依赖什么怎么依赖的不用考虑



为下个项目做准备，学习计划先搁置…