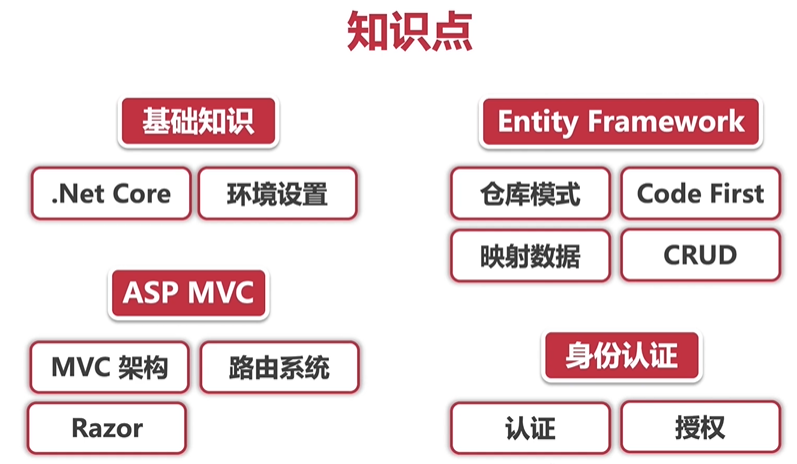
打造你的第一个ASP.NET Core MVC网站应用

## 第1章 ASP.net Core简介

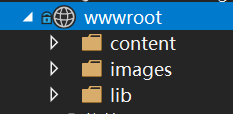
课程知识点



## 第2章 创建实战项目

老师项目github地址：<https://github.com/Yaduo/LanzhouBeefNoodles>

wwwroot文件夹存放前端静态文件；如js、css、图片、视频



2.3.Program与startup

Startup类主要有两个作用：

1. 在ConfigureServices里面注入各种服务组件的依赖；

2. 创建中间件，设置请求通道Request Pipeline。

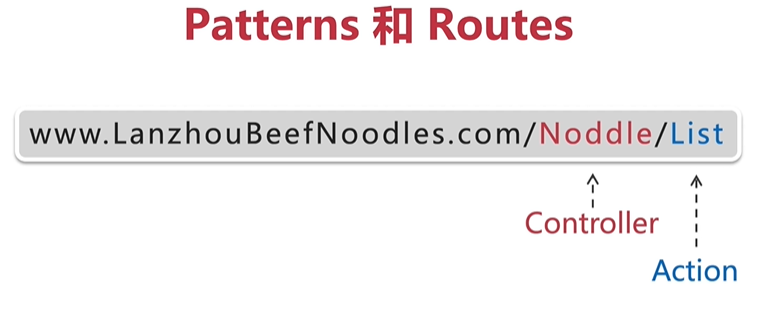
|  |
| --- |
| public class Startup  {  // This method gets called by the runtime. Use this method to add services to the container.  // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940  //在ConfigureServices里面注入各种服务组件的依赖  public void ConfigureServices(IServiceCollection services)  {  //注册Mvc服务的组件  services.AddMvc();  }  // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.  //创建中间件，设置请求通道Request Pipeline  public void Configure(IApplicationBuilder app, IHostingEnvironment env)  {  if (env.IsDevelopment())  {  app.UseDeveloperExceptionPage();  }  //对请求url进行截获处理，异步输出hello from test  app.Map("/test", build =>  {  build.Run(async context =>  {  await context.Response.WriteAsync("hello from test");  });  });  app.Run(async (context) =>  {  await context.Response.WriteAsync("Hello World!");  });  //任何在app.Run之后的代码都不会被执行  }  } |

环境变量设置如下：



## 第3章 深入理解路由

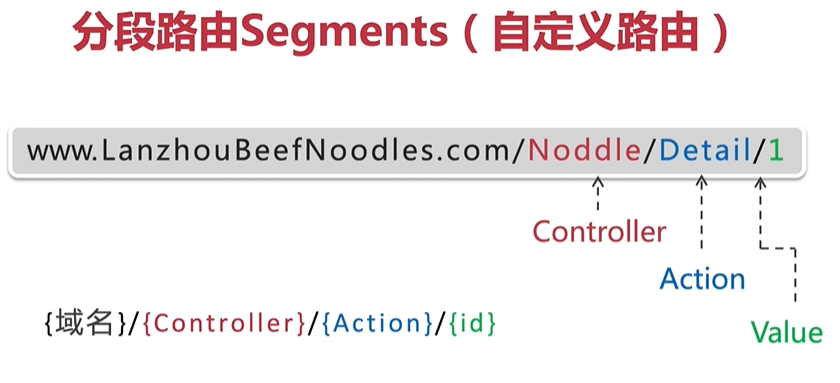
页面url请求在MVC中分别由不同的组件进行处理



url地址映射规则：



带请求参数的url映射地址：



url地址映射规则：



2.1.如何定义url和Controller之间的映射规则？



2.2.自定义默认路由



### 3.1. 路由

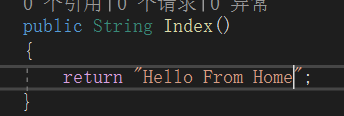
1.路由分为传统路由和特征注解路由

传统默认路由：

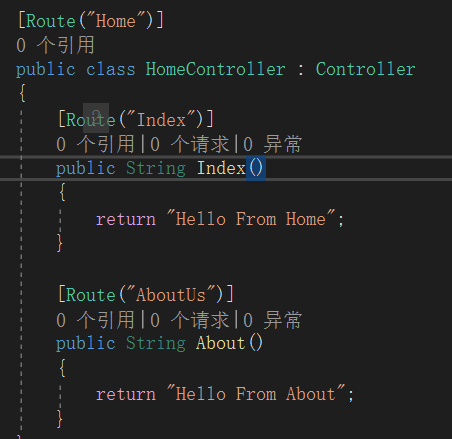
Configure方法中添加如下代码：

|  |
| --- |
| app.UseMvcWithDefaultRoute(); |

会自动映射到：



特征注解路由：



统一配置路由映射：



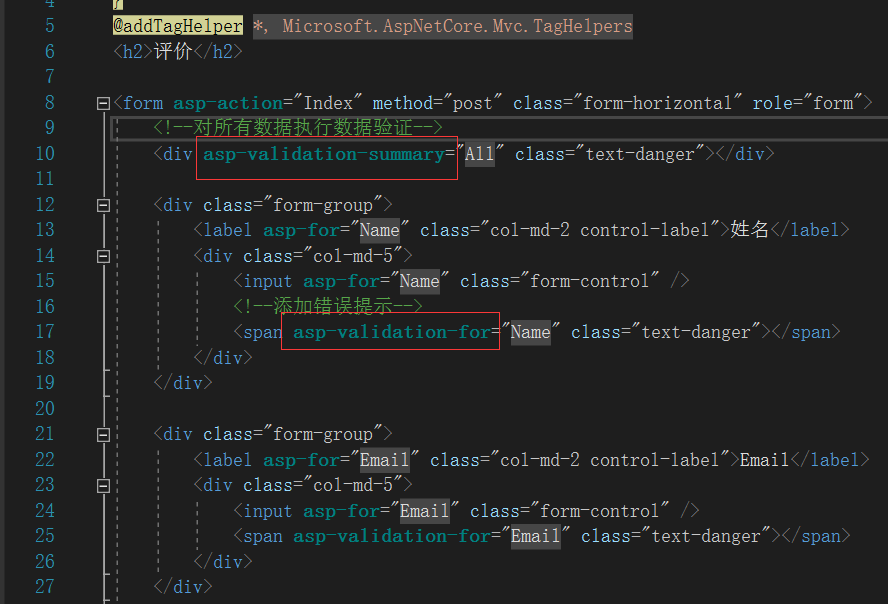
## 第5章 数据模型

视图模型 View Model

View Model中数据与UI界面双向绑定，一旦ViewModel中的数据发生任何变化，结果都会立即反应在UI界面中。

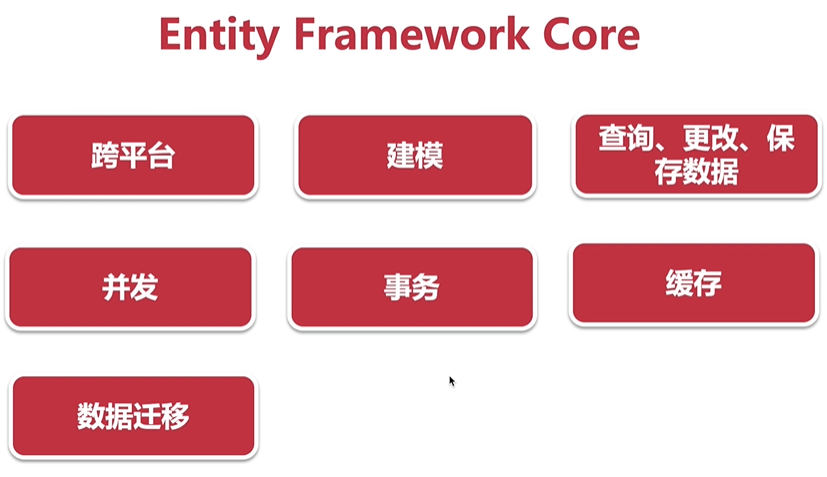
Tag Helper功能：可以进行数据验证。是ModelState内置对象中的一个功能。

元素的颜色改变了。



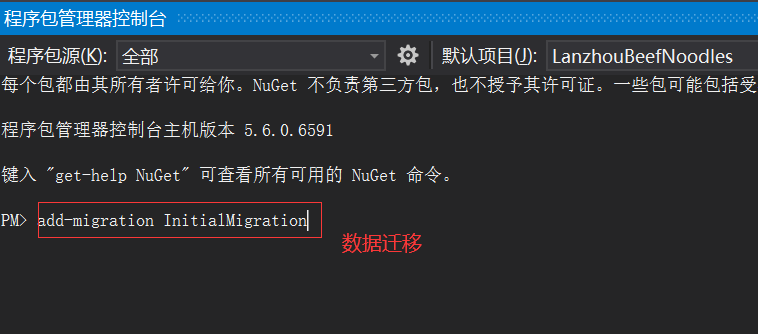
|  |
| --- |
| @addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers |

## 第6章 Entity Framework Core



第6章 EF Core

add-migration 迁移名称

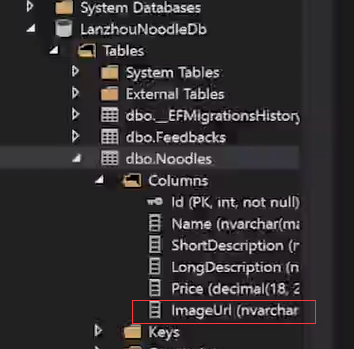


update-database：会执行Up函数来执行数据迁移

完成后可以点击

增加列：add-migration 新增加的名字（NoodleChanges），对应会变更Migrations的数据；

update-database：对应会更新表中的ImageUrl数据列



## 第7章 视图

1.

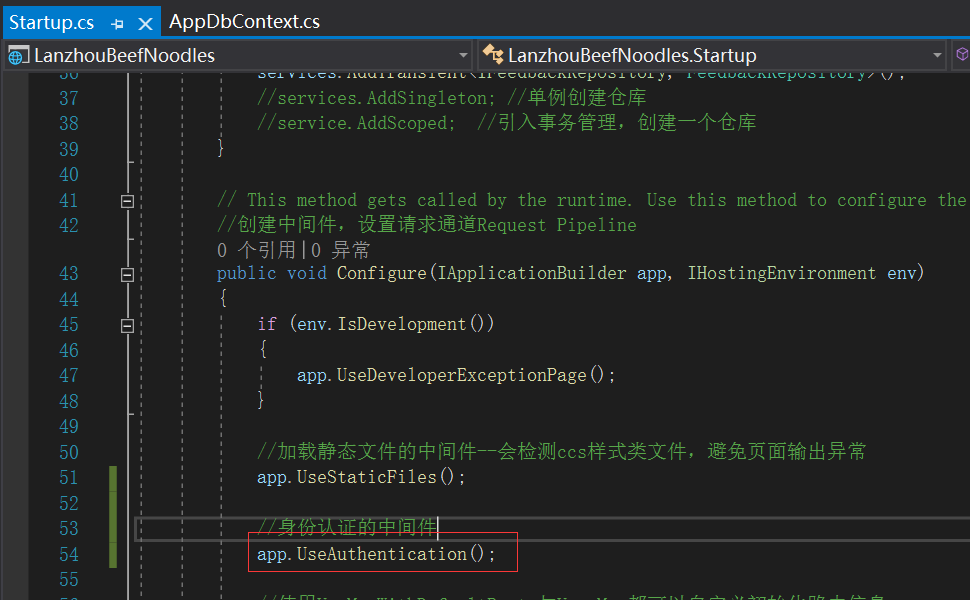
@RenderBody()就是一个占位符，页面在打开时，就会根据控制器和View视图加载对应的页面信息。

## 第8章 ASP.NET Core Identity

ASP.NET Core Identity主要用于ASP.NET 身份验证，用户授权登录、用户注销等访问权限的控制。

使用方式：

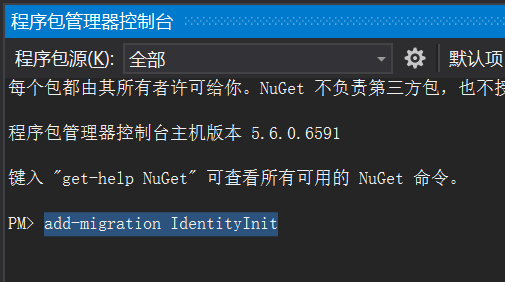
用AppDbContext继承IdentityDbContext<IdentityUser>，自动添加用户表，进行权限管理。



更新数据库：

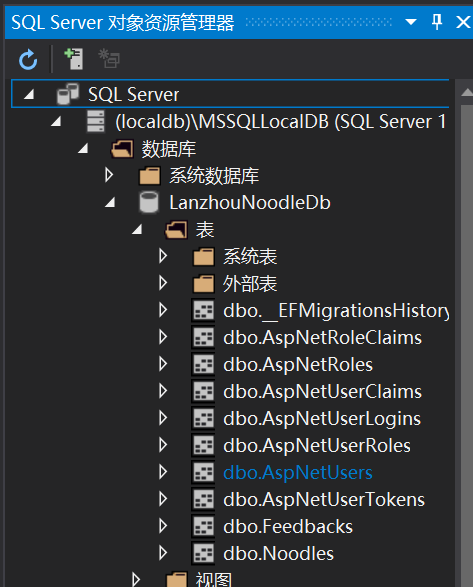
创建身份认证相关的数据库。

add-migration IdentityInit

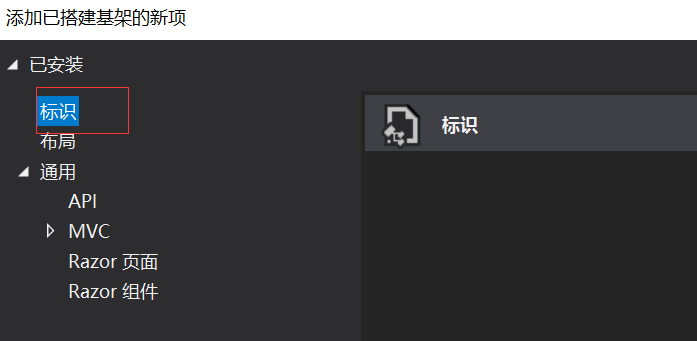


update-database

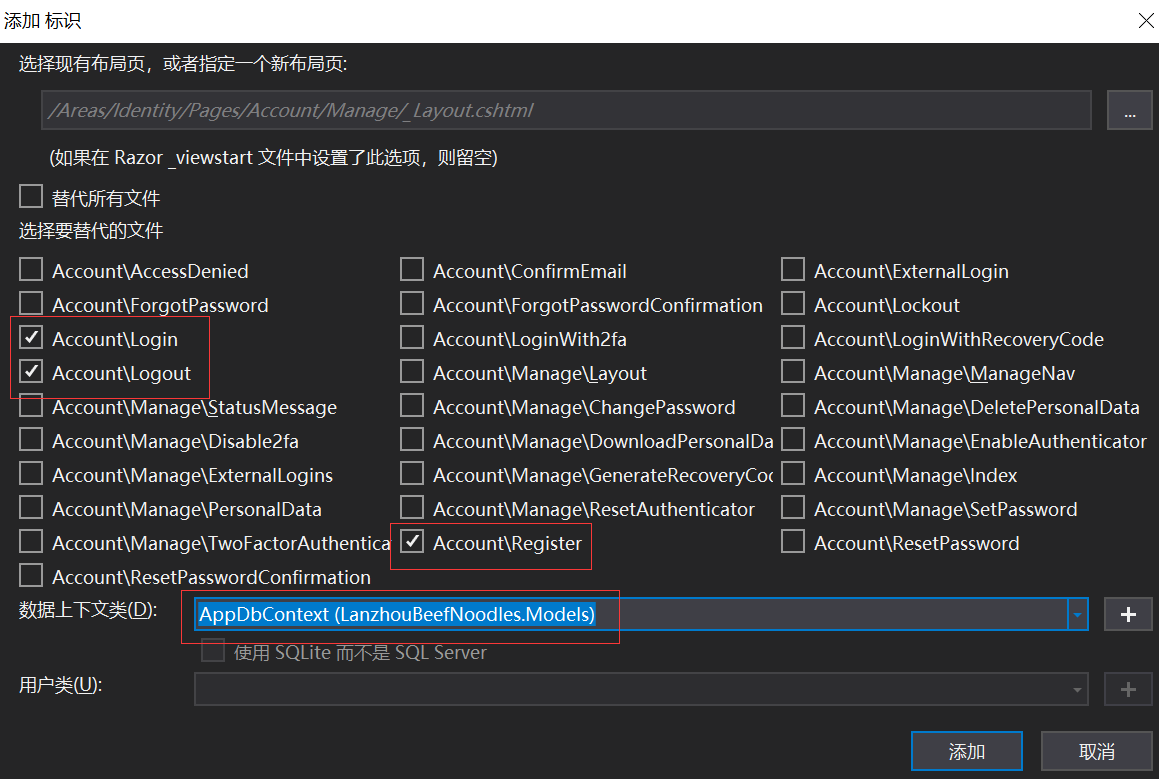
正式启用身份认证功能：



### 8.1.创建Identity身份认证管理



使用Identity脚手架完成页面的添加功能：



### 8.2.简单用户授权管理

基于前台页面做用户登录后的意见评价处理，需要在评价页面控制器Controller中进行用户授权管理。

