# 自定义身份验证用户类

主要过程

1.自定义用户数据支持通过继承IdentityUser。 此类型命名惯例做法ApplicationUser:

|  |
| --- |
| public class **ApplicationUser** : **IdentityUser**  {  public string CustomTag { get; set; }//自定义用户标识  } |

2.使用ApplicationUser作为上下文的泛型参数的类型

|  |
| --- |
| public class **ApplicationDbContext** : IdentityDbContext<**ApplicationUser**>  {  public ApplicationDbContext(DbContextOptions<**ApplicationDbContext**> options)  : base(options)  {  }  protected override void OnModelCreating(ModelBuilder builder)  {  base.OnModelCreating(builder);  }  } |

3.迁移数据库

4.配置startup.cs

|  |
| --- |
| services.AddDefaultIdentity<**ApplicationUser**>()  .AddEntityFrameworkStores**<ApplicationDbContext**>()  .AddDefaultUI(); |

5.用户身份认证

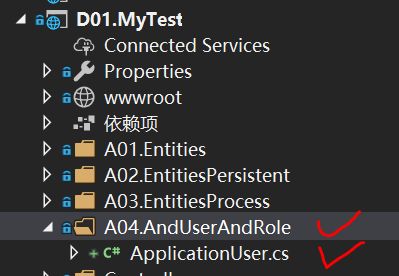
|  |
| --- |
| SignInManager<ApplicationUser> 认证管理  UserManager<ApplicationUser> 用户管理 |

6.本演练（演练4），使用业务类访问用户管理类

# 演练4—自定义扩展用户管理

## 自定义身份验证用户

1. 在项目中创建A04.AndUserAndRole文件夹，并创建ApplicationUser类



ApplicationUser类是扩展IdentityUser，其命名为习惯命名。

1. 在ApplicationUser类代码如下

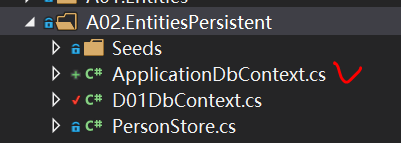
|  |
| --- |
| public class ApplicationUser : IdentityUser  {  [StringLength(100)]  public string FirstName { get; set; } // 姓氏  [StringLength(100)]  public string LastName { get; set; } // 名字  [StringLength(100)]  public string ChineseFullName { get; set; } // 中文全名  [StringLength(50)]  public string MobileNumber { get; set; } // 移动电话，父类中的 PhoneNumber 用于固定电话  public ApplicationUser(string userName) : base(userName)  {  this.UserName = userName;  }  } |

## 新增自定义身份验证用户数据库访问上下文

使用ApplicationUser作为上下文的泛型参数的类型。该上下文也可以使用原有业务上下文，但是需要继承IdentityDbContext<ApplicationUser>

1. 任务在A02.EntitiesPersistent文件夹下新建ApplicationDbContext类

效果如下：



1. ApplicationDbContext代码如下

|  |
| --- |
| public class ApplicationDbContext : IdentityDbContext<ApplicationUser>  {  public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)  : base(options)  {  }  protected override void OnModelCreating(ModelBuilder builder)  {  base.OnModelCreating(builder);  }  } |

## 修改标识配置代码

1. startup.cs中修改上下文和注册服务

效果如下其中，D01.MyTest为项目名称

|  |
| --- |
|  |

1. 在所有视图开始处添加引用（在需要的时候添加）

|  |
| --- |
| @using Microsoft.AspNetCore.Identity  @inject SignInManager<ApplicationUser> SignInManager  @inject UserManager<ApplicationUser> UserManager |

## 数据库迁移

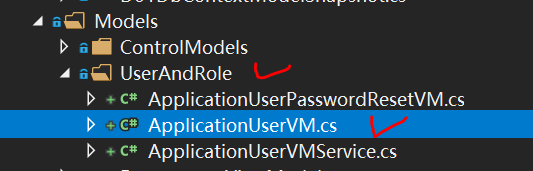
1. 在程序包控制台管理器中运行以下迁移命令，迁移数据库

Applicationxx为迁移名，xx为根据迁移次数变化的编号，自定义

|  |
| --- |
| add-migration Applicationxx -context ApplicationDbContext  update-database -context ApplicationDbContext |

## 添加用户管理视图模型

1. 在Models文件夹中添加UserAndRole文件夹，并在其下添加ApplicationUserVM类



1. ApplicationUserVM类代码如下

|  |
| --- |
| public class ApplicationUserVM  {  public string Id { get; set; }  public string OrderNumber { get; set; }  [Display(Name = "昵称")]  [StringLength(100, ErrorMessage = "昵称超过了100字符。")]  [Required(ErrorMessage = "用户名称不能为空值。")]  public string Name { get; set; }  [Required(ErrorMessage = "用户名不能为空值。")]  [Display(Name = "用户名")]  [StringLength(100, ErrorMessage = "你输入的数据超出限制100个字符的长度。")]  public string UserName { get; set; }  [Display(Name = "登录邮件")]  [Required(ErrorMessage = "电子邮件数据是必须的。")]  [EmailAddress(ErrorMessage = "请输入合法的电子邮件地址。")]  public string Email { get; set; }  [Display(Name = "移动电话")]  [RegularExpression(@"((^13[0-9]{1}[0-9]{8}|^15[0-9]{1}[0-9]{8}|^14[0-9]{1}[0-9]{8}|^16[0-9]{1}[0-9]{8}|^17[0-9]{1}[0-9]{8}|^18[0-9]{1}[0-9]{8}|^19[0-9]{1}[0-9]{8})|^((\d{7,8})|(\d{4}|\d{3})-(\d{7,8})|(\d{4}|\d{3})-(\d{7,8})-(\d{4}|\d{3}|\d{2}|\d{1})|(\d{7,8})-(\d{4}|\d{3}|\d{2}|\d{1}))$)", ErrorMessage = "电话号码数据不合规！"),  Required(ErrorMessage = "移动电话号码数据是必须的。"),  MaxLength(11, ErrorMessage = "电话号码超过11位数！"),  MinLength(11, ErrorMessage = "电话号码长度不足11位数！")]  public string MobileNumber { get; set; }  [Display(Name = "密码")]  [DataType(DataType.Password)]  [Required(ErrorMessage = "密码是必须的。")]  [RegularExpression(@"((^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[$@$!%\*#?&])[A-Za-z\d$@$!%\*#?&]{8,})$)", ErrorMessage = "密码至少8个字符，至少1个小写字母，1个大写字母，1个数字和1个特殊字符！")]  public string Password { get; set; }  [Display(Name = "确认密码")]  [DataType(DataType.Password)]  [Compare("Password", ErrorMessage = "密码必须一致")]  public string PasswordComfirm { get; set; }    public ApplicationUserVM()  {  this.Id = Guid.NewGuid().ToString();  }  } |

## 添加用户管理业务类

1. 在Models文件夹中添加UserAndRole文件夹，并在其下添加ApplicationUserPasswordResetVM类

|  |
| --- |
|  |

1. ApplicationUserPasswordResetVM类代码如下

|  |
| --- |
| public class ApplicationUserPasswordResetVM  {  public Guid UserId { get; set; }    [Display(Name = "新密码")]  [DataType(DataType.Password)]  [Required(ErrorMessage = "密码是必须的。")]  [RegularExpression(@"((^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[$@$!%\*#?&])[A-Za-z\d$@$!%\*#?&]{8,})$)", ErrorMessage = "密码至少8个字符，至少1个小写字母，1个大写字母，1个数字和1个特殊字符！")]  public string Password { get; set; }  [Display(Name = "重复密码")]  [DataType(DataType.Password)]  [Compare("Password", ErrorMessage = "密码必须一致")]  public string PasswordComfirm { get; set; }  public string Code { get; set; }  public string ResetStatus { get; set; } } |

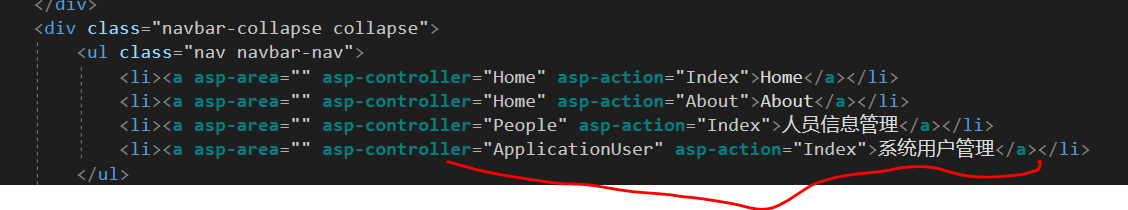
1. 在Models文件夹中添加UserAndRole文件夹，并在其下添加ApplicationUserVMService类

ApplicationUserVMService类通过构造函数引入自定义身份验证，代码如下

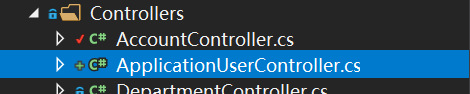
|  |
| --- |
| public class ApplicationUserVMService  {  private readonly UserManager<ApplicationUser> \_userManager;  public ApplicationUserVMService(UserManager<ApplicationUser> userManager )  {  \_userManager = userManager;  }  /// <summary>  /// 返回一个新的视图模型  /// </summary>  /// <returns></returns>  public ApplicationUserVM GetVM()  {  var boVM = new ApplicationUserVM();  return boVM;  }  /// <summary>  /// 根据系统用户对象 Id 返回一个系统用户视图模型对象  /// </summary>  /// <param name="id"></param>  /// <returns></returns>  public async Task<ApplicationUserVM> GetVM(Guid id)  {  var boVM = new ApplicationUserVM();  var bo = await \_userManager.FindByIdAsync(id.ToString());  if (bo != null)  {    \_BoMapToVM(bo, boVM);  }  return boVM;  }  /// <summary>  /// 根据系统用户对象返回一个系统用户视图模型对象  /// </summary>  /// <param name="bo"></param>  /// <returns></returns>  public ApplicationUserVM GetVM(ApplicationUser bo)  {  var boVM = new ApplicationUserVM();  \_BoMapToVM(bo, boVM);  return boVM;  }  /// <summary>  /// 返回全部业务对象对应的视图模型  /// </summary>  /// <param name="boService"></param>  /// <returns></returns>  public List<ApplicationUserVM> GetboVMCollection()  {  var boCollection = \_userManager.Users.AsQueryable().OrderBy(x => x.UserName);  var boVMCollection = new List<ApplicationUserVM>();  var counter = 0;  foreach (var user in boCollection)  {  var boVM = GetVM(user);  boVM.OrderNumber = (++counter).ToString();  boVMCollection.Add(boVM);  }  return boVMCollection;  }  /// <summary>  /// 检查用户名是否重复  /// </summary>  /// <param name="username"></param>  /// <returns></returns>  public async Task<bool> IsUniquelyForUserName(string username)  {  var user = await \_userManager.FindByNameAsync(username);  return (user == null) ? true : false;  }  /// <summary>  /// 保存系统用户数据  /// </summary>  /// <param name="boVM"></param>  /// <returns></returns>  public async Task<bool> SaveBo(ApplicationUserVM boVM)  {  var bo = await \_userManager.FindByIdAsync(boVM.Id.ToString());  if (bo == null)  {  #region 新建用户  bo = new ApplicationUser(boVM.UserName);  \_VMMapToBo(bo, boVM);  var result = await \_userManager.CreateAsync(bo, boVM.Password);  if (result.Succeeded)  {  return true;  }  else  return false;  #endregion  }  else  {  #region 老用户  \_VMMapToBo(bo, boVM);  var result = await \_userManager.UpdateAsync(bo);  if (result.Succeeded)  {  return true;  }  else  {  return false;  }  #endregion  }  }  /// <summary>  /// 删除系统用户  /// </summary>  /// <param name="id"></param>  /// <returns></returns>  public async Task<bool> Delete(Guid id)  {  var bo = await \_userManager.FindByIdAsync(id.ToString());  bool status = true;  try  {  await \_userManager.DeleteAsync(bo);  }  catch  {  status= false;    }  return status;  }  //重置用户密码  public async Task<bool> ResetApplicationUserPassword(ApplicationUserPasswordResetVM resetVM)  {  var user = await \_userManager.FindByIdAsync(resetVM.UserId.ToString());  if (user != null)  {  string tokenString = await \_userManager.GeneratePasswordResetTokenAsync(user);  var updateResult = await \_userManager.ResetPasswordAsync(user, tokenString, resetVM.Password);  if (updateResult.Succeeded)  {  resetVM.ResetStatus = "重置密码成功";  }  else  {  resetVM.ResetStatus = "重置密码失败";  }  }  return true;  }    #region 业务对象和视图模型映射处理  /// <summary>  /// 业务对象的属性转换为视图对象的属性  /// </summary>  /// <param name="bo"></param>  private void \_BoMapToVM(ApplicationUser bo, ApplicationUserVM boVM)  {  boVM.Id = bo.Id;  boVM.UserName = bo.UserName;  boVM.MobileNumber = bo.MobileNumber;  boVM.Email = bo.Email;  boVM.Name = bo.ChineseFullName;  }  /// <summary>  /// 视图对象的属性转换为业务对象的属性  /// </summary>  /// <param name="bo"></param>  private void \_VMMapToBo(ApplicationUser bo, ApplicationUserVM boVM)  {  bo.ChineseFullName = boVM.Name;  bo.UserName = boVM.UserName;  bo.Email = boVM.Email;  bo.MobileNumber = boVM.MobileNumber;  }  #endregion  } |

## 在模板视图中添加应用链接

本演练仅通过操作系统用户管理链接管理用户



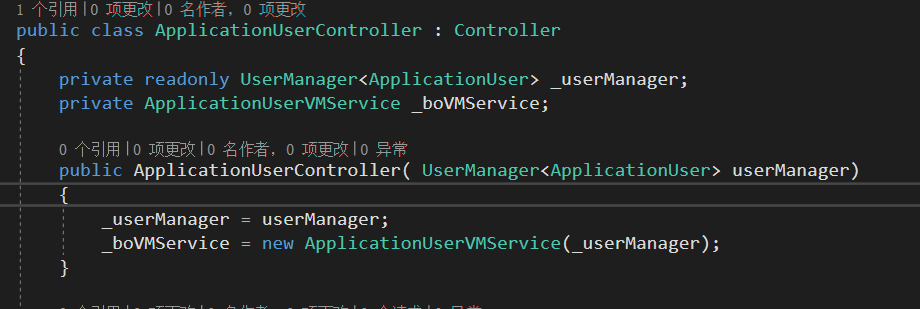
## 添加用户管理控制器ApplicationUserController



1. 在ApplicationUserController通过构造函数引入身份验证服务和用户管理服务

|  |
| --- |
| private readonly UserManager<ApplicationUser> \_userManager;  private ApplicationUserVMService \_boVMService;  public ApplicationUserController( UserManager<ApplicationUser> userManager)  {  \_userManager = userManager;  \_boVMService = new ApplicationUserVMService(\_userManager);  } |

效果如下



### 查看用户列表

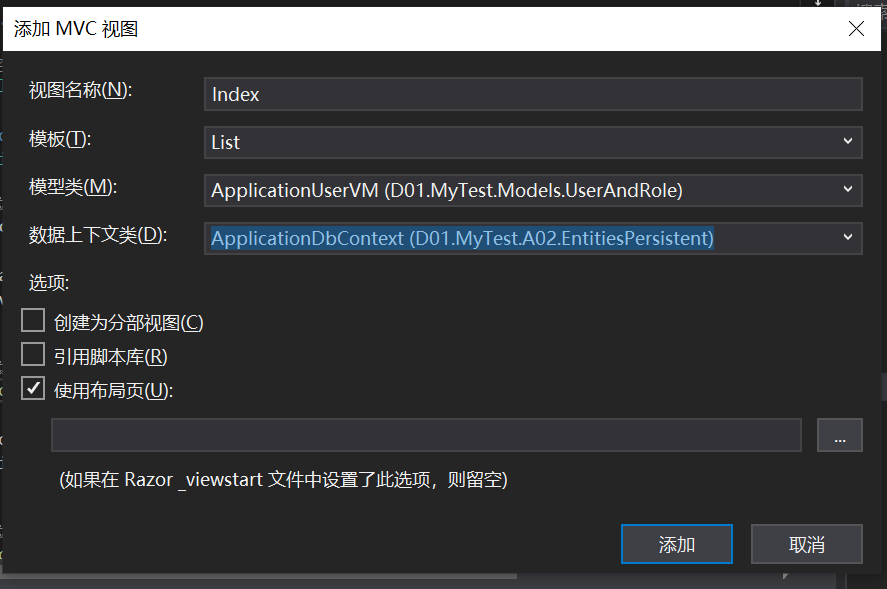
#### 行为

1. 在ApplicationUserController中添加Index行为

|  |
| --- |
| public IActionResult Index()  {  var boVMCollection = \_boVMService.GetboVMCollection();  return View(boVMCollection);  } |

#### 视图

1. 在ApplicationUserController /Index行为上右键单击添加视图



并根据需要去掉一些不需要的显示项目

运行效果如下：



### 新增用户

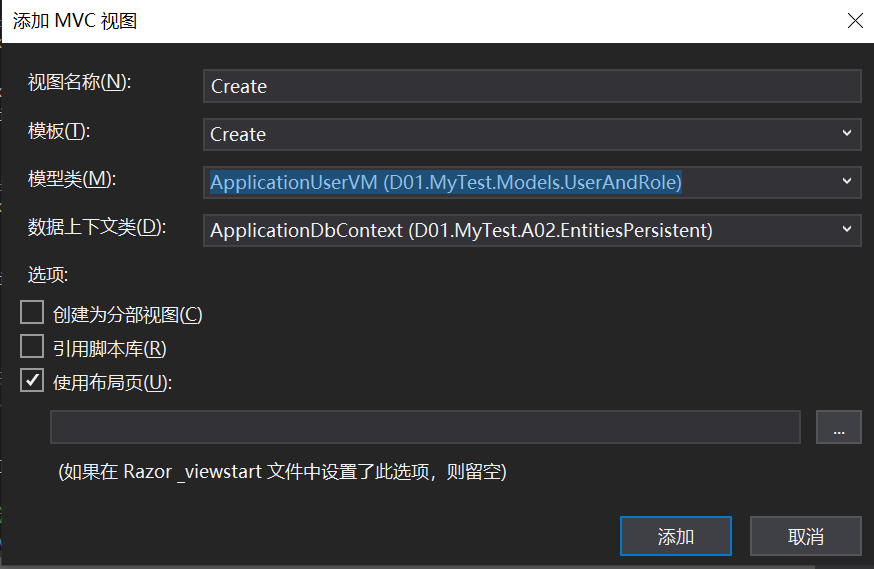
#### 行为

1. 在ApplicationUserController中添加Create行为 HttpGet和HttpHopst

|  |
| --- |
| public IActionResult Create()  {  var boVM = \_boVMService.GetVM();  return View(boVM);  }    [HttpPost]  public async Task<IActionResult> Create(ApplicationUserVM boVM)  {    if (ModelState.IsValid)  {  // 检验用户名的唯一性  var isUniquelyForUserName = await \_boVMService.IsUniquelyForUserName(boVM.UserName);  if (!isUniquelyForUserName)  {  ModelState.AddModelError("UserName", "用户名重复，请重新输入一个。");  }  else  {  // 保存数据  var x = await \_boVMService.SaveBo(boVM);  if (x)  return RedirectToAction("index");  }  }  return View(boVM);  } |

#### 视图

添加新增视图



隐藏或删除一些不需要填写的属性

如：<input type="hidden" asp-for="OrderNumber" />

运行新增后，可以增加用户，并返回用户列表

### 删除用户/编辑用户/重置用户密码

以上操作由同学通过引用完成