

开发项目介绍

Develop Project Introduction



议题

- ◉ 技术/框架/工具
- ◉ 系统架构
- ◉ 项目结构
- ◉ 网站结构
- ◉ 系统管理模块
- ◉ 编码规范
- ◉ 开发示例
- ◉ 网站发布



技术/框架/工具

- ◉ 前端：
 - ◉ 响应式布局、Jquery、[Bootstrap V2](#)
- ◉ 后端：
 - ◉ 思想：[DDD（领域驱动设计）](#)、[TDD（测试驱动设计）](#)、[DI/IOC（依赖注入/控制反转）](#)、[面向接口编辑](#)
 - ◉ 技术、框架：[Asp.net MVC5](#)、[Entity Framework 6](#)、[EF Code First](#)、.Net Framework 4.5、xUnit、Autofac
 - ◉ 工具：Git、VS2013、Sql Server、Nuget包、T4模板



系统架构介绍

◉ EF Code First

不需要预先设计数据表，直接进入代码开发，利用EF数据迁移功能，自动生成需要的数据定义脚本。

◉ DTO 对象

优点：DTO层的作用是为了隔离Domain Model：让DoMain Model的改动不会直接影响到UI；保持Domain Model的安全,不暴露业务逻辑。安全、效率、跨平台。

缺点：太麻烦了。但有了对象影射工具的帮忙，省了不少事。

◉ 接口编程

使用接口编程，实现松散耦合的系统，便于以后升级，扩展。

采用基于接口编程的项目，业务逻辑清晰，代码易懂，方便扩展，可维护性强。

设计模式：依赖倒转的设计原则(DIP)----高层模块不应该依赖于底层模块。



系统架构介绍

◉ 模块化开发

模块化是以分治法为依据。简单说就是把软件整体划分，划分后的块组成了软件。一句话：解决软件的复杂性问题，或说降低软件的复杂性。不至于随着变大而不可控而失败，使其可控，可维护，可扩展。

◉ 依赖注入

解除接口与实现的依赖关系，降低代码的耦合性；

由IoC容器来确定使用哪个具体的实现，并控制对象的生命周期；

使代码更易于测试。

◉ 资源化、国际化

提示信息等集中到资源文件中，而不是遍布在代码的每一处。可为多语言做准备；

日期内容采用UTC时间进行保存，系统显示时采用当地时间显示。

单元测试

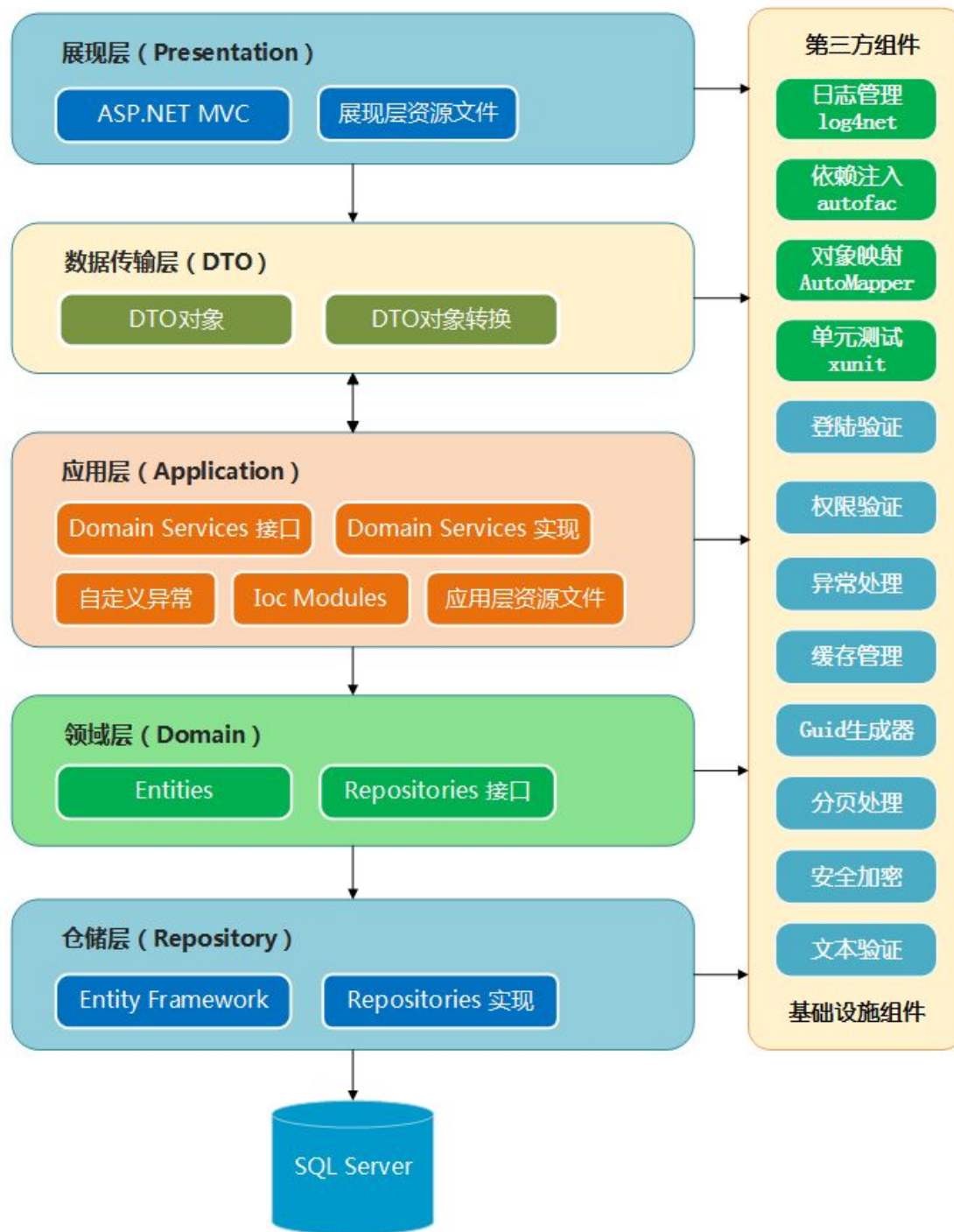
实现高质量的代码，减少bug出现的机会

单元测试不仅能保证项目进度还能优化你的设计

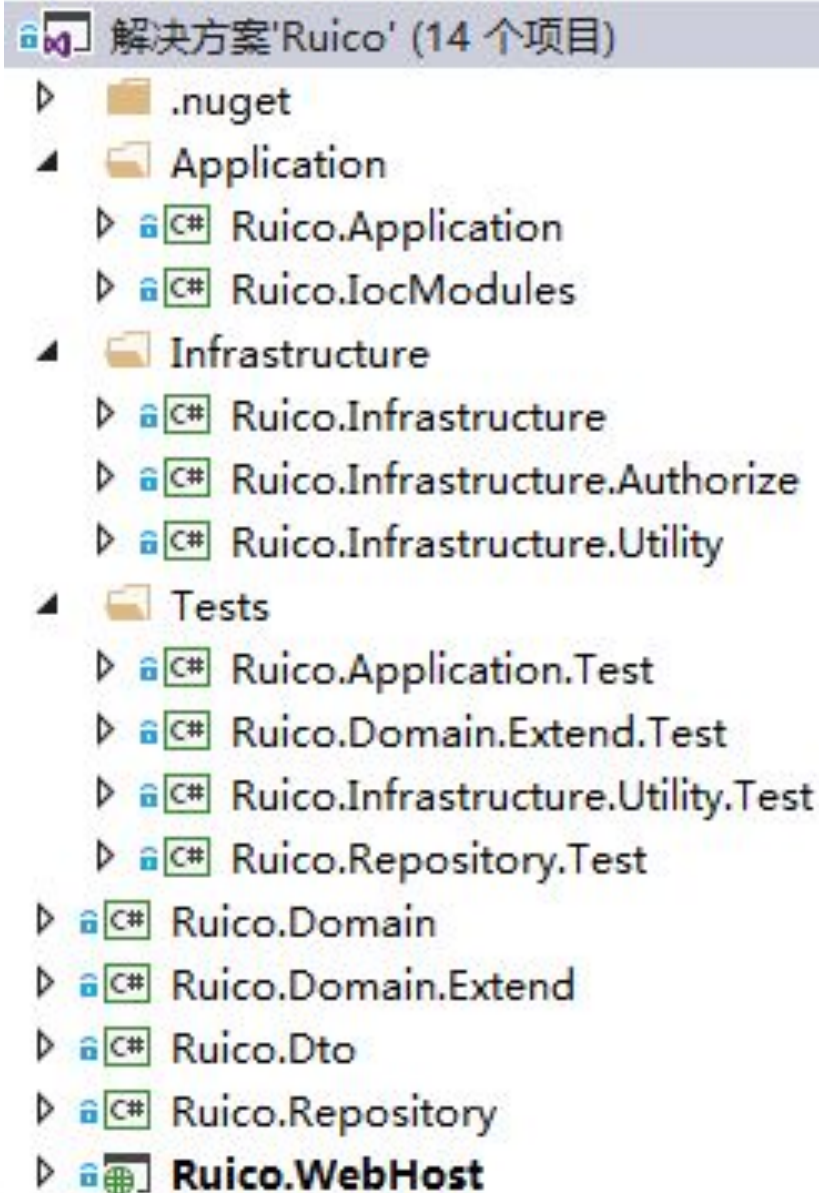
使代码可以放心修改和重构



系统架构图



项目结构



网站结构



顶部模块可以在系统管理的模块管理中设置

拥有权限的菜单，所属的模块才会展示出来

这是一个没有左边菜单的页面



网站结构



拥有权限的菜单，才会在左边展示出来

这是一个有左边菜单的页面



角色管理

Ruico System

角色和用户2

用户管理

角色管理

模块和菜单2

欢迎你admin! 今天是2015年08月09日

我的权限

退出

我的工作台

市场发展

物流管理

系统管理

角色组

[首页](#) > [角色管理](#) > [角色组](#)

查询

新增角色组

用户权限生效

角色组名称	描述	排序	添加时间	操作
管理员		0	2015/07/30 23:08:05	角色 用户 编辑 删除
业务管理员		0	2015/08/02 15:04:03	角色 用户 编辑 删除

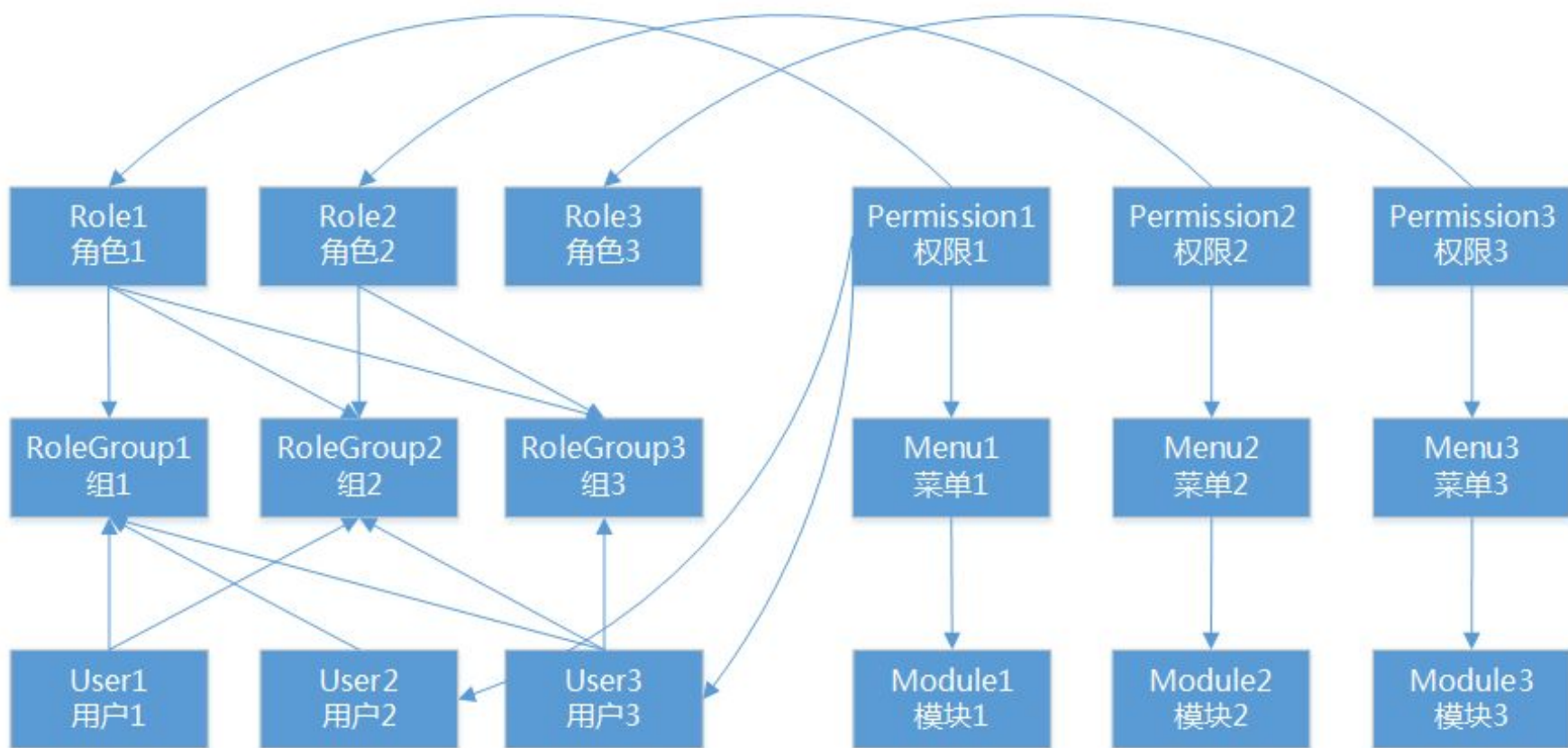
1

2015 © Ruico.





系统管理模块-概要设计

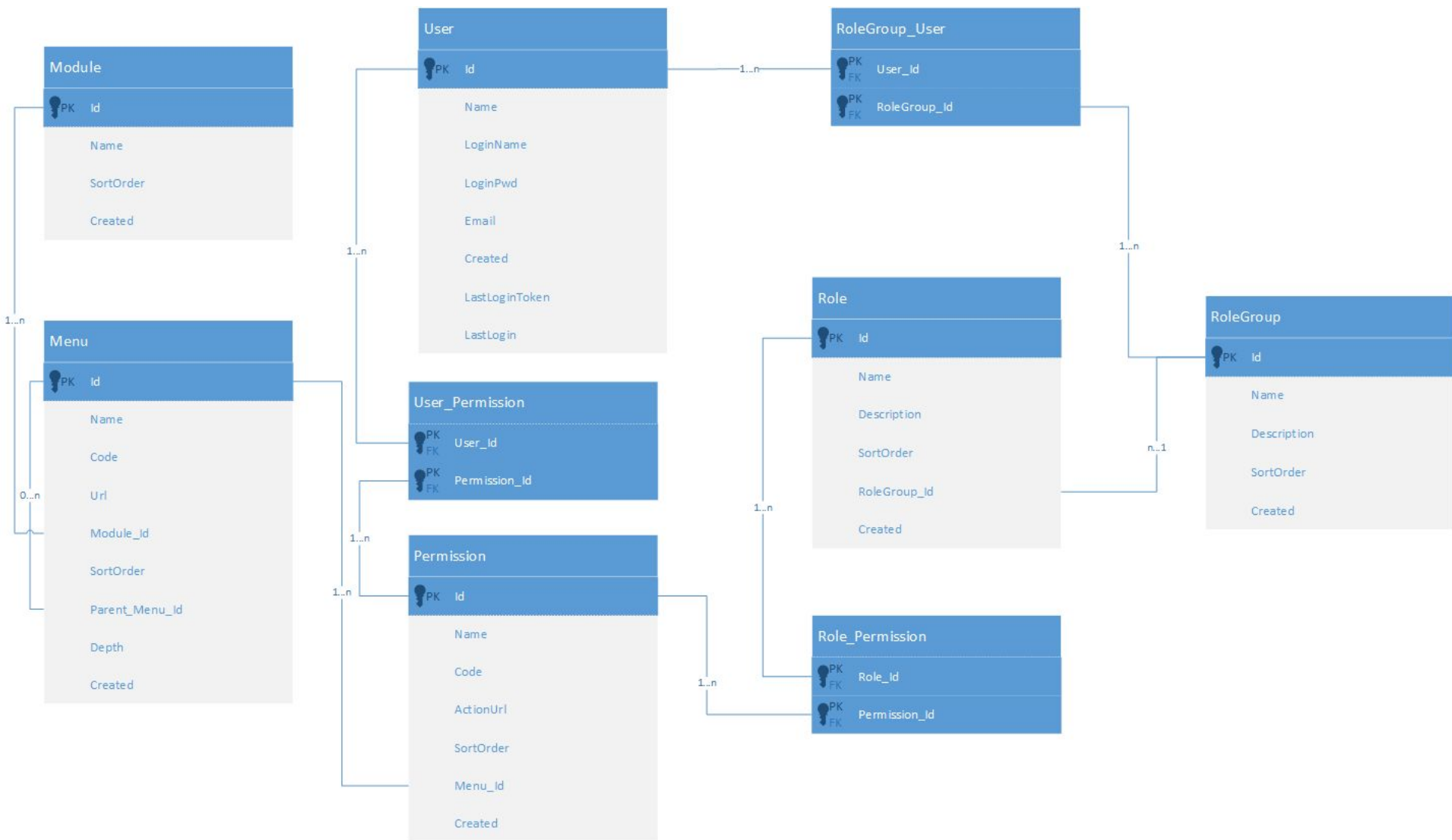


通过用户1 (USER1) 的组找出所有角色就能找到用户1对应的权限
通过一个组就能找到所有权限和所有人
给另一个用户分配一个用户1一样的权限值需要把另一个用户放到相同组中

通过用户的权限可得到客户拥有的菜单
每个模块有单独的菜单
菜单可以拥有一个父级菜单，实现多级菜单

一个组对应多个用户 一个角色对应多个权限
一个组对应多个角色 一个用户可以直接对应多个权限

系统管理模块-详细设计



编码规范

。 字母大小写约定

1. Pascal风格

包含一到多个单词，每一个单词第一个字母大写，其余字母均小写。例如：
`HelloWorld`、`SetName`等。

2. Camel风格（骆驼命名法）

包含一到多个单词，第一个单词首字母小写，其余单词首字母大写。例如：`name`、`productId`等。

。 标识符的大小写规则

除了参数与变量外，所有命名空间名称、类、函数、接口、属性等名称的命名，使用
`Pascal` 风格。

参数与变量的命名，使用`Camel`风格。



编码规范

- ◉ 接口命名以字母 **I** 为前缀，如：**ISomeInterface**
- ◉ 衍生类的末尾使用基类名称，如：从 **Exception** 继承的类型以 **Exception** 结尾
- ◉ 使用英文而不是拼音，如：**User** 比 **YongHu** 顺眼
- ◉ 缩进和间隔：缩进用**TAB**，不用 **SPACES**。
- ◉ 多使用**#region**和**#endregion**代码块
- ◉ 适当的增加空行，来增加代码的可读性
- ◉ 避免使用大文件。如果一个文件里的代码超过**300~400**行，必须考虑将代码分开到不同类中。当然模板生成类与逻辑层类除外。
- ◉ 避免写太长的方法。一个典型的方法代码在**1~25**行之间。如果一个方法发代码超过**25**行，应该考虑将其分解为不同的方法。
- ◉ 尽量按**Resharper**的检查建议，修改你的代码，除非修改后影响代码理解和阅读。



开发示例-角色管理

- ◉ 系统管理模块中的角色管理
- ◉ 一个组对应多个角色
- ◉ 一个角色对应多个权限



开发示例-角色管理-领域实体

```
7 namespace Ruico.Domain.UserSystemModule.Entities
8 {
9     [Table("auth.Role")]
10    20 个引用 | itbud, 1 天之前 | 1 个更改
11    public class Role : EntityBase
12    {
13        84 个引用 | itbud, 1 天之前 | 1 个更改
14        public override Guid Id { get; set; }
15
16        [MaxLength(50)]
17        12 个引用 | itbud, 1 天之前 | 1 个更改
18        public string Name { get; set; }
19
20        [MaxLength(150)]
21        3 个引用 | itbud, 1 天之前 | 1 个更改
22        public string Description { get; set; }
23
24        4 个引用 | itbud, 1 天之前 | 1 个更改
25        public int SortOrder { get; set; }
26
27        2 个引用 | itbud, 1 天之前 | 1 个更改
28        public DateTime Created { get; set; }
29
30        5 个引用 | itbud, 1 天之前 | 1 个更改
31        public virtual RoleGroup RoleGroup { get; set; }
32
33        6 个引用 | itbud, 1 天之前 | 1 个更改
34        public virtual ICollection<Permission> Permissions { get; set; }
35    }
36 }
```



开发示例-角色管理-DbContext

```
6 namespace Ruico.Repository
7 {
8     10 个引用 | itbud, 2 天之前 | 1 个更改
9     public class RuicoUnitOfWork : DbContext, IRuicoUnitOfWork
10     {
11         5 个引用 | itbud, 2 天之前 | 1 个更改
12         public RuicoUnitOfWork()
13             : base("RuicoContext")
14         {
15             Initializer.DbInitializer.Initialize();
16         }
17
18         1 个引用 | itbud, 2 天之前 | 1 个更改
19         public virtual DbSet<RoleGroup> RoleGroups { get; set; }
20
21         5 个引用 | itbud, 2 天之前 | 1 个更改
22         public virtual DbSet<Role> Roles { get; set; }
23
24         2 个引用 | itbud, 2 天之前 | 1 个更改
25         public virtual DbSet<User> Users { get; set; }
26
27         1 个引用 | itbud, 2 天之前 | 1 个更改
28         public virtual DbSet<Menu> Menus { get; set; }
29
30         0 个引用 | itbud, 2 天之前 | 1 个更改
31         public virtual DbSet<Permission> Permissions { get; set; }
32
33         0 个引用 | itbud, 2 天之前 | 1 个更改
34         public virtual DbSet<Module> Modules { get; set; }
35
36         0 个引用 | itbud, 2 天之前 | 1 个更改
37         protected override void OnModelCreating(DbModelBuilder mb)
38         {
39             base.OnModelCreating(mb);
40
41             this.OnModelCreatingCommon(mb);
42
43             this.OnModelCreatingUserSystem(mb);
44         }
45
46         1 个引用 | itbud, 2 天之前 | 1 个更改
47         private void OnModelCreatingCommon(DbModelBuilder mb)
48         {
49             mb.Entity<Menu>().HasRequired(x => x.Module);
50         }
51
52         1 个引用 | itbud, 2 天之前 | 1 个更改
53         private void OnModelCreatingUserSystem(DbModelBuilder mb)
54         {
55             mb.Entity<Role>().HasRequired(x => x.RoleGroup).WithMany(x => x.Roles);
56
57             mb.Entity<Role>().HasMany(x => x.Permissions).WithMany(x => x.Roles).Map(x => x.MapLeftKey(
58                 "Role_Id").MapRightKey("Permission_Id").ToTable("Role_Permission", "auth"));
59         }
60     }
```



开发示例-角色管理-Migration

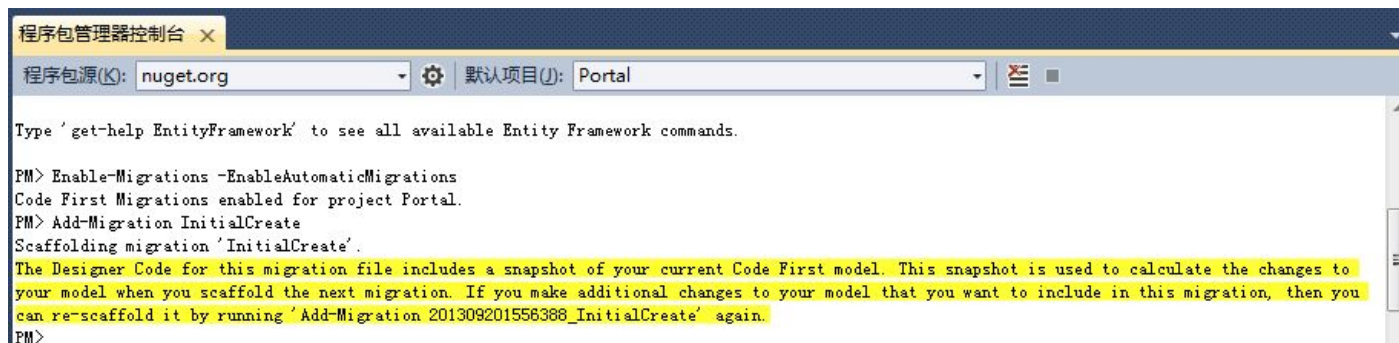
- ◉ **Enable-Migrations** 启用迁移
- ◉ **Add-Migration** 为挂起的Model变化添加迁移脚本
- ◉ **Update-Database** 将挂起的迁移更新到数据库
- ◉ **Get-Migrations** 获取已经应用的迁移



开发示例-角色管理-Migration

- 在程序包管理器控制台，执行语句：

PM> Add-Migration AddRole [示例中为InitialCreate]



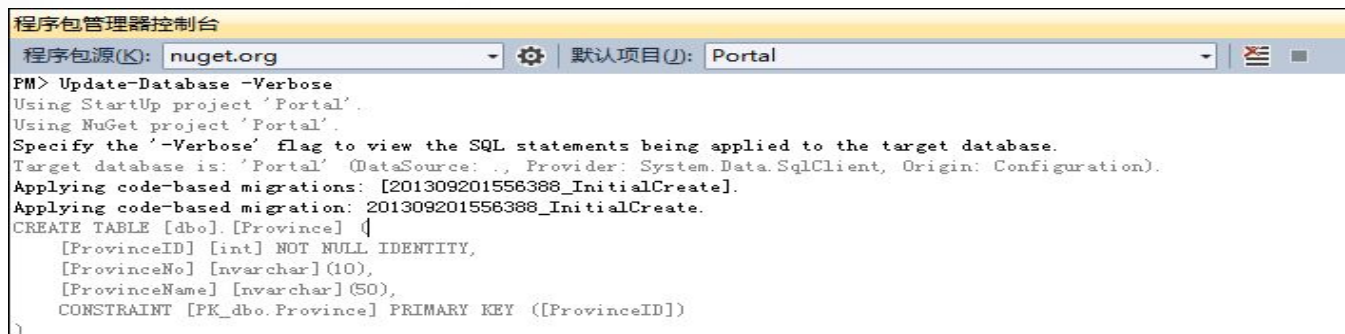
```
程序包管理器控制台 x
程序包源(K): nuget.org 默认项目(J): Portal

Type 'get-help EntityFramework' to see all available Entity Framework commands.

PM> Enable-Migrations -EnableAutomaticMigrations
Code First Migrations enabled for project Portal.
PM> Add-Migration InitialCreate
Scaffolding migration 'InitialCreate'.
The Designer Code for this migration file includes a snapshot of your current Code First model. This snapshot is used to calculate the changes to your model when you scaffold the next migration. If you make additional changes to your model that you want to include in this migration, then you can re-scaffold it by running 'Add-Migration 201309201556388_InitialCreate' again.
PM>
```

- 执行成功后，在Migrations文件夹中新增类文件：
201508111556388_AddRole.cs [示例中为201309201556388_InitialCreate.cs]
- 在程序包管理器控制台，执行语句：

PM> Update-Database -Verbose



```
程序包管理器控制台
程序包源(K): nuget.org 默认项目(J): Portal

PM> Update-Database -Verbose
Using StartUp project 'Portal'.
Using NuGet project 'Portal'.
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Target database is: 'Portal' (DataSource: ., Provider: System.Data.SqlClient, Origin: Configuration).
Applying code-based migrations: [201309201556388_InitialCreate].
Applying code-based migration: 201309201556388_InitialCreate.
CREATE TABLE [dbo].[Province] (
    [ProvinceID] [int] NOT NULL IDENTITY,
    [ProvinceNo] [nvarchar](10),
    [ProvinceName] [nvarchar](50),
    CONSTRAINT [FK_dbo.Province] PRIMARY KEY ([ProvinceID])
)
```



开发示例-角色管理-仓储接口

```
6 namespace Ruico.Domain.UserSystemModule.Repositories
7 {
8     4 个引用 | itbud, 2 天之前 | 1 个更改
9     public interface IRoleRepository : IRepository<Role>
10    {
11        2 个引用 | itbud, 2 天之前 | 1 个更改
12        IPagedList<Role> FindBy(Guid roleGroupId, string name, int pageNumber, int pageSize);
13
14        3 个引用 | itbud, 2 天之前 | 1 个更改
15        bool Exists(Role item);
16    }
17 }
```

```
8 public interface IRepository<TEntity> where TEntity : class
9 {
10     27 个引用 | itbud, 2 天之前 | 1 个更改
11     IUnitOfWork UnitOfWork { get; }
12     40 个引用 | itbud, 2 天之前 | 1 个更改
13     TEntity Get(object key);
14     5 个引用 | itbud, 2 天之前 | 1 个更改
15     IEnumerable<TEntity> FindAll();
16     7 个引用 | itbud, 2 天之前 | 1 个更改
17     void Add(TEntity item);
18     8 个引用 | itbud, 2 天之前 | 1 个更改
19     void Remove(TEntity item);
20     3 个引用 | itbud, 2 天之前 | 1 个更改
21     TEntity Find(Func<TEntity, bool> acquire);
22     4 个引用 | itbud, 2 天之前 | 1 个更改
23     IQueryable<TEntity> Collection { get; }
24 }
25 }
```



开发示例-角色管理-仓储实现

```
11 public class RoleRepository : SpecificRepositoryBase<Role>, IRoleRepository
12 {
13     0 个引用 | itbud, 2 天之前 | 1 个更改
14     public RoleRepository(IRuicoUnitOfWork unitOfWork) : base(unitOfWork)
15     {
16     }
17
18     2 个引用 | itbud, 2 天之前 | 1 个更改
19     public IPagedList<Role> FindBy(Guid roleGroupId, string name, int pageNumber, int pageSize)
20     {
21         IQueryable<Role> entities = Table;
22
23         if (roleGroupId != Guid.Empty)
24         {
25             entities =
26                 entities.Where(x => x.RoleGroup.Id == roleGroupId);
27         }
28
29         if (name.NotNullOrBlank())
30         {
31             entities =
32                 entities.Where(x => x.Name.Contains(name));
33         }
34
35         var totalCountQuery = entities.FutureCount();
36         var resultQuery = entities
37             .OrderBy(x => x.SortOrder)
38             .Skip((pageNumber - 1) * pageSize)
39             .Take(pageSize)
40             .Future();
41
42         var totalCount = totalCountQuery.Value;
43         var result = resultQuery.ToList();
44
45         return new StaticPagedList<Role>(
46             result,
47             pageNumber,
48             pageSize,
49             totalCount);
50     }
51
52     3 个引用 | itbud, 2 天之前 | 1 个更改
53     public bool Exists(Role item)
54     {
55         IQueryable<Role> entities = Table;
56         entities = entities.Where(x => x.RoleGroup.Id == item.RoleGroup.Id && x.Name == item.Name);
57         if (item.Id != Guid.Empty)
58         {
59             entities = entities.Where(x => x.Id != item.Id);
60         }
61         return entities.Any();
62     }
63 }
```



开发示例-角色管理-应用层接口

```
6 namespace Ruico.Application.UserSystemModule
7 {
8     public interface IRoleService
9     {
10         RoleDTO Add(RoleDTO roleDTO);
11
12         void Update(RoleDTO roleDTO);
13
14         void Remove(Guid id);
15
16         List<RoleDTO> FindAll();
17
18         RoleDTO FindBy(Guid id);
19
20         IPagedList<RoleDTO> FindBy(Guid roleGroupId, string name, int pageNumber, int pageSize);
21
22         void UpdateRolePermission(Guid id, List<Guid> permissions);
23
24         List<PermissionDTO> GetRolePermission(Guid id);
25     }
26 }
27
```



开发示例-角色管理-DTO对象

```
3 namespace Ruico.Dto.UserSystem
4 {
    20 个引用 | itbud, 2 天之前 | 1 个更改
5     public class RoleDTO
6     {
        2 个引用 | itbud, 2 天之前 | 1 个更改
7         public Guid Id { get; set; }
8
        0 个引用 | itbud, 2 天之前 | 1 个更改
9         public string Name { get; set; }
10
        0 个引用 | itbud, 2 天之前 | 1 个更改
11        public string Description { get; set; }
12
        0 个引用 | itbud, 2 天之前 | 1 个更改
13        public int SortOrder { get; set; }
14
        0 个引用 | itbud, 2 天之前 | 1 个更改
15        public DateTime Created { get; set; }
16
        3 个引用 | itbud, 2 天之前 | 1 个更改
17        public Guid RoleGroupId { get; set; }
18    }
19 }
```



开发示例-角色管理-应用层实现

```
17 namespace Ruico.Application.UserSystemModule.Imp
18 {
19     public class RoleService : IRoleService
20     {
21         IRepository _Repository;
22         IRoleGroupRepository _RoleGroupRepository;
23         IPermissionRepository _PermissionRepository;
24
25         #region Constructors
26
27         public RoleService(IRepository repository, IRoleGroupRepository _roleGroupRepository,
28             IPermissionRepository permissionRepository)
29         {
30             if (repository == null)
31                 throw new ArgumentNullException("repository");
32
33             _Repository = repository;
34             _RoleGroupRepository = _roleGroupRepository;
35             _PermissionRepository = permissionRepository;
36         }
37
38         #endregion
39
40         public RoleDTO Add(RoleDTO roleDTO)
41         {
42             var role = roleDTO.ToModel();
43             role.Id = IdentityGenerator.NewSequentialGuid();
44             role.Created = DateTime.UtcNow;
45
46             var group = _RoleGroupRepository.Get(roleDTO.RoleGroupId);
47             if (group == null)
48             {
```



开发示例-角色管理-DTO对象转换

```
1 using AutoMapper;
2 using Ruico.Domain.UserSystemModule.Entities;
3
4 namespace Ruico.Dto.UserSystem.Converters
5 {
6     public static class RoleConverters
7     {
8         static RoleConverters()
9         {
10             Mapper.CreateMap<Role, RoleDTO>();
11
12             Mapper.CreateMap<RoleDTO, Role>();
13         }
14
15         public static Role ToModel(this RoleDTO dto)
16         {
17             return Mapper.Map<Role>(dto);
18         }
19
20         public static RoleDTO ToDto(this Role model)
21         {
22             return Mapper.Map<RoleDTO>(model);
23         }
24     }
25 }
26
```



开发示例-角色管理-IocModule

```
8 namespace Ruico.IocModules
9 {
10     1 个引用 | itbud, 2 天之前 | 1 个更改
    public class UserSystemIocModule : Module
11     {
12         1 个引用 | itbud, 2 天之前 | 1 个更改
        protected override void Load(ContainerBuilder builder)
13         {
14             builder.RegisterType<ModuleRepository>().As<IModuleRepository>().InstancePerRequest();
15             builder.RegisterType<ModuleService>().As<IModuleService>();
16
17             builder.RegisterType<MenuRepository>().As<IMenuRepository>().InstancePerRequest();
18             builder.RegisterType<PermissionRepository>().As<IPermissionRepository>().InstancePerRequest();
19             builder.RegisterType<MenuService>().As<IMenuService>();
20
21             builder.RegisterType<RoleRepository>().As<IRoleRepository>().InstancePerRequest();
22             builder.RegisterType<RoleService>().As<IRoleService>();
23
24             builder.RegisterType<RoleGroupRepository>().As<IRoleGroupRepository>().InstancePerRequest();
25             builder.RegisterType<RoleGroupService>().As<IRoleGroupService>();
26
27             builder.RegisterType<UserRepository>().As<IUserRepository>().InstancePerRequest();
28             builder.RegisterType<UserService>().As<IUserService>();
29
30             builder.RegisterType<AuthService>().As<IAuthService>();
31         }
32     }
33 }
```



开发示例-角色管理-Controller

```
11 namespace Ruico.WebHost.Areas.Core.System.Controllers
12 {
13     [AuthorizeFilter]
14     public class RoleController : ControllerBase
15     {
16         IRoleService _roleService;
17
18         IRoleGroupService _roleGroupService;
19
20         IMenuService _menuService;
21
22         IUserService _userService;
23
24         IModuleService _moduleService;
25
26         #region Constructor
27
28         public RoleController(IRoleService roleService, IRoleGroupService roleGroupService,
29             IMenuService menuService,
30             IUserService userService,
31             IModuleService moduleService)
32         {
33             _roleService = roleService;
34             _roleGroupService = roleGroupService;
35             _menuService = menuService;
36             _userService = userService;
37             _moduleService = moduleService;
38         }
39     }
40 }
```



开发示例-角色管理-Controller

```
65 // [Permission("UserSystem:RoleList")]  
66 0 个引用 | itbud, 2 天之前 | 1 个更改  
67 public ActionResult RoleList(Guid groupId, string name, int? page)  
68 {  
69     var list = _roleService.FindBy(groupId, name, page.HasValue ? page.Value : 1,  
70         CustomDisplayExtensions.DefaultPageSize);  
71  
72     var group = _roleGroupService.FindBy(groupId);  
73  
74     ViewBag.GroupName = group == null ? string.Empty : group.Name;  
75     ViewBag.GroupId = groupId;  
76     ViewBag.Name = name;  
77  
78     return View(list);  
79 }  
80 [HttpPost]  
81 0 个引用 | itbud, 2 天之前 | 1 个更改  
82 public ActionResult SearchRoleList(Guid groupId, string name)  
83 {  
84     return Json(new AjaxResponse  
85     {  
86         Succeeded = true,  
87         ShowMessage = false,  
88         RedirectUrl = Url.Action("RoleList", new  
89         {  
90             groupId,  
91             name  
92         })  
93     });  
94 }
```



开发示例-角色管理-Controller



```
187 public ActionResult EditRole(Guid groupId, Guid? id)
188 {
189     var role = id.HasValue ? _roleService.FindBy(id.Value) : new RoleDTO();
190
191     var group = _roleGroupService.FindBy(groupId);
192
193     ViewBag.GroupName = group == null ? string.Empty : group.Name;
194     ViewBag.GroupId = groupId;
195
196     return View(role);
197 }
198
199 [HttpPost]
200 public ActionResult EditRole(Guid groupId, RoleDTO role)
201 {
202     role.RoleGroupId = groupId;
203     if (role.Id == Guid.Empty)
204     {
205         _roleService.Add(role);
206     }
207     else
208     {
209         _roleService.Update(role);
210     }
211
212     return Json(new AjaxResponse
213     {
214         Succeeded = true,
215         ShowMessage = false,
216         RedirectUrl = Url.Action("RoleList", new { groupId = groupId })
217     });
218 }
219
220 public ActionResult RemoveRole(Guid groupId, Guid id)
221 {
222     _roleService.Remove(id);
223
224     return Json(new AjaxResponse
225     {
226         Succeeded = true,
227         ShowMessage = false,
228         RedirectUrl = Url.Action("RoleList", new { groupId = groupId })
229     }, JsonRequestBehavior.AllowGet);
230 }
```

开发示例-角色管理-View

。 略



使用T4模板自动生成资源文件

- 在**Ruico.WebHost**和**Ruico.Application**项目下各有一个**Resources**文件夹

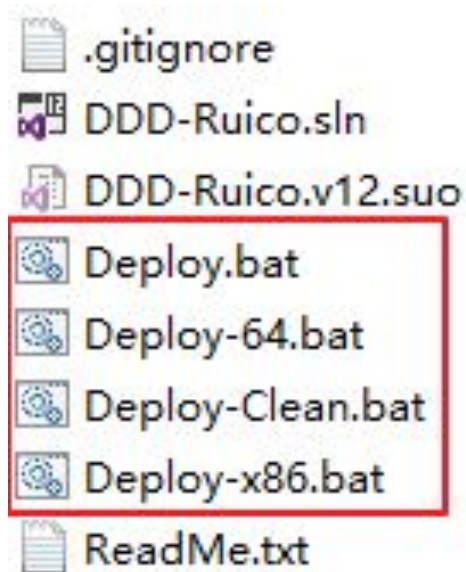
分别有**Xml**目录(存放手工编辑的xml文件)和**Generated**目录(存放自动生成的resx文件)

- 以及两个辅助文件**ResourcesGenerator.tt**和**ResourcesMultipleOutputHelper.ttinclude**
- 需要生成资源文件的内容，写在**Xml**目录下的xml文件中，每个xml文件会对应生成一个资源文件，xml文件格式参考现有的即可
- 新增或编辑xml文件后，在**ResourcesGenerator.tt**中右击选择“运行自定义工具”命令，即可在**Generated**文件夹下自动生成对应的**resx**资源文件

注：删除xml文件，自动生成资源文件时不会自动删除已有的文件，需要手工删除



网站发布



首先在服务器的源代码目录中，通过**Git**命令更新代码

在源代码的根目录，根据服务器操作系统，选择一个**Deploy**的文件执行命令将执行发布，把新生成的发布文件，复制到网站目录中

文件复制过程中，将保持网站根目录中的**Web.config**不被覆盖

