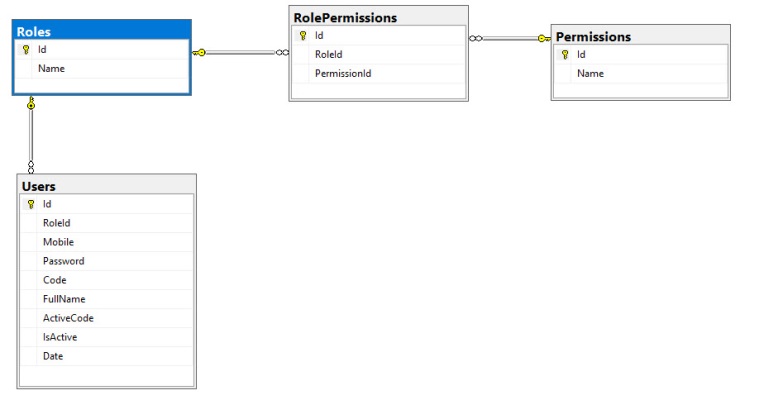
**دیاگرام جداول**



**Entity های لازم برای راه اندازی**

کلاس User

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DigiKala.DataAccessLayer.Entities

{

public class User

{

[Key]

public int Id { get; set; }

public int RoleId { get; set; }

[Display(Name = "شماره موبایل")]

[Required(ErrorMessage = "نباید بدون مقدار باشد")]

[MaxLength(11, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string Mobile { get; set; }

[Display(Name = "کلمه عبور")]

[Required(ErrorMessage = "نباید بدون مقدار باشد")]

[MaxLength(100, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string Password { get; set; }

[Display(Name = "کد ملی")]

[MaxLength(10, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string Code { get; set; }

[Display(Name = "نام و نام خانوادگی")]

[MaxLength(100, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string FullName { get; set; }

[Display(Name = "کد")]

[Required(ErrorMessage = "نباید بدون مقدار باشد")]

[MaxLength(6, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string ActiveCode { get; set; }

[Display(Name = "فعال")]

public bool IsActive { get; set; }

[Display(Name = "تاریخ عضویت")]

[Required(ErrorMessage = "نباید بدون مقدار باشد")]

[MaxLength(10, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string Date { get; set; }

public virtual Role Role { get; set; }

}

}

کلاس RolePermission

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DigiKala.DataAccessLayer.Entities

{

public class RolePermission

{

[Key]

public int Id { get; set; }

public int RoleId { get; set; }

public int PermissionId { get; set; }

[ForeignKey("RoleId")]

public virtual Role Role { get; set; }

[ForeignKey("PermissionId")]

public virtual Permission Permission { get; set; }

}

}

کلاس Role

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

namespace DigiKala.DataAccessLayer.Entities

{

public class Role

{

[Key]

public int Id { get; set; }

[Display(Name = "نام نقش")]

[Required(ErrorMessage = "نباید بدون مقدار باشد")]

[MaxLength(20, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string Name { get; set; }

public virtual ICollection<RolePermission> RolePermissions { get; set; }

public virtual ICollection<User> Users { get; set; }

}

}

کلاس Permission

using System;

using System.Collections.Generic;

using System.Text;

using System.ComponentModel.DataAnnotations;

namespace DigiKala.DataAccessLayer.Entities

{

public class Permission

{

[Key]

public int Id { get; set; }

[Display(Name = "سطح دسترسی")]

[Required(ErrorMessage = "نباید بدون مقدار باشد")]

[MaxLength(50, ErrorMessage = "مقدار {0} نباید بیش تر از {1} کاراکتر باشد")]

public string Name { get; set; }

public virtual ICollection<RolePermission> RolePermissions { get; set; }

}

}

Attribute لازم برای احراز هویت

using System;

using System.Collections.Generic;

using System.Text;

using DigiKala.Core.Interfaces;

using DigiKala.DataAccessLayer.Entities;

**using Microsoft.AspNetCore.Authorization;**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.AspNetCore.Mvc.Filters;**

namespace **DigiKala.Core.Classes**

{

public class **RoleAttribute : AuthorizeAttribute, IAuthorizationFilter**

{

int \_permissionID;

IUser \_iuser;

public RoleAttribute(int permissionID)

{

\_permissionID = permissionID;

}

public void OnAuthorization(AuthorizationFilterContext context)

{

if (context.HttpContext.User.Identity.IsAuthenticated)

{

string username = context.HttpContext.User.Identity.Name;

\_iuser = (IUser)context.HttpContext.RequestServices.GetService(typeof(IUser));

int roleID = \_iuser.GetUserRole(username);

if (!\_iuser.ExistsPermission(\_permissionID, roleID))

{

context.Result = new RedirectResult("/Accounts/Login");

}

}

else

{

context.Result = new RedirectResult("/Accounts/Login");

}

}

}

}