# Entity Framework Core Many-To-Many Relationships

Lo scenario e quello di voler attribuire ad ogni utente una o più figura aziendale.



Di seguito riporto le 3 classi che rappresentano le tabelle che vogliamo realizzare.

public partial class Users

{

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public bool? IsAdmin { get; set; }

public int? Status { get; set; }

public string Avatar { get; set; }

public string Nome { get; set; }

public string Cognome { get; set; }

public DateTime DateExpire { get; set; }

public DateTime LastAccess { get; set; }

public DateTime LastUpdatePwd { get; set; }

public bool IsValid { get; set; }

public IList<UsersMB\_Figura> UserMB\_Figure { get; set; }

}

public class MB\_Figura

{

public int MBFG\_ID { get; set; }

public string MBFG\_Code { get; set; }

public string MBFG\_Descrizione { get; set; }

public IList<UsersMB\_Figura> UserMB\_Figure { get; set; }

}

public partial class UsersMB\_Figura

{

public int UsersId { get; set; }

public Users User { get; set; }

public int MBFG\_ID { get; set; }

public MB\_Figura FiguraAziendale { get; set; }

}

La tabella UsersMB\_Figura rappresenta la relazione Many-To-Many tra Users e MB\_Figura, quindi dobbiamo procedere, nella classe context, alla definizione della chiave primaria che identifica in modo univoco ogni singola relazione :

modelBuilder.Entity<UsersMB\_Figura>()

.HasKey(uf => new { uf.UsersId, uf.MBFG\_ID });

Subito dopo, utilizzando la sintassi fluent, definisco le singole relazioni, come di seguito riportato :

modelBuilder.Entity<UsersMB\_Figura>()

.HasOne(uf => uf.User)

.WithMany(b => b.UserMB\_Figure)

.HasForeignKey(uf => uf.UsersId);

modelBuilder.Entity<UsersMB\_Figura>()

.HasOne(uf => uf.FiguraAziendale)

.WithMany(c => c.UserMB\_Figure)

.HasForeignKey(uf => uf.MBFG\_ID);

A questo punto definisco le mie entità nel modello :

public virtual DbSet<Users> Users { get; set; }

public virtual DbSet<MB\_Figura> MB\_Figura { get; set; }

public virtual DbSet<UsersMB\_Figura> UsersMB\_Figura { get; set; }

Se utilizzo **Automapper** creo le mappature che mi occorrono come ad esempio :

CreateMap<Users, jUser>()

.ForMember(dest => dest.Nominativo, opt => opt.MapFrom(src => string.Concat(src.Cognome, " ", src.Nome)))

.ForMember(dest => dest.LastAccess, opt => opt.MapFrom(src => src.LastAccess.ToString("dd/MM/yyyy hh:mm:ss")))

.ForMember(dest => dest.FigureAziendali, opt => opt.MapFrom(src => src.UserMB\_Figure != null

?

(

src.UserMB\_Figure.Select(x => new jFiguraAziendale

{ mbfg\_Id = x.FiguraAziendale.MBFG\_ID,

mbfg\_Code = x.FiguraAziendale.MBFG\_Code,

mbfg\_Descrizione = x.FiguraAziendale.MBFG\_Descrizione })

)

:

new List<jFiguraAziendale>()));

CreateMap<jUser, Users>();

Quando andrò a leggere l’entità Users avrò quindi la possibilità di sapere quante figure aziendali sono state attribuite all’utente selezionato :

public jUser GetUserById(int Id)

{

Users user = \_context.Users.Include(x => x.UserMB\_Figure).**ThenInclude**(y => y.FiguraAziendale).FirstOrDefault(x => x.Id == Id);

jUser userDto = \_mapper.Map<jUser>(user);

return userDto;

}

**ThenInclude** mi consente di avere a disposizione la proprietà di navigazione che mi espone i dati della figuraaziendale.

## Aggiungere un elemento

// START Aggiunge elementi al many-to-many

Users esempio1 = \_context.Users.Include(x => x.UserMB\_Figure).FirstOrDefault(x => x.Id == 1);

UsersMB\_Figura umf2 = new UsersMB\_Figura()

{

UsersId = esempio1.Id,

MBFG\_ID = 1

};

esempio1.UserMB\_Figure.Add(umf2);

\_context.SaveChanges();

// Rileggo così posso, tramite l'automapper, rimappare correttamente tutte le entità FiguraAziendale

esempio1 = \_context.Users.Include(x => x.UserMB\_Figure)

.ThenInclude(y => y.FiguraAziendale)

.FirstOrDefault(x => x.Id == 1);

return \_mapper.Map<jUser>(esempio1);

// STOP Aggiunge elementi al many-to-many

## Cancellare tutti gli ementi

// START Cancella tutti gli elementi dalla lista many-to-many

Users esempio = \_context.Users.Include(x => x.UserMB\_Figure)

.ThenInclude(y => y.FiguraAziendale)

.FirstOrDefault(x => x.Id == 1);

esempio.UserMB\_Figure.**Clear()**;

\_context.SaveChanges();

return \_mapper.Map<jUser>(esempio);

// STOP Cancella tutti gli elementi dalla lista many-to-many

## Cancellare un determinato elemento

// START Cancella un elemento dalla lista many-to-many NON FUNZIONA il REMOVE mentre il REMOVEAT si

// quindi devo scorrere tutti gli elementi sulla tabella di join ed eseguire il removeat sull'elemento

// di interesse

Users esempio3 = \_context.Users.Include(x => x.UserMB\_Figure).ThenInclude(y => y.FiguraAziendale).FirstOrDefault(x => x.Id == 1);

for (int i = 0; i < esempio3.UserMB\_Figure.Count(); i++)

{

if (esempio3.UserMB\_Figure[i].UsersId == esempio3.Id && esempio3.UserMB\_Figure[i].MBFG\_ID == 1)

{

esempio3.UserMB\_Figure.RemoveAt(i);

}

}

\_context.ChangeTracker.Entries();

\_context.SaveChanges();

return \_mapper.Map<jUser>(esempio3);

// STOP Cancella un elemento dalla lista many-to-many