Escuelita Crombie 3er Edición

Seeds

Qué es Data seed en entity framework core?

Data seed, o importación de datos es la capacidad de importar datos iniciales en una base de datos al principio de la aplicación. Es una práctica muy útil cuando tenemos que precargar datos de referencia o estáticos a nuestro sistema.

Un ejemplo muy claro son las monedas con el símbolo y código, ya que no suelen cambiar, suelen ser un ejemplo (si se utilizan, claro) de precarga de datos.

Opcion 1: EjemploContext.cs

Instanciar las seeds directamente en el OnModelCreating

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{

        modelBuilder.Entity<Usuario>().HasData(
          new Usuario { Id = 1, Nombre = "Juan Perez" },
          new Usuario { Id = 2, Nombre = "Maria López" }
        );

    }
```

Opcion 2 : EjemploContext.cs

Crear una clase Seed, en este caso UserSeed y luego instanciar en el OnModelCreating

```
public class UserSeed : IEntityTypeConfiguration<Usuario>
{
  public void Configure(EntityTypeBuilder<Usuario> builder)
  {
    builder.HasData(
      new Usuario { Id = 1, Nombre = "Juan Perez" },
      new Usuario { Id = 2, Nombre = "Maria López" }
      );
  }
}
```

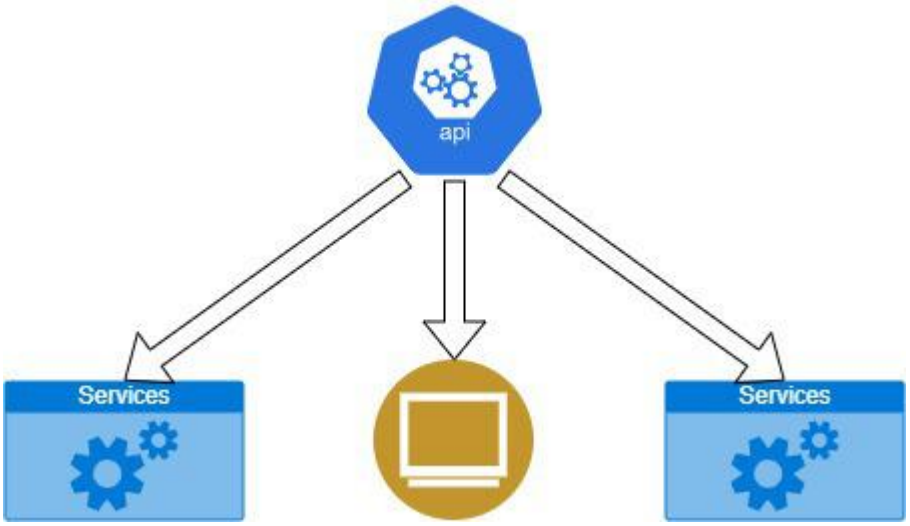
```
  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {

            modelBuilder.ApplyConfiguration(new UserSeed());

      }
```

Diferencia entre Dto y entidad

Qué es un DTO

Como su nombre indica un `DTO` es un "*Data Transfer Object*" y es el objeto que vamos a devolver desde nuestras API hacia otros servicios. Y únicamente debe contener datos, nada de lógica de negocio.

Este objeto debe ser serializable.

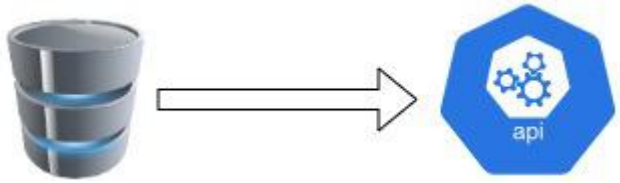


2 - Qué es una Entity

Una entidad se compone de dos puntos

A. Un objeto que representa datos de la base de datos; Como podría ser un objeto que represente cada fila de la base de datos.
B. Entidad que encapsula reglas críticas de nuestra aplicación que están relacionadas con este objeto. Y que pueden contener cierta lógica de negocio.

Hay que tener en cuenta que NUNCA debe contener más propiedades que las que contiene en la base de datos.



Get

Include

En Entity Framework Core, el método Include se utiliza para cargar datos relacionados de entidades asociadas

Por defecto, EF Core utiliza carga diferida (lazy loading), lo que significa que las propiedades de navegación relacionadas no se cargan automáticamente. El método Include permite hacer una carga ansiosa (eager loading) de esas propiedades.

Obtener respuestas con los Json serializados

```
builder.Services.AddControllers()
  .AddJsonOptions(options =>
  {
    options.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles;
    options.JsonSerializerOptions.WriteIndented = true;
  });
```

DireccionService.cs

```
public class DireccionService
{
  private readonly TiendaContext _context;

  public DireccionService(TiendaContext context)
  {
    _context = context;
  }

  public async Task<List<Direccion>> GetAllDireccionesAsync()
  {
    return await _context.Direcciones
      .Include(d => d.Usuario)
      .ToListAsync();
  }
```

```
    public async Task<Direccion?> GetDireccionByIdAsync(int id)
    {
        return await _context.Direcciones
            .Include(d => d.Usuario)
            .FirstOrDefaultAsync(d => d.Id == id);
    }
}
```

ThenInclude

Cuando hay relaciones anidadas (por ejemplo, Usuario -> Productos -> Categoria), se utiliza ThenInclude *después* de Include.

```
    public async Task<List<Direccion>> GetAllUsuariosAsync()
    {
        return await _context.Usuarios .Include(u => u.Productos)
                        .ThenInclude(p => p.Categoria)
                    .ToListAsync();
    }
```

SELECT

Select permite especificar exactamente qué datos necesitas de la base de datos.
Se usa para proyectar solo las propiedades necesarias de las entidades.
Evita transferir datos innecesarios desde la base de datos al servidor de la aplicación.

```
public async Task<List<Direccion>> GetAllUsuariosWhitSelectAsync()
    {
        return await _context.Usuarios
            .Select(u => new
            {
                    u.Id,
                    u.Nombre,
                    Direccion = new { u.Direccion.Calle, u.Direccion.Ciudad },
```

```
                Productos = u.Productos.Select(p => new { p.Nombre, p.Precio })
            })
            .ToListAsync();
    }
```