EXPERT NETWORK

Web Development with ASP.NET Core 7

S2

Internship

Our **Full Stack .NET Internship** will start soon.

Wanna join us?

- Send your CV at hr@expertnetwork.ro.
- Application Deadline: **March 19th**

For more details access our website: expertnetwork.ro

EXPERT NETWORK

The Sessions

1. Data Access
2. Concepts and Techniques
3. ASP.NET Core Introduction
4. ASP.NET Core Advanced
5. Deploy in the Cloud

*Note that each session builds upon the previous one.

For this session

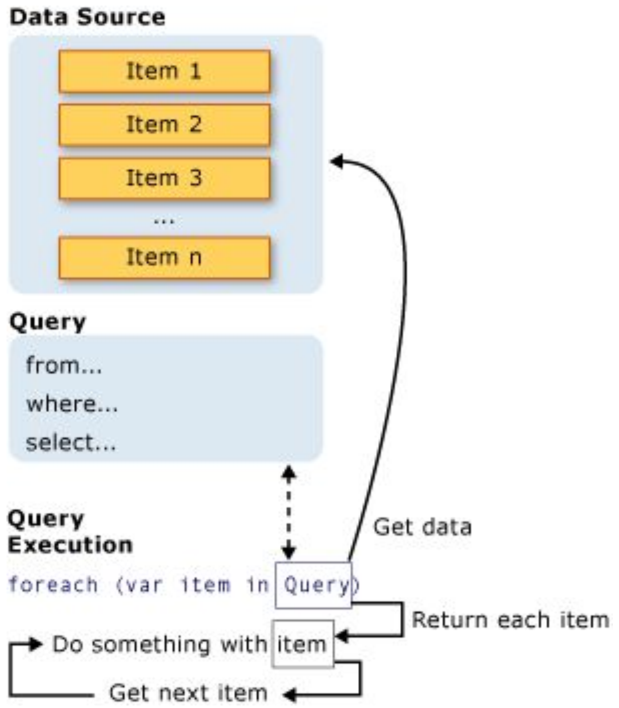
- LINQ
- Repository
- Inversion of Control
- Dependency Injection

*We will start from https://github.com/AlexandruCristianStan/FII-Practic-2023

LINQ

All LINQ query operations consist of three distinct actions:

1. Obtain the data source.
2. Create the query.
3. Execute the query.



https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

EXPERT NETWORK

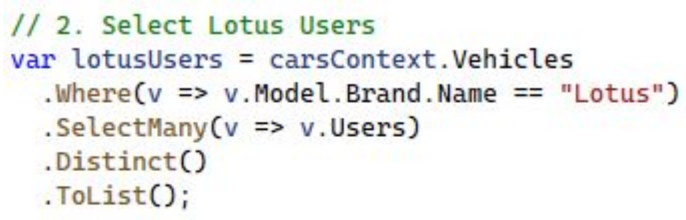
LINQ Operators

Filtering: Where
Projection: Select, SelectMany
Partitioning: Take, Skip, TakeWhile, SkipWhile
Grouping: GroupBy
Ordering: OrderBy, OrderByDescending, ThenBy, ThenByDescending
Conversion: ToList, **ToArray**, **ToDiscionary**
Aggregate: Count, Sum, Min, Max, Average
Quantifiers: Any, All

LINQ | Filtering

We can pass a **lambda expression** in the **Where** method to filter exactly what we want from a collection.

Usually this will translate into an expression in the **WHERE** SQL clause.



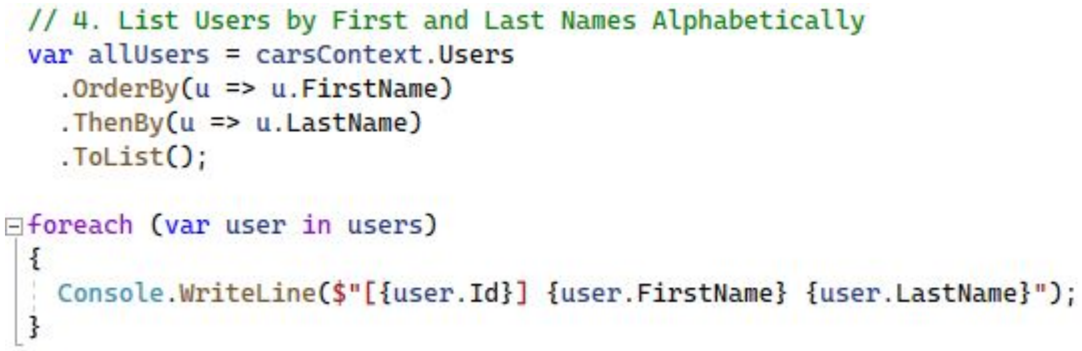
```
// 2. Select Lotus Users
var lotusUsers = carsContext.Vehicles
   .Where(v => v.Model.Brand.Name == "Lotus")
   .SelectMany(v => v.Users)
   .Distinct()
   .ToList();
```

https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/filtering-data

EXPERT NETWORK

LINQ | Ordering

OrderBy - Sorts values in ascending order.
OrderByDescending - Sorts values in descending order.
ThenBy - Performs a secondary sort in ascending order.
ThenByDescending - Performs a secondary sort in descending order.

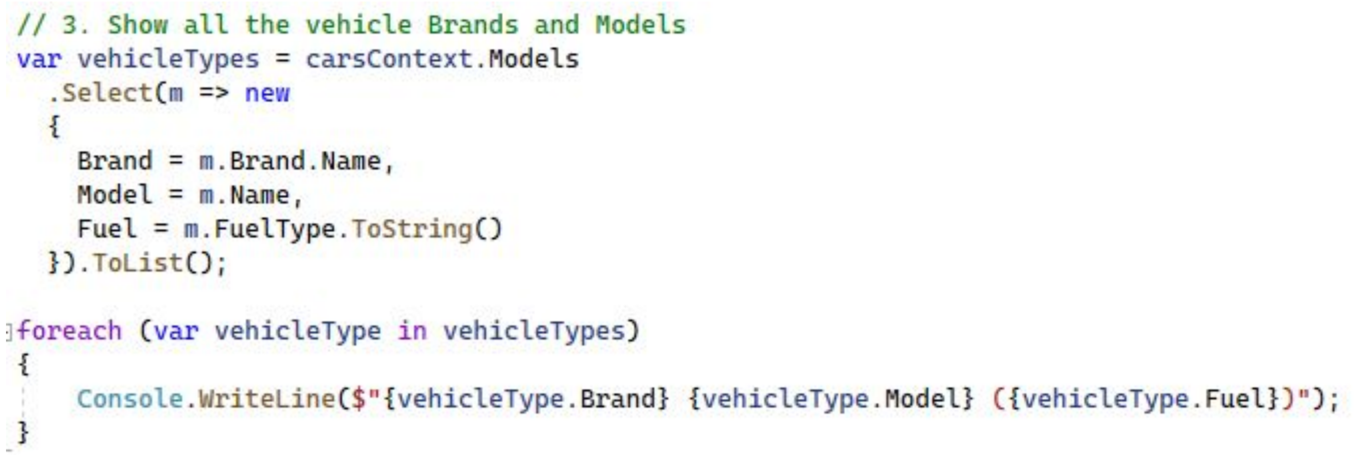


```
// 4. List Users by First and Last Names Alphabetically
var allUsers = carsContext.Users
  .OrderBy(u => u.FirstName)
  .ThenBy(u => u.LastName)
  .ToList();

foreach (var user in users)
{
  Console.WriteLine($"[{user.Id}] {user.FirstName} {user.LastName}");
}
```

https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/sorting-data

LINQ | Projection

Select - Projects values that are based on a transform function.
SelectMany- Projects sequences of values that are based on a transform function and then flattens them into one sequence.



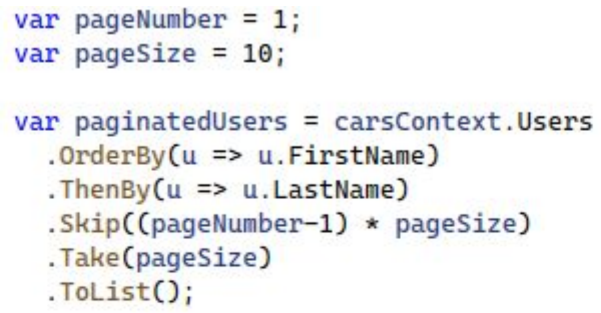
```
// 3. Show all the vehicle Brands and Models
var vehicleTypes = carsContext.Models
  .Select(m => new
  {
    Brand = m.Brand.Name,
    Model = m.Name,
    Fuel = m.FuelType.ToString()
  }).ToList();

foreach (var vehicleType in vehicleTypes)
{
    Console.WriteLine($"{vehicleType.Brand} {vehicleType.Model} ({vehicleType.Fuel})");
}
```

https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/projection-operations

EXPERT NETWORK

LINQ | Partitioning

Skip - Skips elements up to a specified position in a sequence.
Take - Takes elements up to a specified position in a sequence.

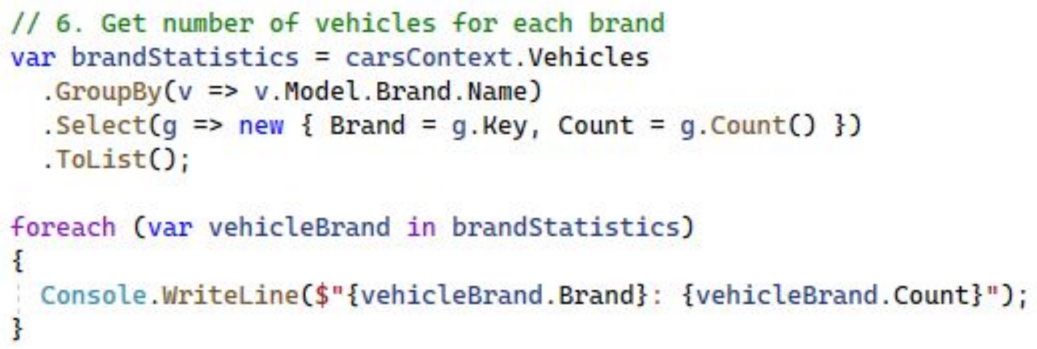


```
var pageNumber = 1;
var pageSize = 10;

var paginatedUsers = carsContext.Users
  .OrderBy(u => u.FirstName)
  .ThenBy(u => u.LastName)
  .Skip((pageNumber-1) * pageSize)
  .Take(pageSize)
  .ToList();
```

https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/partitioning-data

LINQ | Grouping

GroupBy - Groups elements that share a common attribute. Each group is represented by an IGrouping<TKey,TElement> object.



```
// 6. Get number of vehicles for each brand
var brandStatistics = carsContext.Vehicles
  .GroupBy(v => v.Model.Brand.Name)
  .Select(g => new { Brand = g.Key, Count = g.Count() })
  .ToList();

foreach (var vehicleBrand in brandStatistics)
{
  Console.WriteLine($"{vehicleBrand.Brand}: {vehicleBrand.Count}");
}
```

Repository Pattern

- Is intended to create an abstraction layer between the data access layer and the business logic layer of an application.
- Help insulate your application from changes in the data store and can facilitate automated Unit Testing or test-driven development (TDD).

Exercise 1.1: Create a User Repo

The **UserRepository** should have methods for:

- Create User
- SearchByName
- Delete User

Unit of Work

- The unit of work (UOW) class serves one purpose: to make sure that when you use multiple repositories, they share a single database context.

- When a unit of work is complete you can call the SaveChanges method on that instance of the context and be assured that all related changes will be coordinated.

https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

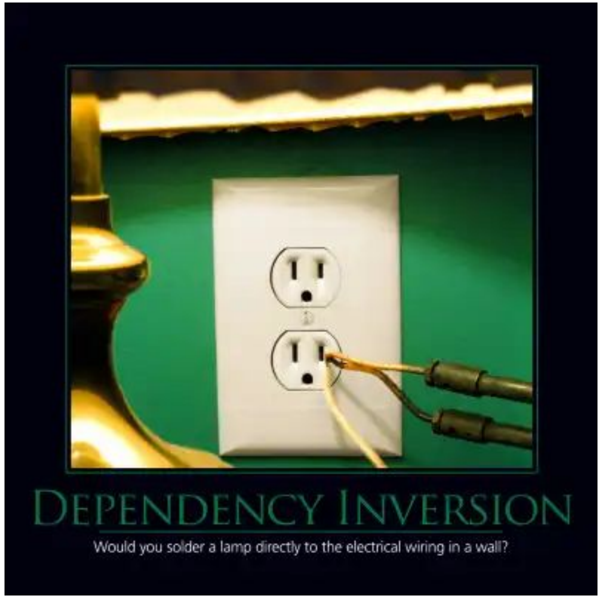
Exercise 1.2: EF Create Fake Users

1. Install **Bogus** NuGet package.
2. Create a Faker for User class.
3. Generate fake Users.
4. Add Users in the Users DbSet.

EXPERT NETWORK

Dependency Inversion

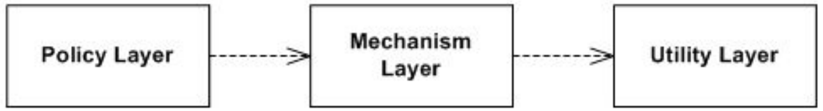
The Dependency Inversion Principle (DIP) states that high level modules should not depend on low level modules; both should depend on abstractions. Abstractions should not depend on details. Details should depend upon abstractions.


DEPENDENCY INVERSION
Would you solder a lamp directly to the electrical wiring in a wall?

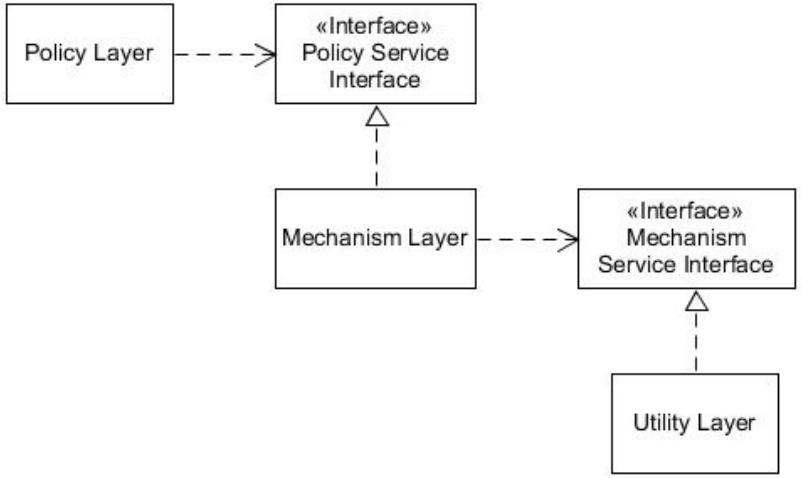
EXPERT NETWORK

Dependency Inversion

A **dependency** is an object that another object depends on.

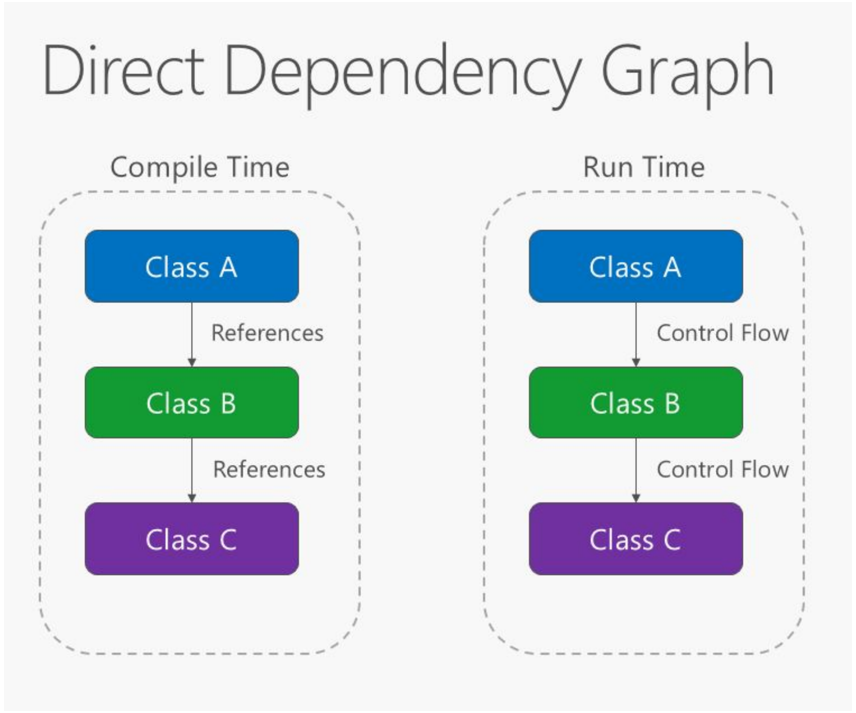


vs


Policy Layer
«Interface» Policy Service Interface
Mechanism Layer
«Interface» Mechanism Service Interface
Utility Layer

EXPERT NETWORK

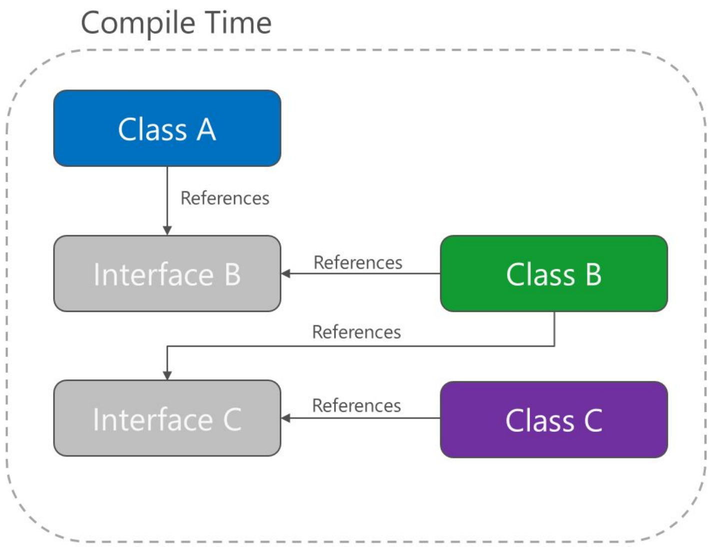
Dependency Inversion

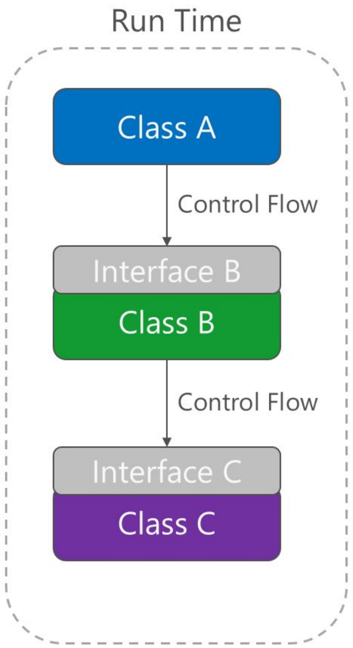


EXPERT NETWORK

Dependency Inversion

Inverted Dependency Graph





EXPERT NETWORK

Dependency Inversion

Benefits:

- Avoid high coupling
- Encourage reusability of higher layers
- Simplify Unit Testing
- Makes your code easier to maintain

EXPERT NETWORK

Dependency Injection

- The dependency injection (DI) software design pattern, which is a technique for achieving Inversion of Control (IoC) between classes and their dependencies.

- You can read more on how to configure DI in .NET by following the link: https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection

https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0

Service lifetimes

- **Transient** - services are created each time they're requested from the service container.
- **Scoped** - for web applications, a scoped lifetime indicates that services are created once per client request (connection).
- **Singleton** - services are created once and every subsequent request of the service implementation from the dependency injection container uses the same instance.

EXPERT NETWORK

Exercise 2: Setup DI

1. We will define a ServiceCollection
2. From it we will create a ServiceProvider
3. We will use the provider to get an instance of type IUserService
4. We will call the methods that we built

EXPERT NETWORK

Don’t forget the Feedback Form !!!

See you next time!

EXPERT NETWORK