Code Academy

Lecturer

Rokas Slaboševičius

Mini API Development and Dependency Injection

Data

Today you will learn

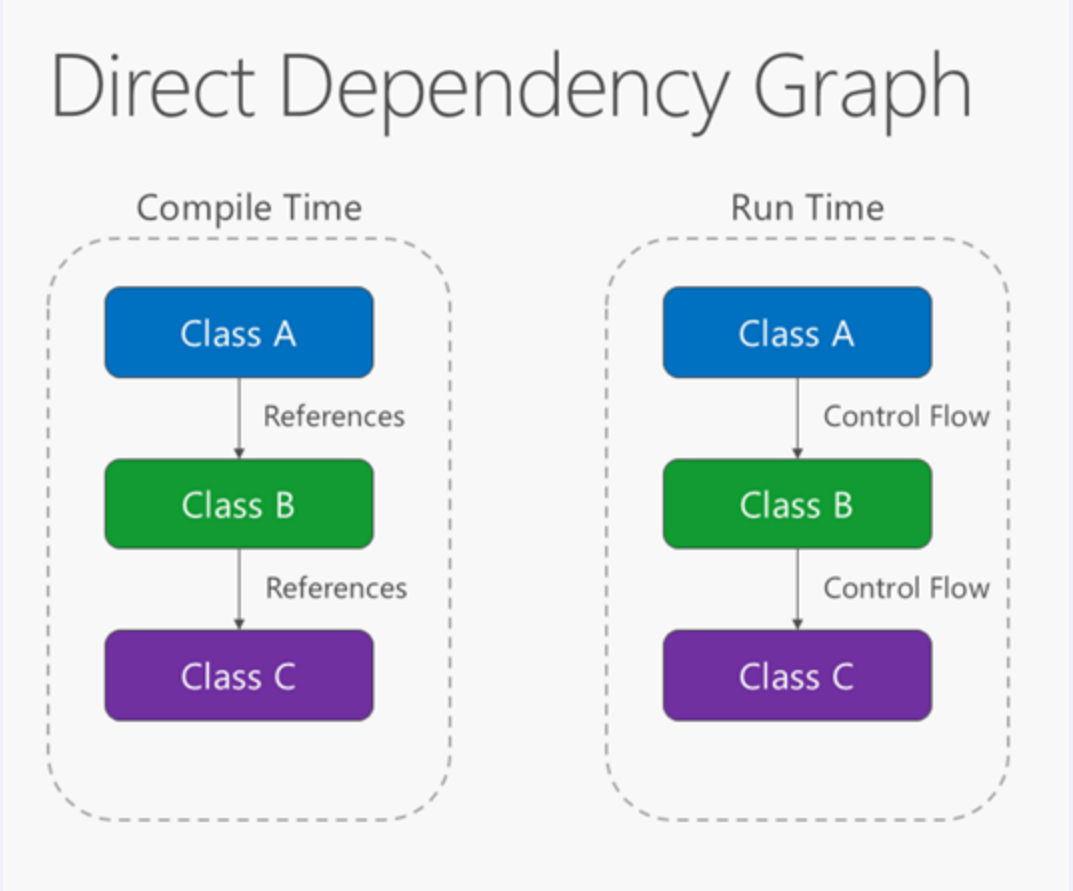
01 Dependency Injection

02 Inversion of control

Repeating the basics of Controllers and how to make HTTP requests

Dependency injection

Let's start by explaining the dependencies of the programme graph. Here we see that the class that is called both at compile time and runtime is considered to be a dependency

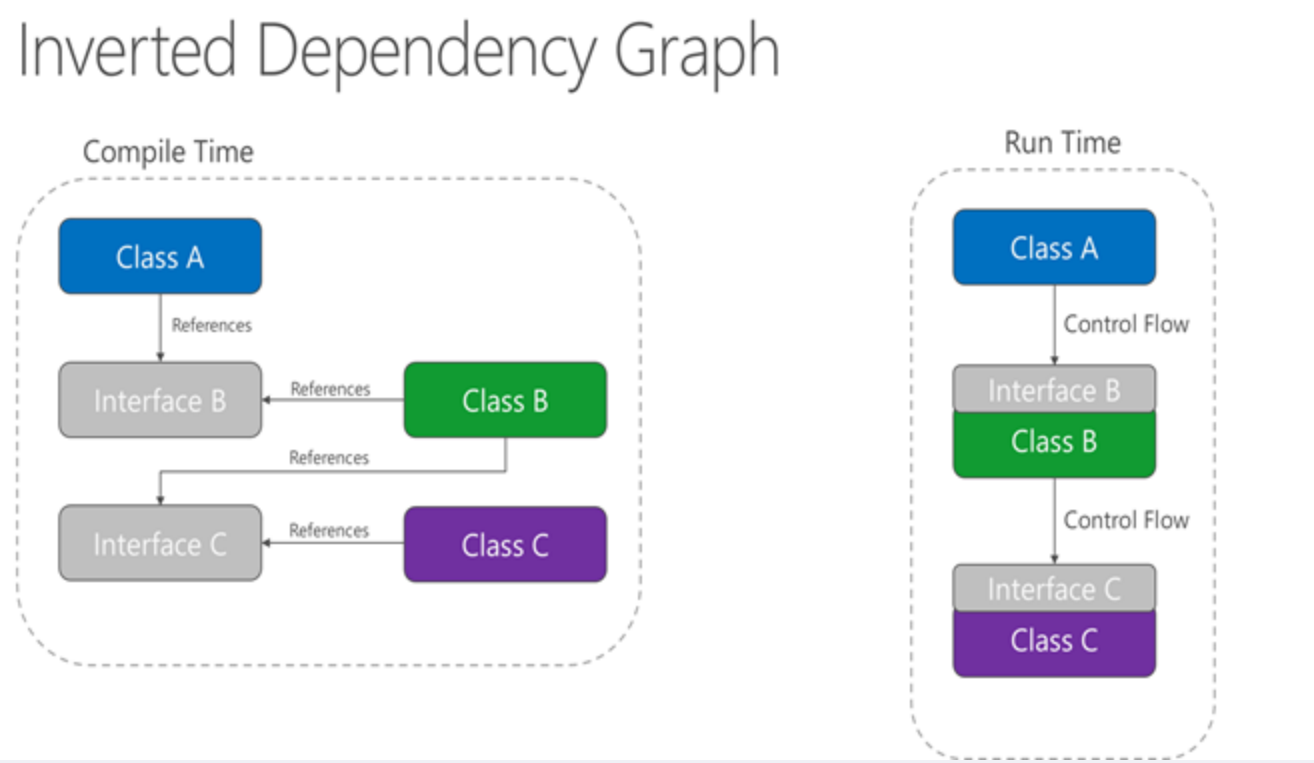


Inverted Dependency

The first step towards IoC (inversion of control) are inverted liabilities.

As we can see at compile time, class one does not call others. Everything is hidden behind the interfaces.

In this way the classes are independent of each other and everything is connected by interfaces whose **implementations** we can replace them with others.



Inversion of Control

The principle of Inversion of Control is to take control from the human.

In programming, this is used to easily create dependencies.

For example: we have a FileService class and we want the FileService class to have a DatabaseService class, but in order to assign the DatabaseService class, we will need to specify a new DatabaseService object when initializing the FileService object ourselves. If the DatabaseService class has even more dependencies then even more code will be required when creating the FileService object.

With Dependency Injection we can operate through interfaces and the application will find what it needs to create services.

Service registration

The difference between Scoped and Singleton:

```
services.AddSingleton<IApplication, Application>();
services.AddScoped<IService, Service>();
```

There is also Transient, but more on that later.

.NET API Project

Mini API Development and Dependency Injection

Task 1

1. Set up a new ASP.NET Core API project.
2. Create a simple model: Define a `Product` model with properties for `Id`, `Name`, and `Price`.
3. Implement a basic in-memory data store: Create a static list of `Product` objects that will act as your in-memory database.
4. Develop a GET endpoint to retrieve all products: Implement an API endpoint that returns all products from your in-memory data store.
5. Create a POST endpoint to add a new product: Implement an API endpoint that adds a new `Product` to your in-memory data store.
6. Implement Dependency Injection for a service: Create a `ProductService` that handles data access logic for products and use dependency injection to access it in your API.
7. Refactor the GET and POST handlers to use `ProductService`: Modify your API endpoints to use the `ProductService` for data operations.
8. Add basic validation in your POST endpoint: Ensure the product name and price are provided when adding a new product.
9. Implement a DELETE endpoint to remove a product: Create an API endpoint that allows deleting a product by its `Id`.
10. Test your API with Postman or a similar tool:** Use Postman to test all your API endpoints, verifying that each behaves as expected.

Headline

www.youtube.com

Useful information