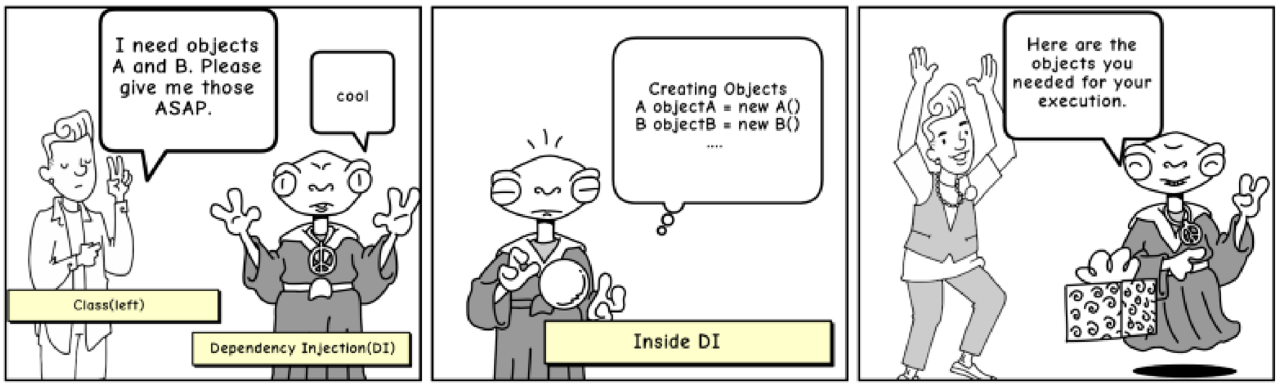
Lesson 41 - Dependency Injection

What is a dependency?

Dependency or dependent means *relying* on something for support. Like if I say we are relying too much on mobile phones than it means we are dependent on them. In context to programming, when class `A` uses some functionality of class `B`, then its said that `A` has a dependency of `B`.

What is dependency injection?

In **.NET**, before we can use methods of other classes, we first need to create the object of that class (i.e. `A` needs to create an instance of `B`). Transferring this task of creating the object to someone else and directly using the dependency is called *dependency injection*.


I need objects A and B. Please give me those ASAP.
cool
Class(left)
Dependency Injection(DI)
Creating Objects A objectA = new A() B objectB = new B() ....
Inside DI
Here are the objects you needed for your execution.

This comic was created at www.MakeBeliefsComix.com. Go there and make one now!

Why should we use dependency injection?

Let's say we have a `Car` class which contains various objects such as wheels, engine, etc. Here the car class is responsible for creating all the dependency objects. Now, what if we decide to ditch `MRFWheel` in the future and want to use `YokohamaWheel`?

We will need to *recreate* the car object with a new `Yokohama` dependency.

- To replace `MRFWheel` with `YokohomaWheel`, the `Car` class must be modified.
- If `MRFWheel` has dependencies, they must also be configured by the `Car` class. In a large project with multiple classes depending on `Car`, the configuration code becomes scattered across the app.

- This implementation is difficult to unit test.

You can think of DI as the middleman in our code who does all the work of creating the preferred wheels object and providing it to the `Car` class. It makes our Car class **independent** from creating the objects of Wheels, Battery, etc. It solves these problems by:

- The use of an interface or base class to abstract the dependency implementation.
- Registration of the dependency in a service container. ASP.NET Core provides a built-in service container, IServiceProvider. Services are typically registered in the app's Program.cs file.
- Injection of the service into the constructor of the class where it's used. The framework takes on the responsibility of creating an instance of the dependency and disposing of it when it's no longer needed.

Steps to implement dependency injection

1. Create an interface which defines your dependency, so if we consider the above example, for wheels, we create an interface `IWheel` which will define a wheel.
2. Implement this interface by a concrete type, such as `MRFWheel` and/or `YokohomaWheel`.
3. Register your dependency with your `builder.Services` in your `Program.cs`.
4. Modify your `Car` class to contain the `IWheel` interface instead of a concrete class, and assign it in the constructor.

> For more information, kindly visit the following article

Dependency injection in ASP.NET Core

Learn how ASP.NET Core implements dependency injection and how to use it.

learn.microsoft.com