**Dependency Injection Container**

**IService Collection and IServiceProvider**

Picture that you have class called “Logger” and this class implements an interface called “ILogger”. So, You can’t use the class by itself, you have to use the new Keyword in order to create an instance of the logger class or an object of type logger class. You can call the class as the “Metadata”, and you can think of Object as the “Data”. Metadata is also referred to as data for data.

It’s type of data that describes the actual data, but itself it’s not the actual data.

A screen shot of a computer

Description automatically generated

So, in dependency injection there is a bucket or container which we put all the metadata inside it.

For example – imagine that in the bucket your metadata is that a logger has to be translated to a class called default logger, because in dependency injection, we rely on interfaces in order to reduce the coupling and reduce the dependency.  
Instead of putting classes and objects in the metadata bucket, our metadata is like this. We normally say every time you see an interface called a logger, what I want is an object of type default logger. So we put all the metadata in a bucket, and that bucket, which is our dependency injection container, is represented in .Net with a class called “Service Collection”.

So, ServiceCollection which implements an interface called “IServiceCollection” that is your Dependency Injection container and it includes your metadata, it doesn’t include your actual data, it’s only metadata that sits in your bucket. And your metadata begins with your “Interface” and then it resolves 90% of time a class or an Object.

Now how do we get the actual objects that we need?

* So because of the Single responsibility principle, which is one of the object oriented design principles, we want everything to have only one responsibility. So “ServiceCollection” is only responsible for including and containing the metadata. It doesn’t actually create the actual Objects
* For you to do that it will give you an Object of type “ServiceProvider”, which implements an interface called “IServiceProvider” and “ServiceProvider” is responsible to create instances of objects and pass them to YOU.
* So, we’ll deal with ServiceCollection to put our metadata in it and then “ServiceCollection” will give us an instance of service provider and then we always ask “ServiceProvider” to give us the Objects that we need.

