Dependency Injection of Internals

By Andrew Hinkle

Working with Legacy applications you'll find class libraries littered with public classes implemented by other class libraries that were never meant for consumption. To mitigate the object leakage developers change public classes to [internal](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/internal) and expose their usage via a few limited public classes.

Taking advantage of "[poor man's dependency injection](https://softwareengineering.stackexchange.com/questions/364090/is-poor-mans-dependency-injection-a-good-way-to-introduce-testability-to-a-lega)" (PMDI) all of your internal classes may have two constructors. One constructor expects an implementation of an interface and a second that creates a new instance of the implementation. This tightly couples the implementation details with the class. The best part is that you can [add unit tests that inject mock versions](https://stackoverflow.com/questions/20423714/mocking-using-moq-in-c-sharp) ([TestDoubles](https://martinfowler.com/bliki/TestDouble.html)) of the interfaces such as repositories, configuration settings, etc. This allows you to test just the logic in the class and not what is injected.

Of course you can take it to the next stage and remove the second constructor causing the tight coupling. To accomplish this feat you now move the creation of the implementation in the calling class. At this point you've just moved the cheese up a layer that now maintains that dependency. In order to take full advantage of this we introduce [Inversion of Control frameworks](https://stackoverflow.com/questions/21288/which-net-dependency-injection-frameworks-are-worth-looking-into).

Here's the rub, once you get out of the class library, you can't use the internal classes you want to use for the implementation. Sure, you could change them to public classes, but now your library is naked for all to see. You could update the class library properties to add an [InternalsVisibleTo](https://stackoverflow.com/questions/42235401/unit-testing-internal-methods-in-vs2017-net-standard-library) to the calling assembly, but that defeats the whole purpose given the tight coupling to the assemblies. Perhaps one of the frameworks can handle registering internals, but I haven't found it.

While working with .NET Core Dependency Injection I ran into the same problem yet again.

In this article we'll create a brand new ASP .NET Core WebApi with a class library.