**Implementing Dependency Injection in Sitecore**

There are a few different ways to implement dependency injection in Sitecore. Once upon a time, the only way to implementing DI was to use third-party libraries like [Simple Injector](https://simpleinjector.org/index.html). Since Sitecore 8.2 release, Sitecore includes build in dependency injection based on [Microsoft’s Dependency Injection for ASP.NET Core](https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/dependency-injection?view=aspnetcore-3.0). We can still use third-party modules for DI in the latest Sitecore and it is still a valid option with a few benefits over the out-of-the box DI solutions.

Choosing approach can be daunting. Even [this StackExchange post](https://sitecore.stackexchange.com/questions/490/best-approach-for-dependency-injection-in-sitecore-8-2) doesn’t give us a definitive answer. At the end of the day, whatever solution you implement is going to work. Just pick whatever feels right for the solution you are working on. Recently, I implemented DI in Sitecore 8.2 with the out-of-box Sitecore Dependency Injection. The reason for this choice is:

* Sitecore is already using it. For that reason, it will be supported by Sitecore. All we need to do is to register our custom configurator.
* We did not need to add extra third-party module dependency to the existing solution.
* The performance is said to be as fast as the best performing third-party DI containers, like Simple Injector.
* Habitat implements it. All I needed to do is to take [the example code](https://github.com/Sitecore/Habitat/tree/master/src/Foundation/DependencyInjection/code) and implement it in the existing solution. Although the solution is for Sitecore 9, it works for 8.2.
* The DI container is referenced from the composition root with this solution (see details for [this fantastic blog](https://www.linkedin.com/pulse/dependency-injection-sitecore-helix-based-application-arvind-gehlot-1/?lipi=urn%3Ali%3Apage%3Ad_flagship3_profile_view_base_post_details%3BAPrVTkRQTTWBzyB378DrJQ%3D%3D) post).
* I implemented DI in an already matured Sitecore code base. Habitat implementation has the minimum disruption to the existing solution and allowed us to introduce DI slowly to the entire code base.

When I implemented DI, I started with implementing it in a Feature project for my own humble Sitecore that I installed on my laptop for learning. There are heaps of posts that help you to implement DI in the real-world solution. This post gives you the example of implementing DI at a project level just to understand how it works.

This is fairly quick and simple. But, I promise this is a fun and satisfying activity. Let’s get started.

**Aim**

In the [previous post](https://www.mydatahack.com/using-data-source-in-controller-renderings-sitecore/), we created a controller depending on repository. The dependency injection is done in the constructor. We want to register FrontEndTestController and FrontEndTestRepository in the DI container so that we can get rid of the line injecting the concrete implementation.

<script src="https://gist.github.com/mydatahack/d7be623b24408f8d5cad47e5fbeda492.js"></script>

At the end of this post, the poor man’s DI will be gone for good!

**Steps**

1. **Add dlls for Microsoft DI**

We need to add two dlls for the project (in this example, SitecoreDev.Feature.Design). These dlls come with the software installation. I suggest to copy & paste from the web root bin directory and move them to the code base lib folder.

* Microsoft.Extensitons.DependencyInjection
* Microsoft.Extensions.DependencyInjection.Abstractions.

1. **Register dependencies**

Create RegisterDependencies.cs and register dependencies manually as below.

<script src="https://gist.github.com/mydatahack/8f5496124ab092bf9cf9ee42f6a0d08e.js"></script>

1. **Register service**

Create a config file Feature.Design.config in App\_Config/Include. This will allow Sitecore to register dependencies.

<script src="https://gist.github.com/mydatahack/770be77ef0e66185fd57f845f7b053da.js"></script>

1. **Remove the poor man’s DI constructor**

We can now remove the constructor instantiating the actual implementation.

<script src="https://gist.github.com/mydatahack/b263dfe4734b158370cd741d9bb9e0c2.js"></script>

That’s it. You can also check the container registration from <http://[instance]/sitecore/admin/showservicesconfig.aspx>.

**Next Steps**

Now that you know how this works, all you need to do is to register dependencies when the application starts through Sitecore pipeline. Here are the steps I took.

* Create a DependencyInjection project in Foundation and move the entire code from Habitat solution [here](https://github.com/Sitecore/Habitat/tree/master/src/Foundation/DependencyInjection/code).
* Cleaned up ServiceCollectionExtensions methods (removed unnecessary methods and turned unnecessary public methods to privates).
* Added Scoped into Lifetime enum and ServiceCollectionExtensions methods for full lifetime management support.
* Wrote unit tests on ServiceCollectionExtensions.
* Added dependency registration attributes (as described [here](https://www.linkedin.com/pulse/dependency-injection-sitecore-helix-based-application-arvind-gehlot-1/?lipi=urn%3Ali%3Apage%3Ad_flagship3_profile_view_base_post_details%3BAPrVTkRQTTWBzyB378DrJQ%3D%3D)) to a few classes and implemented in one controller to start with.

Now, it’s your turn to implement DI in your Sitecore solutions.

**References**

There are so many wonderful references that we can use. Here are the ones I recommend you to read.

* [Dependency injection from offical Sitecore 9 documentation](https://doc.sitecore.com/developers/90/sitecore-experience-manager/en/dependency-injection.html)
* [Dependency injection in Sitecore Helix based application in practical way-Part II](https://www.linkedin.com/pulse/dependency-injection-sitecore-helix-based-application-arvind-gehlot-1/?lipi=urn%3Ali%3Apage%3Ad_flagship3_profile_view_base_post_details%3BAPrVTkRQTTWBzyB378DrJQ%3D%3D)
* [Dependency injection in Sitecore 8.2](https://kamsar.net/index.php/2016/08/Dependency-Injection-in-Sitecore-8-2/)
* [Habitat solution](https://github.com/Sitecore/Habitat/tree/master/src/Foundation/DependencyInjection/code)