**StarLabs 2022 - Documentation**

1. **IOptions Patterns**

***How to use the IOptions pattern for configuration in ASP.NET Core?***

Almost every project will have some settings that need to be configured and changed depending on the environment, or secrets that you don't want to hard code into your repository. The classic example is connection strings and passwords etc which in ASP.NET 4 were often stored in the <applicationSettings> section of web.config. In ASP.NET Core this model of configuration has been significantly extended and enhanced. Application settings can be stored in multiple places - environment variables, appsettings.json, user secrets etc - and easily accessed through the same interface in your application. Further to this, the new configuration system in ASP.NET allows (actually, enforces) strongly typed settings using the IOptions<> pattern.

The options pattern uses classes to provide strongly typed access to groups of related settings. When configuration settings are isolated by scenario into separate classes, the app adheres to two important software engineering principles:

1. [Encapsulation](https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles#encapsulation):
   * Classes that depend on configuration settings depend only on the configuration settings that they use.
2. [Separation of Concerns](https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles#separation-of-concerns):
   * Settings for different parts of the app aren't dependent or coupled to one another

***Binding the configuration to your classes***

In order to ensure your appsettings.json file is bound to the MySettings class, you need to do 2 things

1. Setup the ConfigurationBuilder to load your file
2. Bind your settings class to a configuration section

[More on: (Ctrl + Click - here)](https://andrewlock.net/how-to-use-the-ioptions-pattern-for-configuration-in-asp-net-core-rc2/)