Consuming Services Inside Startup

During Startup.ConfigureServices

During application startup, the Startup.ConfigureServices method is called to register services and configuration with DI.
This method allows those services to be consumed across the application.

However. During the execution logic in ConfigureServices, there is no active service provider to pull from.
So, we are somewhat prevented from accessing registered elements, as we register others.

But, the `IServiceCollection` that we register services to (in ConfigureServices), can be “built” to provide a temporary service provider that gives us access to anything already registered.

Here’s an example of how to make a temporary service provider, from inside ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IFooService, FooService>();

    // Build the intermediate service provider
    var sp = services.BuildServiceProvider();

    // This will succeed.
    var fooService = sp.GetService<IFooService>();
    // This will fail (return null), as IBarService hasn't been registered yet.
    var barService = sp.GetService<IBarService>();
}
```

The above example shows that we have registered a FooService.

And somewhere after that, we build a temporary service provider that can give us an instance of FooService, using the GetService call.

Limitations

Normally, this technique could be used to retrieve registered configuration data, from an IOptions instance.
That's a fair use of this technique, as config doesn't during startup.

However. We do need to remember that any singleton instances returned from a temporary service provider, like above, will NOT be the same instance that would be returned from the live service provider.

So. If we truly wanted to access singleton services with this technique, we would have to explicitly instantiate the singleton service, and register that instance with the IServiceCollection.

But in doing so, we already have an object reference of the service, and don't require this technique.

So, this technique is only valuable for accessing transient or scoped service instances, or retrieving configuration, needed to setup services and other things.

During Startup.Configure

The Configure method gets called after ConfigureServices has registered everything with DI.

So, once the Configure method is executing, we can resolve a healthy service provider, if we need services or config during the Configure method.

We can get a service provider instance in a couple of ways.

One is, we can resolve it from IApplicationBuilder, like this:

```
public void Configure(IApplicationBuilder app)
{
    var serviceProvider = app.ApplicationServices;
    var hostingEnv = serviceProvider.GetService<IHostingEnvironment>();
}
```

The above example shows us resolving the root service provider, and using it to get an instance of IHostingEnvironment.

Another way to get a service provider in Configure, is to include the service provider in the method parameters.
Specifically, we can add any DI-registered services/config as a method parameter of Startup.Configure.

We can even add the root service provider, this way.

Here’s an example of directly accessing the service provider in Startup.Configure:

```
public void Configure(
    IApplicationBuilder application,
    IServiceProvider serviceProvider)
{
    // By type.
    var service1 = (MyService)serviceProvider.GetService(typeof(MyService));

    // Using extension method.
    var service2 = serviceProvider.GetService<MyService>();

    // ...
}
```

Revision #3
Created 30 April 2025 01:34:19 by glwhite
Updated 30 April 2025 02:11:34 by glwhite