Duplicating .NET Core DI

NOTE: If you are looking for how to access DI services, see this: HowTo Access DI Services

Here are the steps to duplicating the .NET Dependency Injection process, to create a ServiceProvider instance. This is especially useful in unit testing, for classes and services that need the IServiceProvider.

Creating a ServiceProvider takes three steps:

- Initiate the Service Collection
- Register services
- Build the Provider

Below is a method call that does the needed steps, and returns a IServiceProvider.
It provides a callback

```
private IServiceProvider Setup_DI()
{
    // Need a usable reference to the service provider...
    ServiceProvider? _serviceProvider = null;

    // Initiate the service collection...
    IServiceCollection services = new ServiceCollection();



    // Register any services and config to be available via DI...

    //services.Add(new ServiceDescriptor(typeof(InterfaceA), typeof(ClassA), ServiceLifetime.Singleton));
    //services.AddSingleton<InterfaceB, ClassB>();

    // Add a FAKE Object Security Service....
    services.AddScoped<OGA.Auth.OSS.IObjectSecurityService, ObjectSecurityService_Fake>();

    // Declare the remapping class that allows the user management service to look like a directory service, to the User Security Cache...
    // This service is consumed by the constructor of th User Context Caching service.
    services.AddSingleton<OGA.Auth.UserSecContext.IDirectoryService_for_UserSecurityCache_v2, DirectoryService_Adapter_for_USC_v2>();
```

```
    // And as well, register the directory service adapter, directly, as we require other methods of it, that are not in the interface...
    services.AddSingleton<DirectoryService_Adapter_for_USC_v2, DirectoryService_Adapter_for_USC_v2>();

    // Declare the User Security Context Caching service...
    // This service provides group membership and account status for the JWT middleware and Object Security Service.
    services.AddScoped<OGA.Auth.UserSecContext.USecContext_CachingAgent_v2>();



    // Create the ServiceProvider, so it can be used...
    _serviceProvider = services.BuildServiceProvider();



    // Return the provider...
    return _serviceProvider;
}
```

Revision #2
Created 30 April 2025 21:00:01 by glwhite
Updated 30 April 2025 21:06:38 by glwhite