HowTo Access DI Services

> NOTE: If you are looking for how to generate an instance of ServiceProvider, outside of NET Core runtime, like for a unit test, see this: Duplicating .NET Core DI

Here's a good reference on how DI scoping works: The dangers and gotchas of using scoped services in IConfigureOptions

See this for resolving services during startup: Consuming Services Inside Startup

Any service added in ConfigureServices via startup.cs or program.cs, can be access multiple ways.

Controller Constructor

Injected into the constructor of a controller, and used by an action:

```
public class HomeController : Controller
{
    private INotificationHelper helper = null;

    public HomeController(INotificationHelper helper)
    {
        this.helper = helper;
    }

    public IActionResult Index()
    {
        helper.Notify();

        return View();
    }
}
```

Controller Action

Injected into an action (if only a few actions of a controller need the service):

```
public IActionResult Index([FromServices]INotificationHelper helper)
{
    helper.Notify();
    return View();
}
```

Unknown Service Needs

If many services are needed by a controller, or it is unknown which services will be required, the service provider can be injected instead:

```
public class HomeController : Controller
{
    private IServiceProvider provider = null;

    public HomeController(IServiceProvider provider)
    {
        this.provider = provider;
    }

    public IActionResult Index()
    {
        INotificationHelper helper = (INotificationHelper)
        provider.GetService(typeof(INotificationHelper));

        helper.Notify();

        return View();
    }
}
```

From an Http Context

Any service can be used where an HTTP context is available:
NOTE: Be sure to test that the returned service instance is NOT null before use.

```
public IActionResult Index()
{
    INotificationHelper helper = (INotificationHelper)
HttpContext.RequestServices.GetService(typeof(INotificationHelper));

    return View();
}
```

Services from New Scope

Here’s a method of how to create a service or data context from a new scope.
This can be useful if spawning an asynchronous database operation from a web controller, that must run independently (fire-and-forget).
Such a scenario would normally trigger a disposed exception because the data context would get disposed with the rest of the resources in the web call (because they went out of scope).

This method creates a new scope, from a service provider.

```
static private void Accept_UserDevice_TokenData(IServiceScopeFactory svcscopefactory, DeviceToken_DTO
tokendata, string corelationid)
{
     // Run all this inside its own thread...
     _ = Task.Run(() =>
     {
         ClientToken_v1 ct = null;

         IServiceScope newscope = null;
         try
         {
             // Create the service scope in a try-finally, so it will get automatically disposed...
             newscope = svcscopefactory.CreateScope();

             Bliss.Notifications.DataContexts.DataContext dbctx = null;
             try
```

```
            {
                // Create the data context inside a try-finally, so it can be automatically disposed...
                dbctx =
newscope.ServiceProvider.GetRequiredService<Bliss.Notifications.DataContexts.DataContext>();

                try
                {
                    // Do something with the newly scoped context...
                    var tks = dbctx.ClientTokens_v1.Where(m => m.UserId == tokendata.UserId && m.DeviceId ==
tokendata.DeviceId);
                }
                catch (Exception e)
                {
                    OGA_SharedKernel.Logging_Base.Logger_Ref?.Info("Failed to store new client push token.");
                }
            }
            finally
            {
                dbctx?.Dispose();
            }
        }
        catch(Exception e)
        {
            int x = 0;
        }
        finally
        {
            newscope?.Dispose();
        }
    });
}
```

And, here's an article on how to create a static factory for creating a service provider instance, anywhere: https://doc.xuwenliang.com/docs/dotnet/1611

Here's how to get a service scope, anywhere.
The trick is that everything is stored in the IHost instance.

Expose a reference of the host instance in Program.cs.
Then, reference that wherever a scope and services are required, like this:

```
var scope = Program.host_ref.Services.CreateScope();

var svcprovider = scope.ServiceProvider;

// Get a database setup instance we can work with...
var datasvc = svcprovider.GetService<IDBInitianizationService>();
if (datasvc == null)
{
    // No database setup service was created.
    // Check setup for startup order...

    NETCore_Common.Logging.Logging.Logger_Ref?.Error(
        "Program:Is_Database_UptoDate - Database setup service was not found. Check Setup.cs to ensure service
is registerd.");

    return -1;
}
// Have a user service reference.
```

Revision #3
Created 8 October 2025 02:54:23 by glwhite
Updated 8 October 2025 03:00:42 by glwhite