Implementing logging in an ASP.NET Core application using the built-in logging providers.

**Step-by-Step Guide to Implement Logging in ASP.NET Core**

**1. Create a New ASP.NET Core Project**

If you don't have an existing project, you can create a new one using the .NET CLI:

bash

dotnet new mvc -n SimpleLoggingExample

cd SimpleLoggingExample

**2. Configure Logging in Program.cs**

By default, ASP.NET Core includes logging configuration, but let's ensure it's properly set up.

**Example** Program.cs:

csharp

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

})

.ConfigureLogging(logging =>

{

logging.ClearProviders();

logging.AddConsole();

logging.AddDebug();

// Add other logging providers as needed

});

}

**3. Inject the Logger into a Controller**

Inject the ILogger service into a controller where you need to log information.

**Example: HomeController.cs**:

csharp

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

\_logger.LogInformation("Index action called.");

return View();

}

public IActionResult Error()

{

\_logger.LogError("An error occurred.");

return View();

}

}

**4. Configure Logging Levels in appsettings.json**

Adjust the logging levels in the appsettings.json file to control the verbosity of logs.

**Example** appsettings.json:

json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

}

}

**Summary**

1. **Create a New Project**: Create a new ASP.NET Core MVC project using the .NET CLI.
2. **Configure Logging**: Set up logging providers in Program.cs.
3. **Inject ILogger**: Inject the ILogger service into controllers where logging is needed.
4. **Configure Logging Levels**: Adjust logging levels in the appsettings.json file.

By following these steps, you can quickly and effectively implement logging in your ASP.NET Core application using the built-in logging providers. This simple example should help you get started with logging and provide a foundation for more advanced logging scenarios. If you have any more questions or need further examples, feel free to ask3