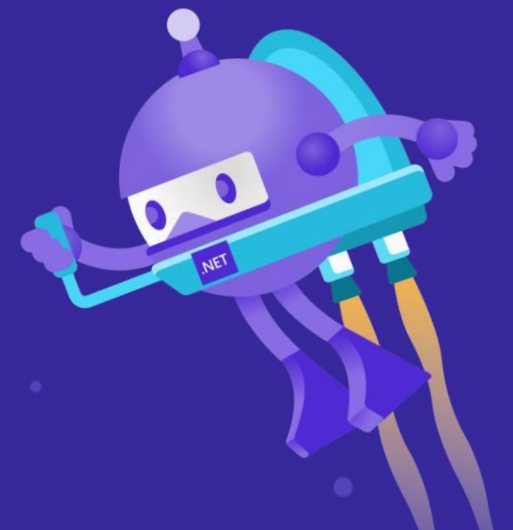
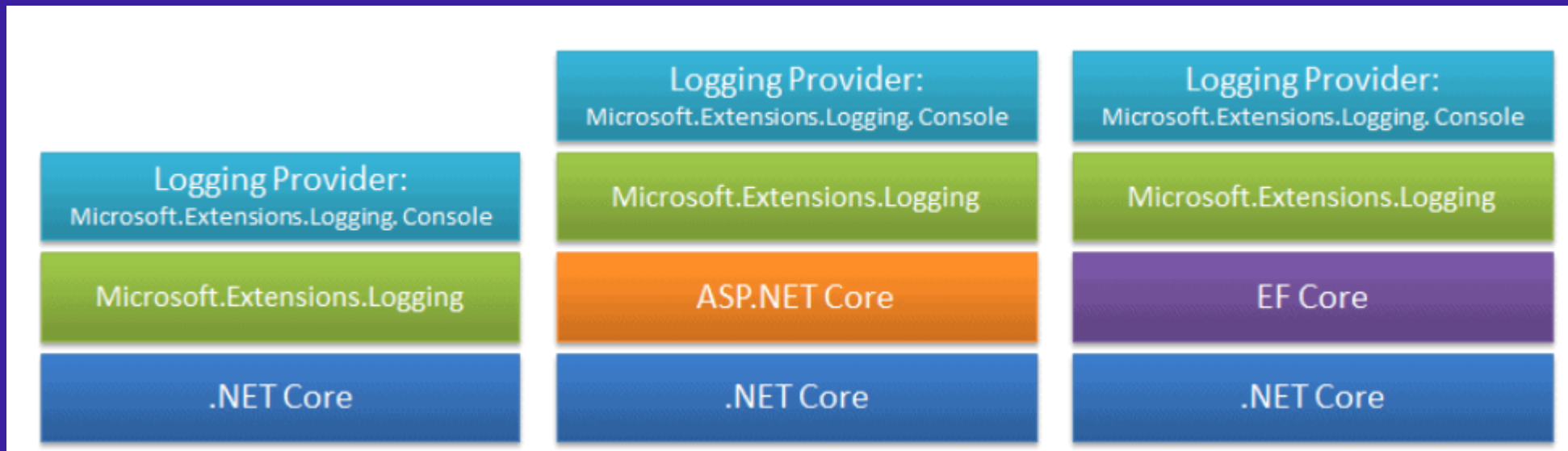


Day 15: Logging in ASP.NET Core

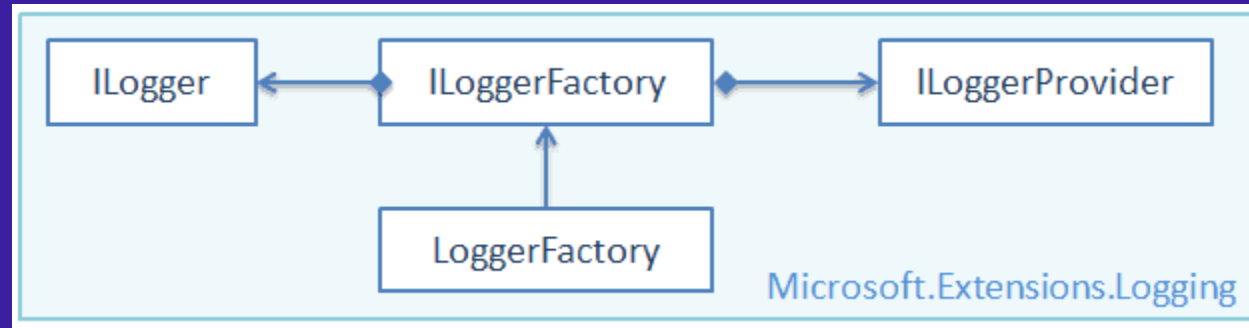


Logging

- ASP.NET Core has built in support for logging
- Building Blocks
 - Logging API – Included with Microsoft.Extensions.Logging package
 - Logging Providers - onsole, Debug, TraceListeners etc
- Supports a variety of built in as well as third party providers



Logging API



- ASP.NET Core has built in support for logging
- Building Blocks
 - Logging API – Included with Microsoft.Extensions.Logging package
 - Logging Providers - onsole, Debug, TraceListeners etc
- Supports a variety of built in as well as third party providers

Logging Providers

NuGet Package	Output Target
Microsoft.Extensions.Logging.Console	Console
Microsoft.Extensions.Logging.AzureAppServices	Azure App Services 'Diagnostics logs' and 'Log stream' features
Microsoft.Extensions.Logging.Debug	Debugger Monitor
Microsoft.Extensions.Logging.EventLog	Windows Event Log
Microsoft.Extensions.Logging.EventSource	EventSource/EventListener
Microsoft.Extensions.Logging.TraceSource	Trace Listener

Logging Providers – Third Party

elmah.io	https://elmah.io/
Gelf	https://docs.graylog.org/en/2.3/pages/gelf.html
JSNLog	https://jsnlog.com/
KissLog.net	https://kisslog.net/
Log4Net	https://logging.apache.org/log4net/
NLog	https://nlog-project.org/
Loggr	https://loggr.net/
Plogger	https://www.nuget.org/packages/InvertedSoftware.PLogger.Core/
Sentry	https://sentry.io/welcome/
Serilog	https://serilog.net/
StackDriver	https://cloud.google.com/dotnet/docs/stackdriver#logging

Logging in ASP.NET Core – Default Behavior

- Logging is configured during bootstrapping
- Call to `CreateDefaultBuilder` adds the logging providers
 - Console
 - Debug
 - EventSource
 - EventLog – Windows Only
- To override default,

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
        {
            logging.ClearProviders();
            logging.AddConsole();
        })
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
```

Logging Configuration

- Provided under *Logging* section of *appsettings.json* file
- *LogLevel* property is used to specify the default logging level
- Varying Levels can be set for different categories
- Default log level is *Information*

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```

Log Levels

LogLevel	Value	Method	Description
Trace	0	LogTrace	Contain the most detailed messages. These messages may contain sensitive app data. These messages are disabled by default and should not be enabled in production.
Debug	1	LogDebug	For debugging and development. Use with caution in production due to the high volume.
Information	2	LogInformation	Tracks the general flow of the app. May have long-term value.
Warning	3	LogWarning	For abnormal or unexpected events. Typically includes errors or conditions that don't cause the app to fail.
Error	4	LogError	For errors and exceptions that cannot be handled. These messages indicate a failure in the current operation or request, not an app-wide failure.
Critical	5	LogCritical	For failures that require immediate attention. Examples: data loss scenarios, out of disk space.
None	6		Specifies that no messages should be written.

Creating Logs

- To create logs, use an *ILogger<TCategoryName>* object from DI
- Uses a log category of the full qualified name of type *Worker*
- Calls *LogInformation* method to log at the *Information* level

```
public class Worker : BackgroundService
{
    private readonly ILogger<Worker> _logger;

    public Worker(ILogger<Worker> logger) =>
        _logger = logger;

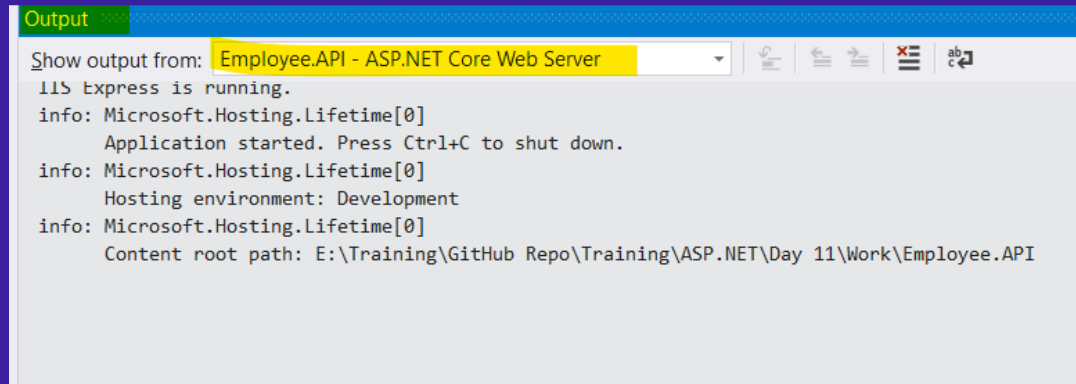
    protected override async Task ExecuteAsync(CancellationToken stop)
    {
        while (!stoppingToken.IsCancellationRequested)
        {
            _logger.LogInformation("Worker running at: {time}", DateTimeOffset.Now);
            await Task.Delay(1000, stoppingToken);
        }
    }
}
```

Applying Filtering Rules

- When an *ILogger<TCategoryName>* object is created, the *ILoggerFactory* object selects a single rule per provider to apply to that logger.
- Rule Precedence
 - Select all rules that match the provider or its alias. If no match is found, select all rules with an empty provider.
 - From the result of the preceding step, select rules with longest matching category prefix. If no match is found, select all rules that don't specify a category.
 - If multiple rules are selected, take the last one.
 - If no rules are selected, use `MinimumLevel`.

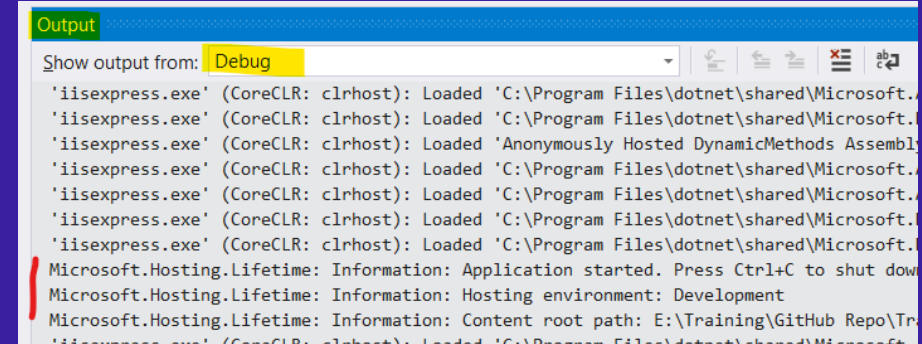
Viewing Log Output

- Visual Studio



The screenshot shows the Visual Studio Output window with the dropdown menu set to 'Employee.API - ASP.NET Core Web Server'. The output text is as follows:

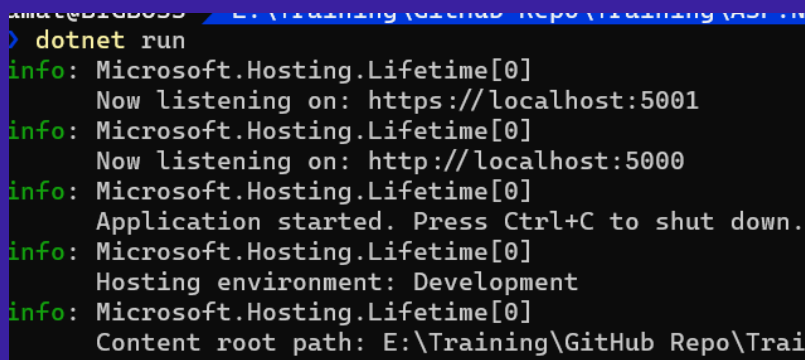
```
Show output from: Employee.API - ASP.NET Core Web Server
IIS Express is running.
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: E:\Training\GitHub Repo\Training\ASP.NET\Day 11\Work\Employee.API
```



The screenshot shows the Visual Studio Output window with the dropdown menu set to 'Debug'. The output text is as follows:

```
Show output from: Debug
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft...'
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft...'
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft...'
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft...'
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft...'
'iisexpress.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft...'
Microsoft.Hosting.Lifetime: Information: Application started. Press Ctrl+C to shut down
Microsoft.Hosting.Lifetime: Information: Hosting environment: Development
Microsoft.Hosting.Lifetime: Information: Content root path: E:\Training\GitHub Repo\Tr...
```

- Console Window



The screenshot shows a terminal window with the following output:

```
> dotnet run
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: E:\Training\GitHub Repo\Trai
```

Thanks for joining!

