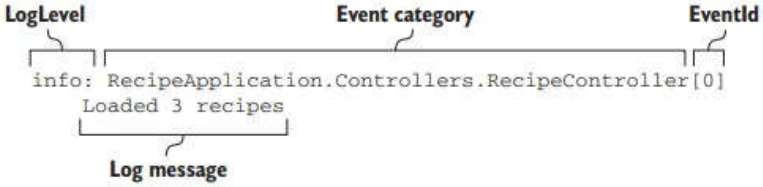
A logging provider is the component that stores or displays logs. For example, the Console log provider displays logs on the console. Similarly, the Debug log provider displays logs on the Debug window in Visual Studio.

```
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>();

  webBuilder.ConfigureAppConfiguration(config =>
                config.AddJsonFile("appsettings.json")); // Loads configuration values from
appsettings.json

          webBuilder.ConfigureLogging((ctx,config) => {
            config.AddConfiguration(
              ctx.Configuration.GetSection("Logging")); //Loads the log filtering configuration from
the Logging section

              config.AddConsole();//Adds a console provider to the app
        });

        });
  }
appsettings.json //used to filter logs by categories and loging provider
"Logging": { //// Default, all providers.
  "LogLevel": {
    "Default": "Warning",
    "Microsoft": "Information"
  }
}

  "Logging": {
  "Debug": { //for the debug loging provider use this settings
      "LogLevel": {
        "Default": "Warning",
        "EmployeeManagement.Controllers.HomeController": "Warning",
        "EmployeeManagement.Models.SQLEmployeeRepository": "Warning",//the most specific
category wins
        "Microsoft": "Warning"
      }
    }
}
```

In simple terms, LOG CATEGORY is the fully qualified name of the class that logged the message.
`Default and Microsoft are categories Default is our project namespace so all classes in our project will have Log Level specefied by the value next to Default category.`



Log level—The log level of the log is how important it is and is defined by the LogLevel enum. Event category—The category may be any string value, but it's typically set to the name of the class creating the log. For ILogger<T>, the full name of the type T is the category.
`filters are applied from current to above for example log level debug will show all logs from Debug to Critical`

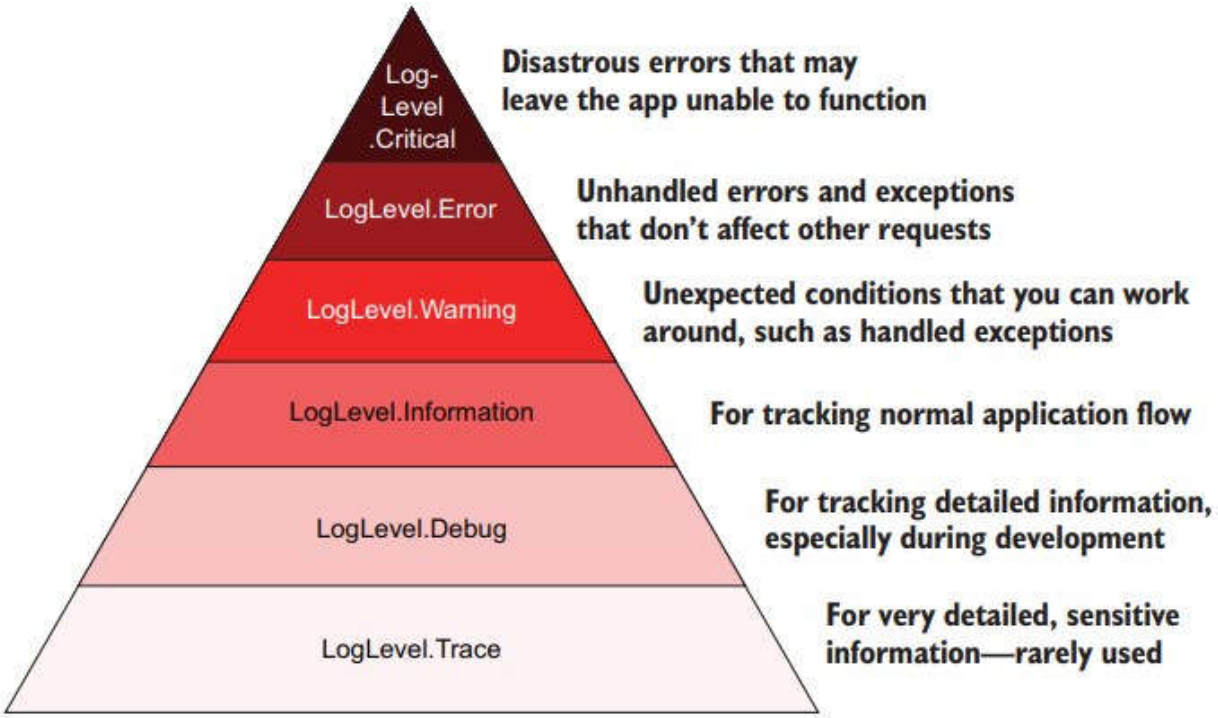


Figure 17.5 The pyramid of log levels. Logs with a level near the base of the pyramid are used more frequently but are less important. Logs with a level near the top should be rare but are important.

```
        builder.ConfigureAppConfiguration((hostingContext, config) =>
            {
                var env = hostingContext.HostingEnvironment;

        config.AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
                        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
true, reloadOnChange: true);

                if (env.IsDevelopment())
                {
                    var appAssembly = Assembly.Load(new
AssemblyName(env.ApplicationName));
                    if (appAssembly != null)
                    {
                        config.AddUserSecrets(appAssembly, optional: true);
                    }
}

config.AddEnvironmentVariables();

                if (args != null)
                {
                    config.AddCommandLine(args);
                }
            })
            .ConfigureLogging((hostingContext, logging) =>
            {
    logging.AddConfiguration(hostingContext.Configuration.GetSection("Logging"));
    logging.AddConsole();
    logging.AddDebug();
    logging.AddEventSourceLogger();
}).


        //.ConfigureWebHost(WebHostBuilder =>
        //    {
        //        WebHostBuilder.UseStartup<Startup>();
        //    });

        //.ConfigureLogging((ctx, builder) =>
        // {
        //    builder.AddConfiguration(
        //    ctx.Configuration.GetSection("Logging"));
        //    builder.AddConsole();
        //    builder.ClearProviders();
       // addDebug e show output from Debug
        //});
```