Working with Configuration Providers

Steve Gordon

.NET Engineer and Microsoft MVP

@stevejgordon | www.stevejgordon.co.uk

Overview

Configuration providers

How configuration is populated

Configuring with environment variables

Configuring with command line arguments

Securing configuration secrets

- Development - User secrets
- Production – Azure Key Vault

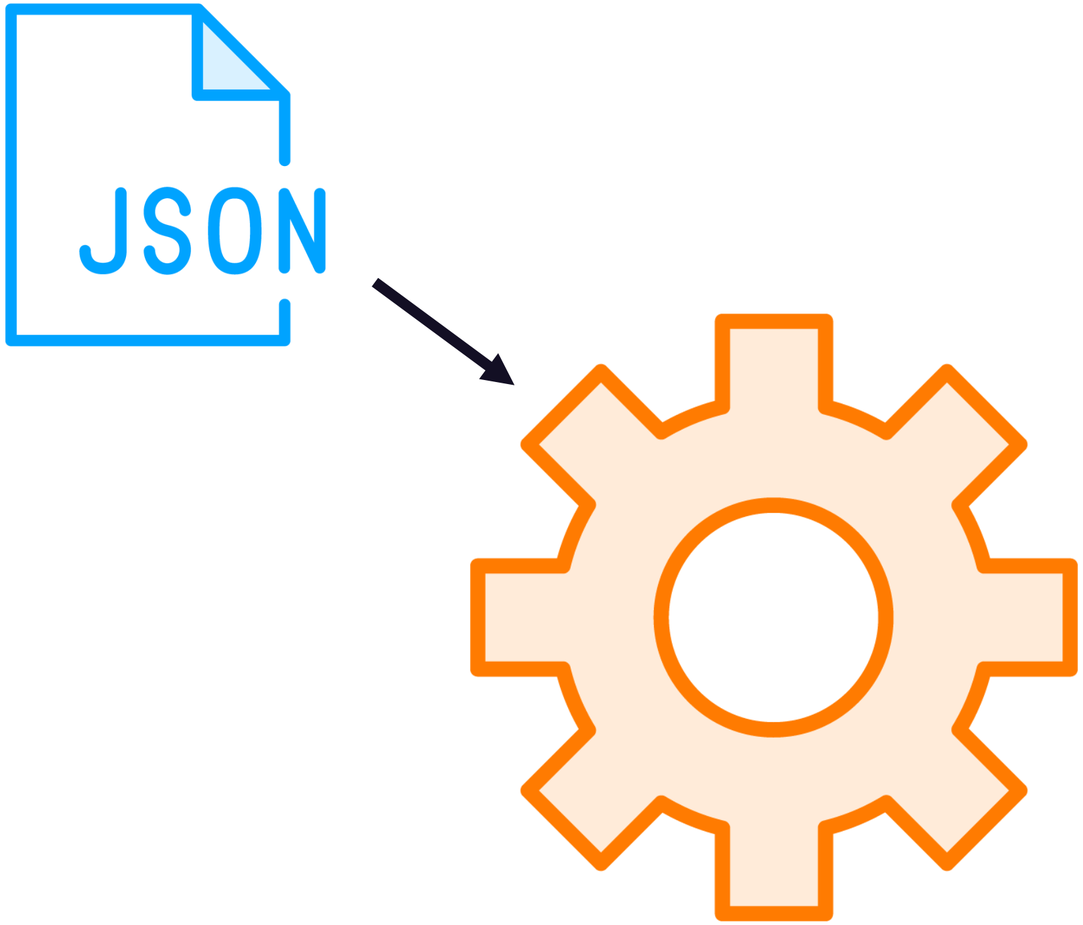
Configuration from AWS Parameter Store

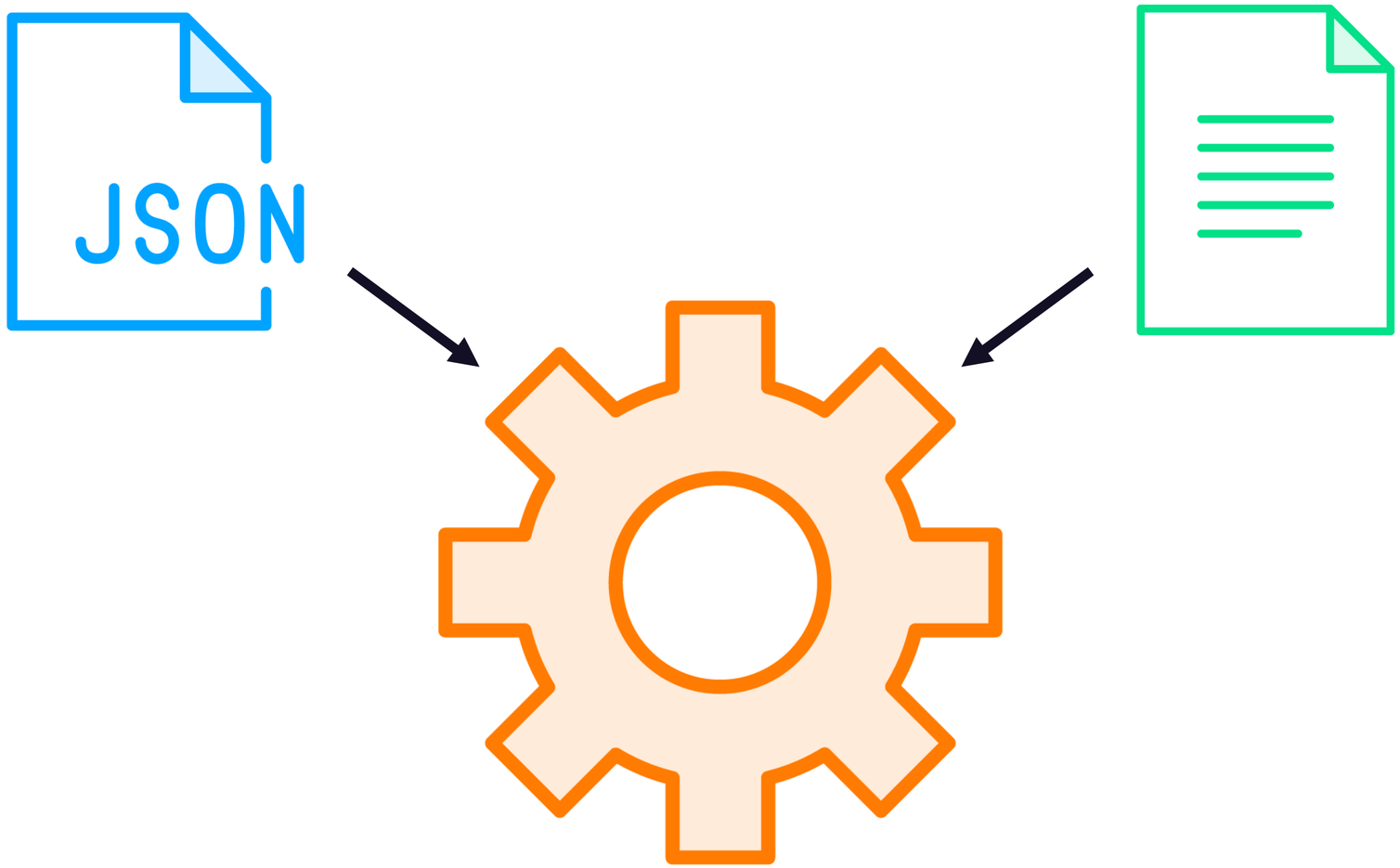
Controlling configuration provider order

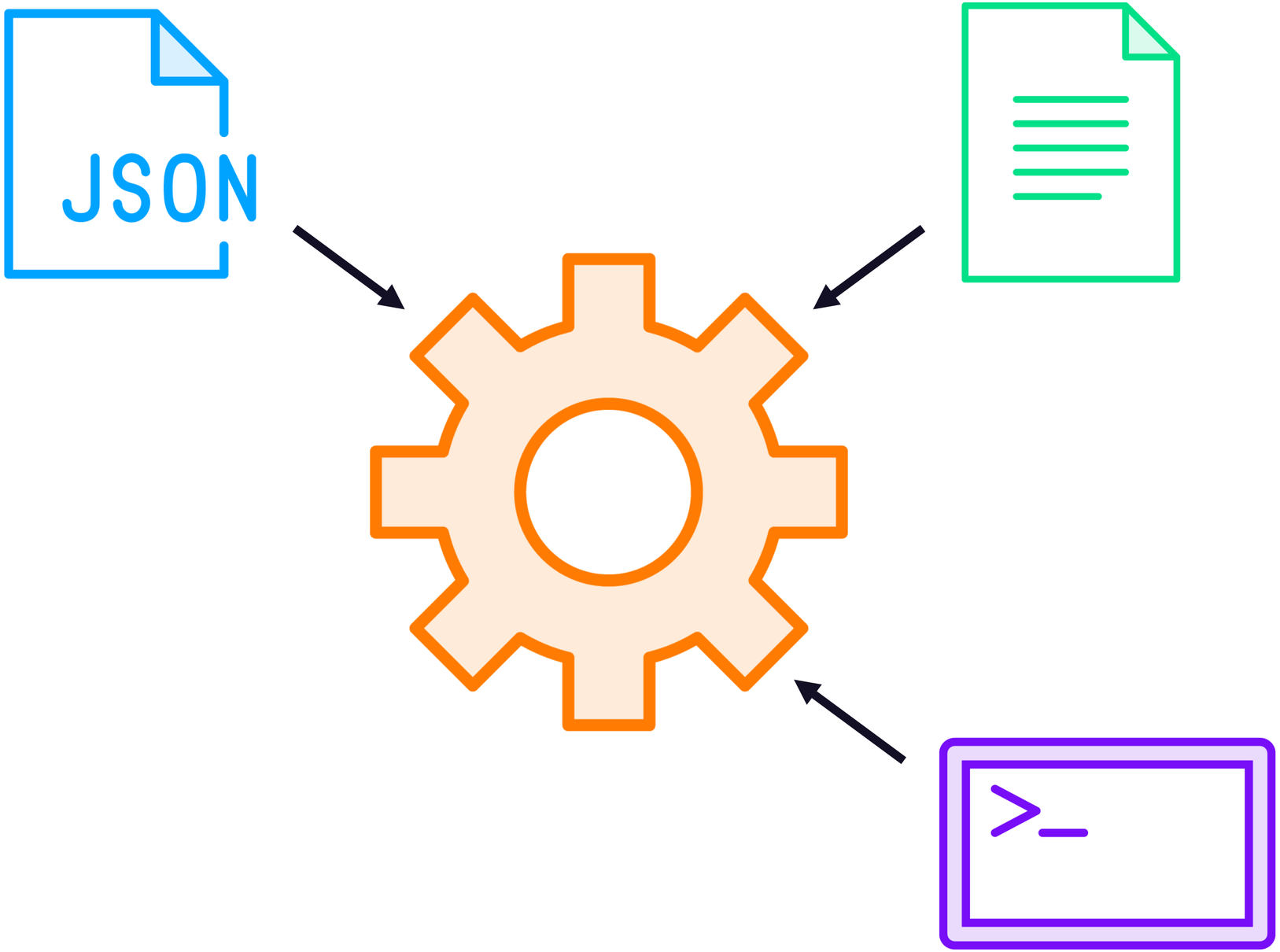
Defining a custom configuration provider

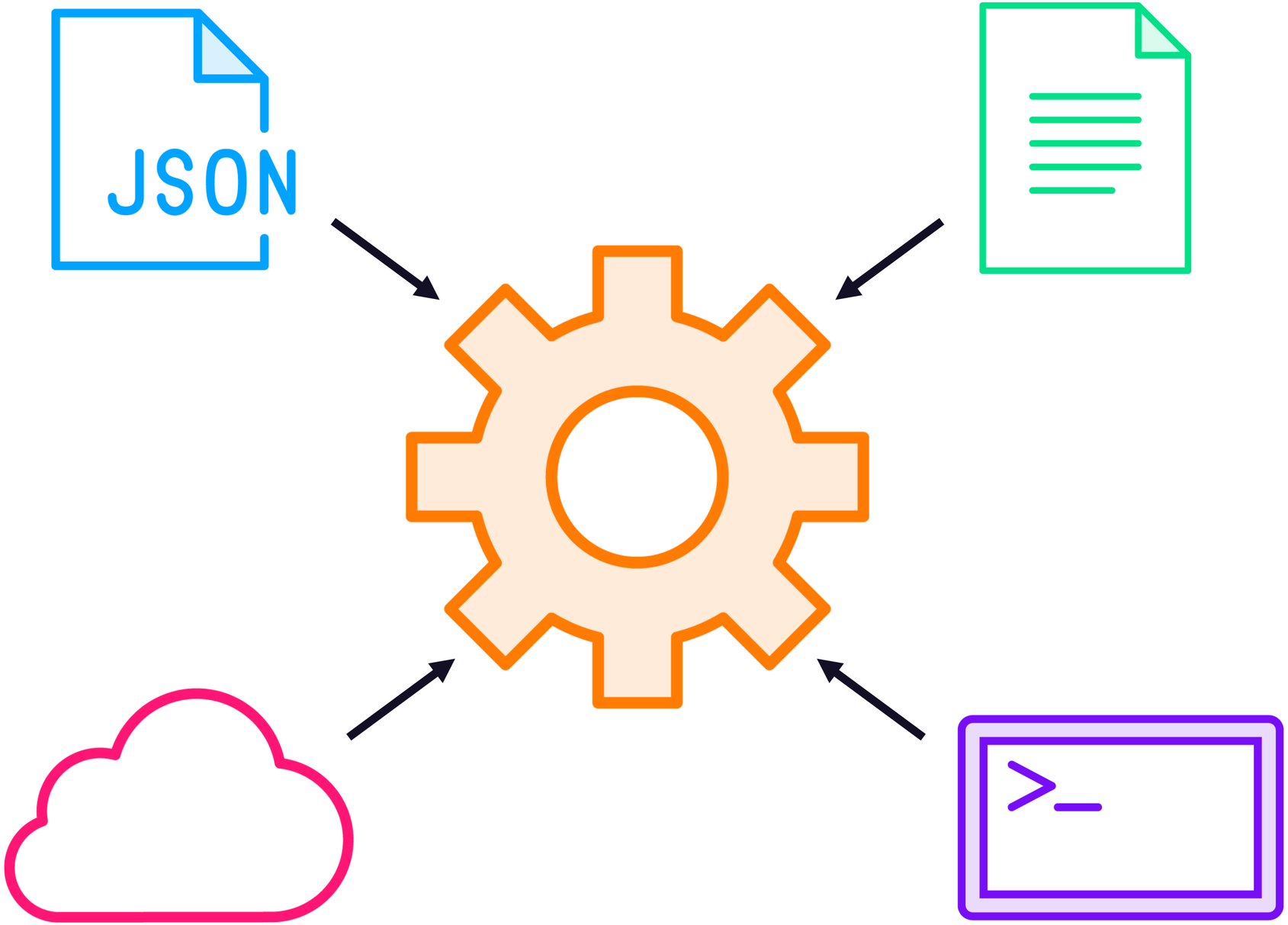
Debugging configuration

Configuration Providers

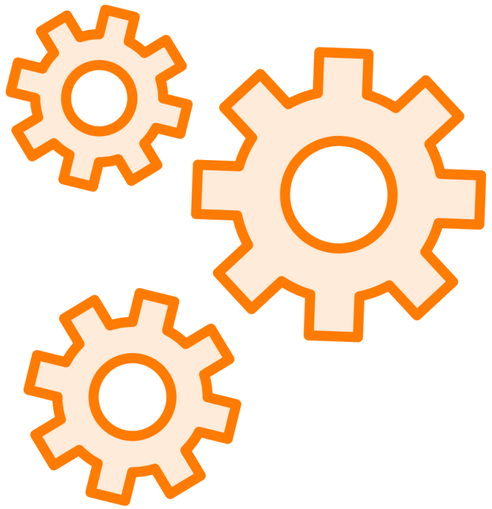

JSON


JSON

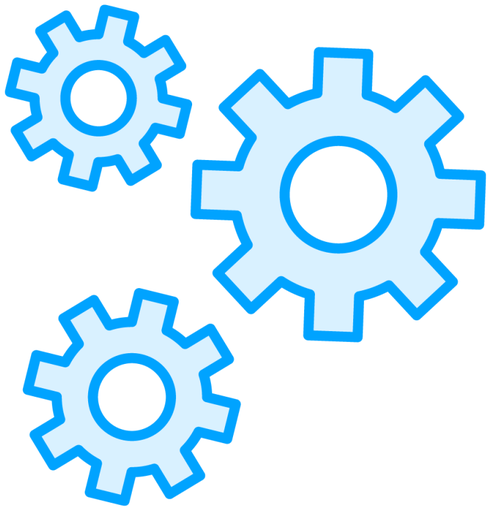

JSON
>_


JSON
>_

Where Configuration Applies



Host configuration



Application configuration

Configuration Providers

Executed in order to load configuration

Key/values pairs are added or replaced by each configuration provider

The order of the configuration providers affects the final configuration values

A set of default providers are added by the host builder

Aside...

IWebHostBuilder.CreateDefaultBuilder(string[] args)

```
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() && !string.IsNullOrEmpty(env.ApplicationName))
      {
            var appAssembly = Assembly.Load(new AssemblyName(env.ApplicationName));
            if (appAssembly != null)
            {
                  config.AddUserSecrets(appAssembly, optional: true);
            }
      }
      config.AddEnvironmentVariables();
      if (args != null)
      {
            config.AddCommandLine(args);
      }
})
```

Default Application Configuration Source Order

Sources can override configuration entries from prior sources

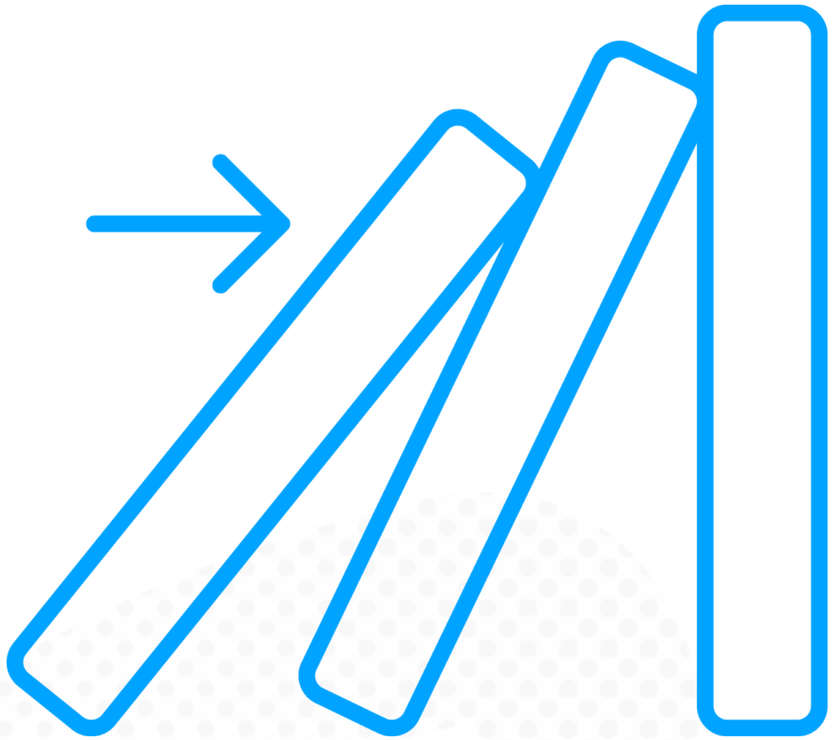
JSON (appSettings.json)

JSON (appSettings.Development.json)

User secrets (in development)

Environment variables

Command line arguments



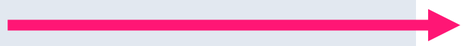
How Configuration is Populated

appSettings.json

```
{
  "HomePage": {
    "Title": "A Good Title",
    "ShowGallery": true
  },
  "MaxNewsItems": 10
}
```

appSettings.json

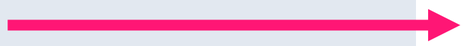
```
{
  "HomePage": {
    "Title": "A Good Title",
    "ShowGallery": true
  },
  "MaxNewsItems": 10
}
```



HomePage:Title = “A Good Title”

appSettings.json

```
{
  "HomePage": {
    "Title": "A Good Title",
    "ShowGallery": true
  },
  "MaxNewsItems": 10
}
```

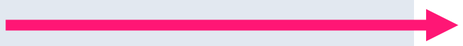


HomePage:Title = “A Good Title”

HomePage:ShowGallery = true

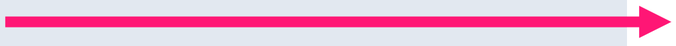
appSettings.json

```
{
  "HomePage": {
    "Title": "A Good Title",
    "ShowGallery": true
  },
  "MaxNewsItems": 10
}
```



HomePage:Title = “A Good Title”

HomePage:ShowGallery = true



MaxNewsItems = 10

appSettings.Production.json

```
{
  "HomePage": {
    "Title": "A Better Title",


  },
  "MaxNewsItems": 10
  "HideAds“: true
}
```

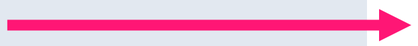
HomePage:Title = “A Good Title”

HomePage:ShowGallery = true

MaxNewsItems = 10

appSettings.Production.json

```
{
  "HomePage": {
    "Title": "A Better Title",


  },
  "MaxNewsItems": 10
  "HideAds“: true
}
```



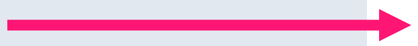
HomePage:Title = “A Better Title”

HomePage:ShowGallery = true

MaxNewsItems = 10

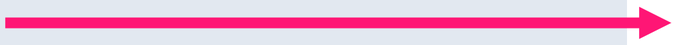
appSettings.Production.json

```
{
  "HomePage": {
    "Title": "A Better Title",


  },
  "MaxNewsItems": 10
  "HideAds“: true
}
```



HomePage:Title = “A Better Title”

HomePage:ShowGallery = true



MaxNewsItems = 10

appSettings.Production.json

```
{
  "HomePage": {
    "Title": "A Better Title",


  },
  "MaxNewsItems": 10
  "HideAds“: true
}
```

HomePage:Title = “A Better Title”

HomePage:ShowGallery = true

MaxNewsItems = 10

HideAds = true

Environment Variables

HomePage__ShowGallery = true

HideAds = false

HomePage:Title = “A Better Title”

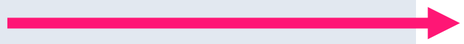
HomePage:ShowGallery = true

MaxNewsItems = 10

HideAds = true

Environment Variables

```
HomePage__ShowGallery = true

HideAds = false
```



HomePage:Title = “A Better Title”

HomePage:ShowGallery = true

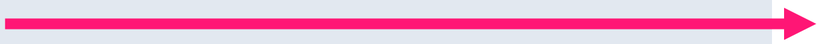
MaxNewsItems = 10

HideAds = true

Environment Variables

HomePage__ShowGallery = true

HideAds = false



HomePage:Title = “A Better Title”

HomePage:ShowGallery = true

MaxNewsItems = 10

HideAds = false

Define configuration using environment variables

Define environment variables using launchSettings.json

Default Application Configuration Source Order

Sources can override configuration entries from prior sources

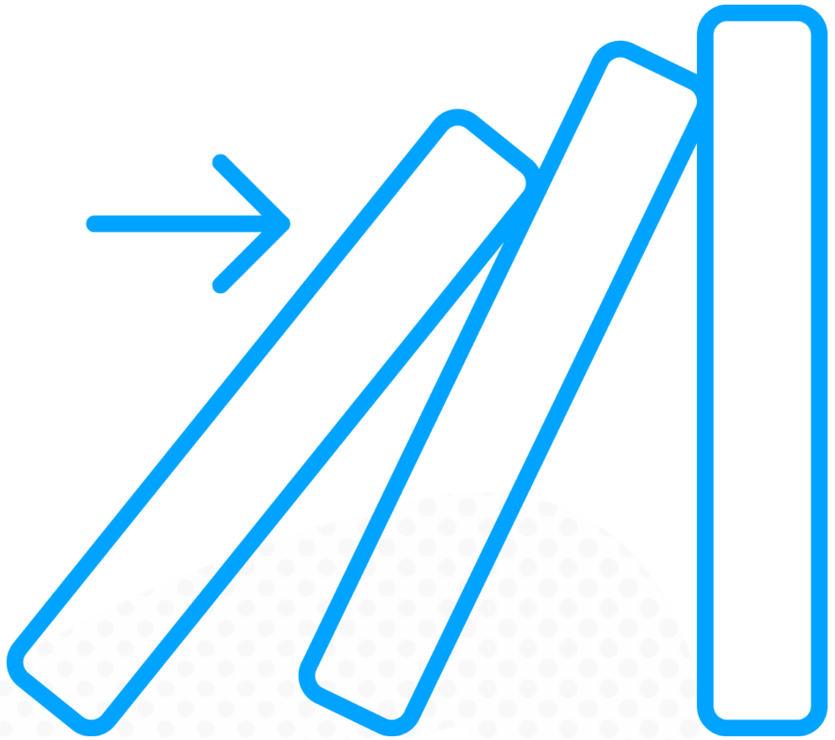
JSON (appSettings.json)

JSON (appSettings.Development.json)

User secrets (in development)

Environment variables

Command line arguments



Environment variables are a cross-platform and container-compatible way to provide configuration.

Pass configuration as command line arguments

Command Line Arguments

```
dotnet run Features:Greeting:GreetingColour="#000000"

dotnet run /Features:Greeting:GreetingColour=#000000
```

Don’t mix and match formatting styles for command line arguments.

Securing Sensitive Data in Configuration

```
{
  "HomePage": {
    "Title": "A Good Title",
    "ShowGallery": true
  },
  "MaxNewsItems": 10,
  "UsersApiKey" : "SUPERSECRET"
}
```

◀This title is not sensitive

```
{
  "HomePage": {
    "Title": "A Good Title",
    "ShowGallery": true
  },
  "MaxNewsItems": 10,
  "UsersApiKey" : "SUPERSECRET"
}
```

◀This title is not sensitive

◀This poses a security risk

Developer Responsibility

Security is always a priority

Consider the sensitivity of configuration values if exposed

Avoid exposing sensitive data within insecure sources

Code reviews should highlight security risks

A true story!

Keep your secrets, secret!

Secure secrets in development

- User secrets CLI
- Managing user secrets

User secrets help protect non-production secrets, during development.

Secure secrets in production

- Azure Key Vault

I have authenticated to Azure using the Azure CLI.

How to authenticate .NET apps to Azure services using the .NET Azure SDK

learn.microsoft.com/dotnet/azure/sdk/authentication

Load configuration from AWS

- Define configuration using Systems Manager: Parameter Store
- Add the AWS Parameter Store configuration provider

Configure AWS credentials

docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-config-creds

Customize the order of configuration providers

- Clear the default providers
- Register providers using IOptionsBuilder

IWebHostBuilder.CreateDefaultBuilder(string[] args)

```
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() && !string.IsNullOrEmpty(env.ApplicationName))
      {
            var appAssembly = Assembly.Load(new AssemblyName(env.ApplicationName));
            if (appAssembly != null)
            {
                  config.AddUserSecrets(appAssembly, optional: true);
            }
      }
      config.AddEnvironmentVariables();
      if (args != null)
      {
            config.AddCommandLine(args);
      }
})
```

Load configuration from a database

Define a configuration provider

Define a configuration source

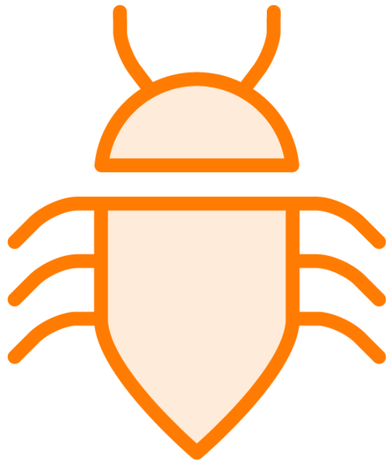
Register the custom provider with the IConfigurationBuilder

Registering Directly on the ConfigurationManager

```
var builder = WebApplication.CreateBuilder(args);

using var connection = new
    SqliteConnection(builder.Configuration
        .GetConnectionString("SqliteConnection"));

await connection.OpenAsync();

...

builder.Configuration.AddEfConfiguration(o =>
    o.UseSqlite(connection));
```

Debug configuration during development

Debugging Configuration



During development it can be useful to view the final configuration for an application

Debugging allows verification of which configuration values have been applied

ConfigurationRootExtensions

Generates a human-readable view of the configuration showing where each value came from.

```
public static string GetDebugView (this IConfigurationRoot root);
```

Summary

Default configuration providers

Configuring applications

- Environment variables
- Command line arguments

Securing configuration

- User Secrets
- Azure Key Vault

Loading configuration from AWS

Configuration loading and order

Customizing configuration providers

Defining a custom configuration provider

Debugging configuration

Congratulations!

Steve Gordon

.NET Engineer and Microsoft MVP

@stevejgordon | www.stevejgordon.co.uk