.NET Developer Toolkit

# USER SECRETS

## What we’ll cover

- Why User Secrets?
- Configuration in .NET
- Implementing User Secrets

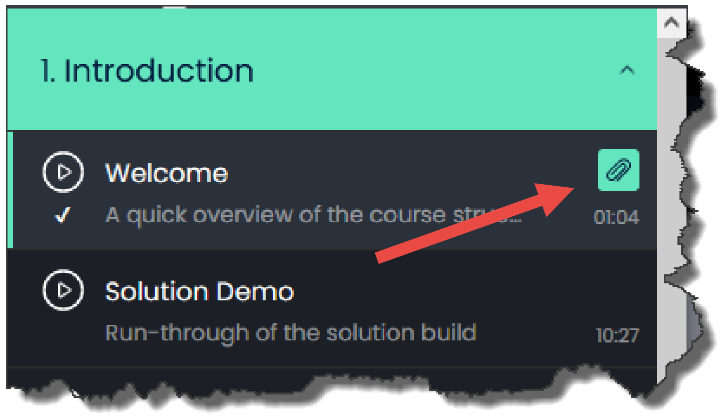
## Ingredients

- .NET 6 SDK (free)
- VS Code (free)

## Prerequisites

- Dependency Injection [Optional]

# Course Downloads



1. Introduction

Welcome

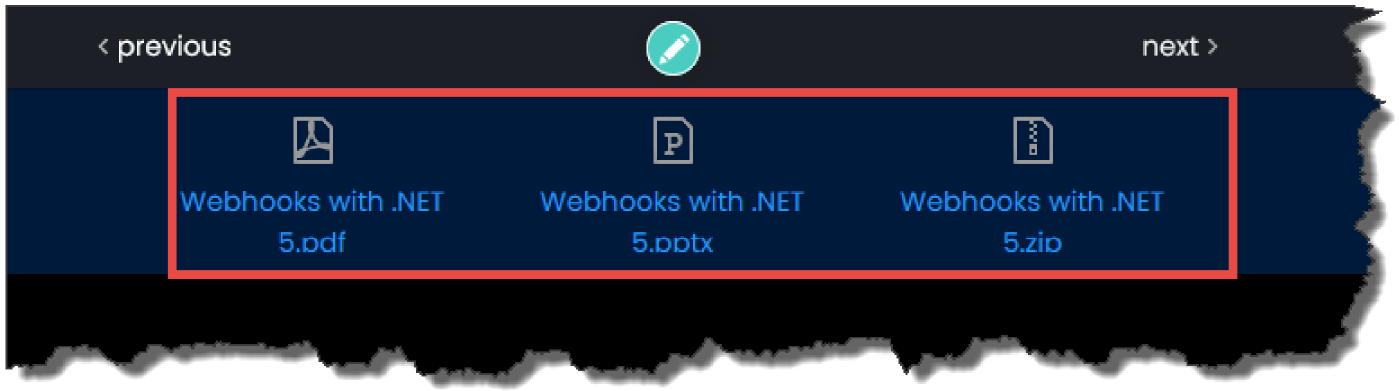
A quick overview of the course stru...

01:04

Solution Demo

Run-through of the solution build

10:27



< previous

next >

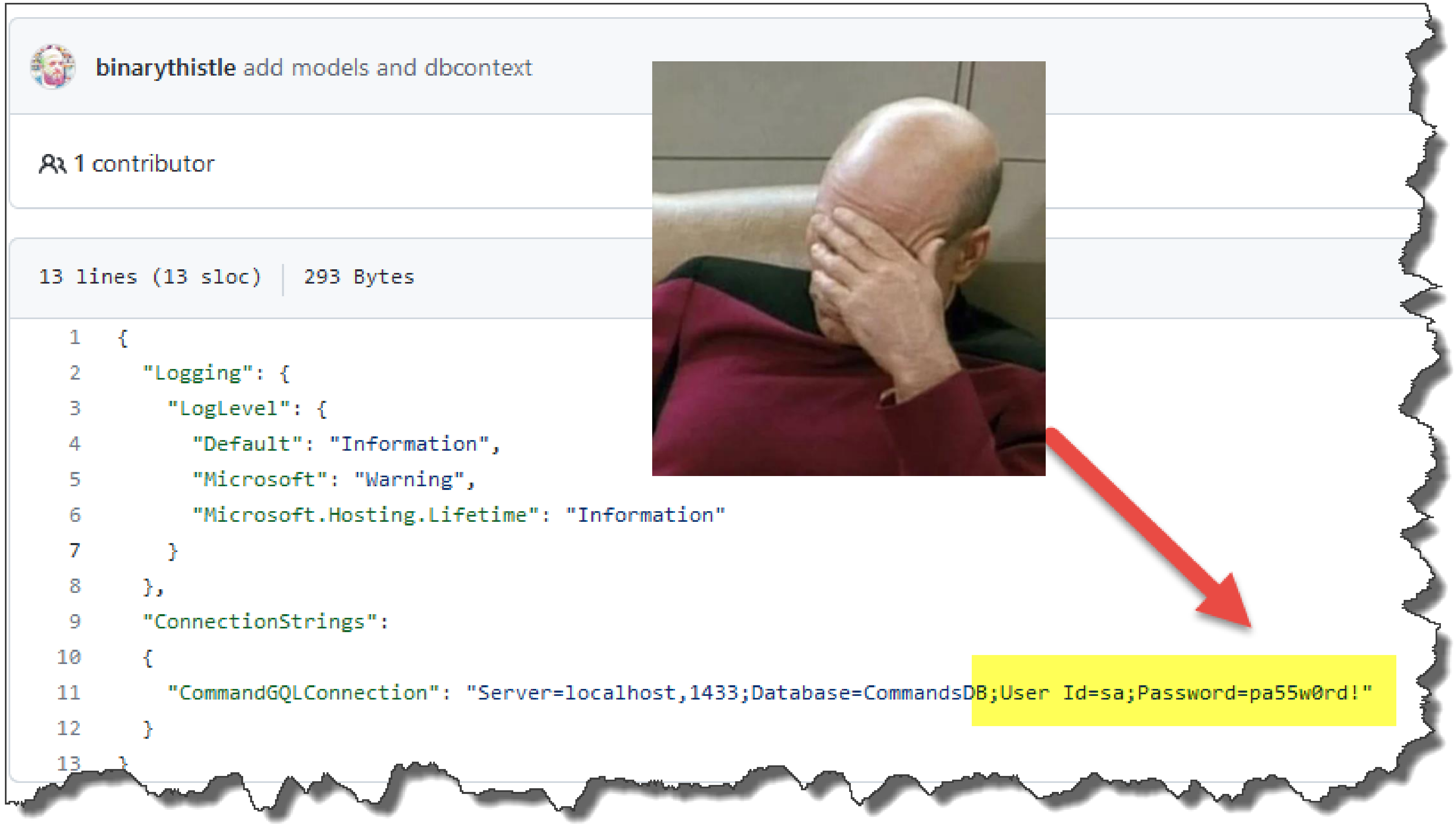
Webhooks with .NET 5.pdf

Webhooks with .NET 5.pptx

Webhooks with .NET 5.zip

## Why User Secrets?

# We (should) all have secrets…

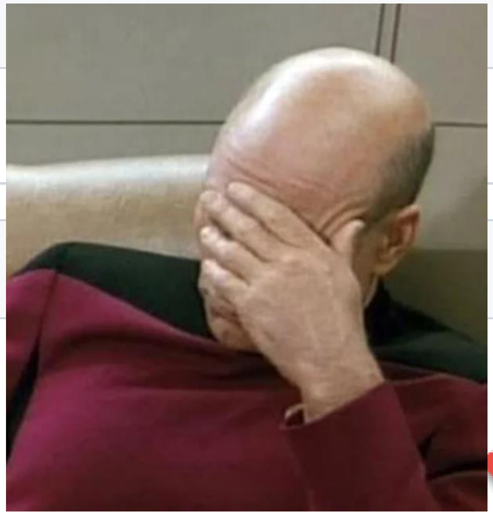


binarythistle add models and dbcontext

1 contributor

13 lines (13 sloc) | 293 Bytes

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "ConnectionStrings":
  {
    "CommandGQLConnection": "Server=localhost,1433;Database=CommandsDB;User Id=sa;Password=pa55w0rd!"
  }
}
```



# For “real” projects this is not a good idea…

- Sensitive information should not be *exposed.*
- *Separated* it from the code
- Accessed through a *controlled means* no matter the environment:
    - Production
    - Test
    - Staging
    - Development

## Secrets in Development

- Environment Variables
  - Avoids the use of config files
  - Still stored in plain text
  - Should be scoped to the user or app (not the “machine”)
- User Secrets (Secret Manager)
  - Secrets are managed, (how they are stored is transparent to the user)
  - Scoped to the Project or Projects
  - Stored as config files (in plain text)
  - Stored on the local machine using the users profile folder

## Configuration in .NET

# Configuration in .NET

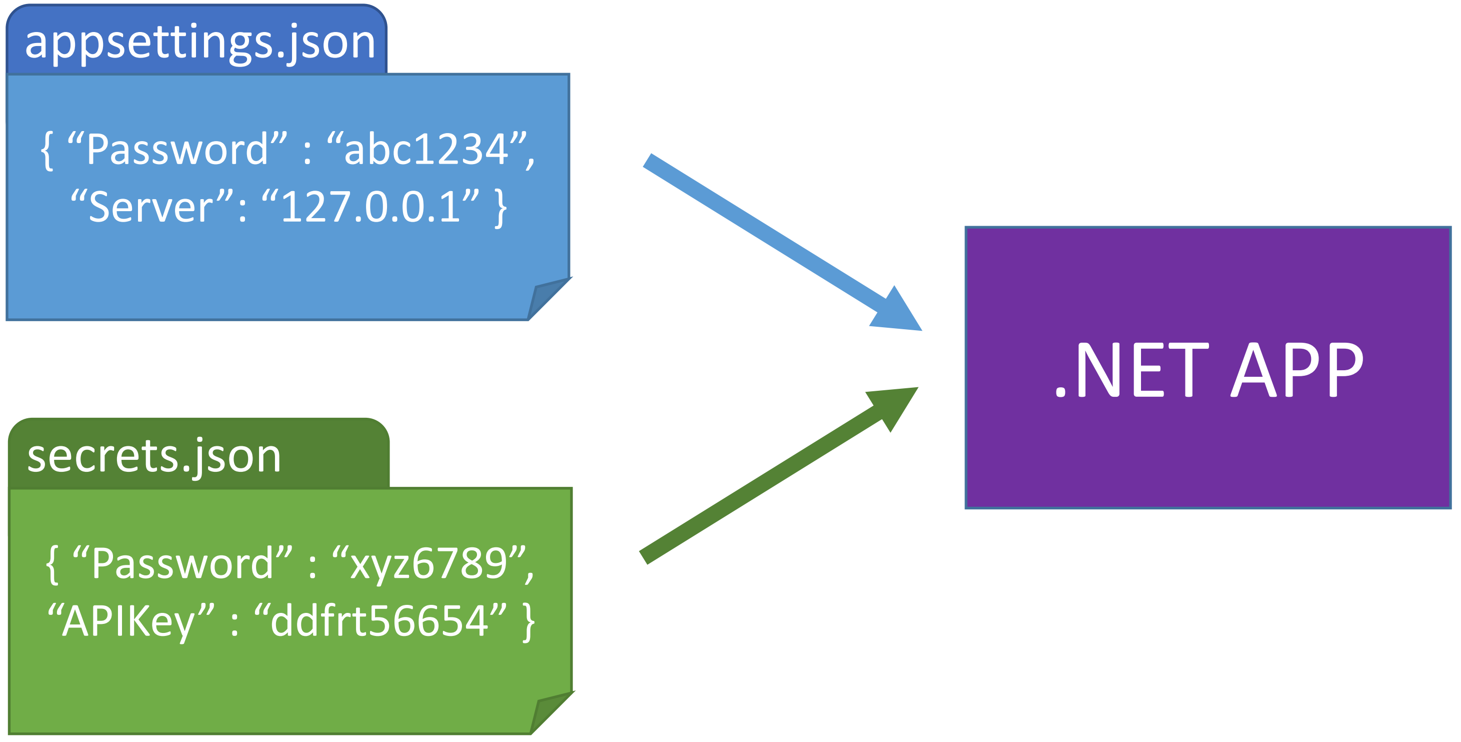
- Configuration performed using 1 or more *configuration providers*
- These provides read key-values pairs from *configuration sources*
- Most .NET projects have a (default) configuration set up
  - Console apps do not so you'd have to manually configure
- Configuration sources have a default order of precedence
  - This can be configured though

## Default Hierarchy

| Order of Read* | Configuration Source | Configuration Provider |
|---|---|---|
| 1 | appsettings.json | File Configuration Provider |
| 2 | appsettings.<Environment>.json | File Configuration Provider |
| 3 | User Secrets *(only in Development)* | App Secrets (Secret Manager) |
| 4 | Environment Variables | Environment Variables |
| 5 | Command Line | Command-line configuration provider |

* If the same key exists in a later source, it will over-write the previous one: last loaded key wins.

## Example


appsettings.json
{ “Password” : “abc1234”,
“Server”: “127.0.0.1” }
secrets.json
{ “Password” : “xyz6789”,
“APIKey” : “ddfrt56654” }
.NET APP

“Password” : “xyz6789”
“Server” : “127.0.0.1”
“APIKey” : “ddfrt56654”

## Let’s Code!