Configuration

.NET FUNDAMENTALS

Agenda

- What is configuration?
- .NET Configuration
- Default Configuration
- Providers
- Summary/Next Steps
- Lab/Demo

What is configuration?

- Configuration files give developers and administrators control and flexibility over the way applications run
- Config files allow application settings to be set/changed at run-time
- Settings in configuration files eliminate the need to recompile an application every time a setting changes
- “Build once, deploy many” is accomplished by leveraging config files

.NET Configuration

- Microsoft Documentation: NET Fundamentals - Configuration
- Default Configuration – can be overridden
- Access hierarchical configuration data - use colon “:”
- A variety of configuration providers

Default Configuration

- In Program.cs, `CreateDefaultBuilder` provides default configuration (ordered):
  - ChainedConfigurationProvider (also adds IConfiguration to dependency injection)
  - appsetting.json (if exists)
  - appsettting.{*environment*}.json – if environment is not set, Production is assumed.
  - App/User Secrets (secrets.json) – used locally when environment is set to Development
  - Environment variables (provider)
  - Command-line arguments (provider)

Providers

- File: JSON, XML, INI
- User Secrets
- In memory: i.e. Dictionary
- Environment Variables
- Command Line
- Azure App Configuration
- Azure Key Vault
- Custom

Summary/Next Steps

- Create and use config files
  - Avoid hard-coding values
  - Avoid manual changes by administrators
- Use Distinct config files for each environment
- Avoid using appsettings.json
- Avoid “stacking” config files; i.e. appsettings.json + appsettings.{env}.json
- Do not store username/password combinations or security keys in config files that are checked into source control.

Lab/Demo

Source Code Available at

https://github.com/bradthecoder/Samples.Configuration