.NET-Anwendungen konfigurieren

Wege und Möglichkeiten

Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor

Microsoft® SQL Server®

.NET Core

ASP.NET Core

Azure Cosmos DB

DOTNETCONSULTING

Mein Service- Ihr Benefit

- Individuelle Inhouse Trainings
- (Online on-demand) Projektbegleitung
- Beratung
  - Problemanalyse und Lösungen
  - Technologieentscheidungen

Agenda

- Provider
- Quellen
- IOptions-Pattern
- Validierung
- User Secrets
- Custom Provider


pienene
the latvian green studio
pienene
latvian green studio
P
Tikai
ar atļaujām
Nr. 33 - 36
Latviešu
dizaina veikals
& Kafetērija
pienene
CAFÉ SHOP
KAFE VEIKALS
Latvian
design store
& Cafe
44
Fabisiak

Provider

Konfigurationsprovider

Provider	NuGet-Paket
JSON-Datei	`Microsoft.Extensions.Configuration.Json`
XML-Datei	`Microsoft.Extensions.Configuration.Xml`
INI-Datei	`Microsoft.Extensions.Configuration.Ini`
Umgebungsvariablen	`Microsoft.Extensions.Configuration.EnvironmentVariables`
Programmargumente	`Microsoft.Extensions.Configuration.CommandLine`
In-Memory-Collection	–
Key-per-File	`Microsoft.Extensions.Configuration.KeyPerFile`
Azure Key Vault	`Azure.Extensions.AspNetCore.Configuration.Secrets`
User Secrets (nur ASP.NET Core)	`Microsoft.Extensions.Configuration.UserSecrets`
Custom	–

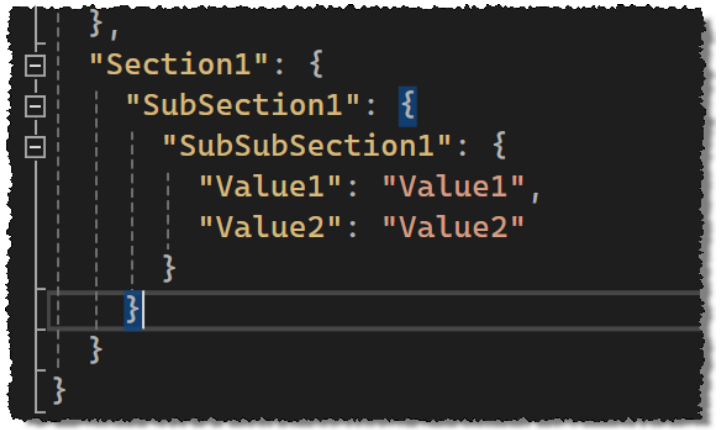
```
Install-Package Microsoft.Extensions.Configuration.Json (s.o.)
```

Quellen & Werte

Quellen & Werte

- Provider können mehrfach verwendet werden
- Auf die Reihenfolge kommt es an!
- Alle Quellen werden der Reihe nach verarbeitet
  - Jeder Wert wird unter einem Schlüssel gespeichert
    - `Section1:SubSection1:SubSubSection1:Value1`
  - Bereits vorhandene Werte mit gleichen Schlüssel werden überschrieben

Unabhängig von der Quelle!



Quellen – ASP.NET Core

- Environment variables (Prefix “ASPNETCORE_”)
- Environment variables (Prefix “DOTNET_”)
- appsettings.json
- appsettings.{Environment}.json
- Secret Manager (Development environment)
- Environment variables (kein Prefix)
- Command-line arguments

https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/

Quellen – GenericHost

- Environment variables (Prefix “DOTNET_”)
- appsettings.json
- appsettings.{Environment}.json
- Secret Manager (Development environment)
- Environment variables (kein Prefix)
- Command-line arguments

https://learn.microsoft.com/en-us/dotnet/core/extensions/generic-host?tabs=appbuilder

Demo

IOptions-Pattern

IOptions, IOptionsSnapshot & IOptionsMonitor

`IOptions<TOptions>`

- Konfigurationsdaten lesen, nachdem die App gestartet wurde

`IOptionsSnapshot<TOptions>`

- Ist in Szenarios nützlich, in denen Optionen bei jeder Anforderung neu berechnet werden sollten..

`IOptionsMonitor<TOptions>`

- Wird verwendet, um Optionen abzurufen und Benachrichtigungen über Optionen für TOptions-Instanzen zu verwalten
- Unterstützt:
  - Änderungsbenachrichtigungen
  - Benannte Optionen
  - Erneut ladbare Konfiguration
  - Selektive Optionsvalidierung (IOptionsMonitorCache<TOptions>)

Konfiguration via DI zur Verfügung stellen

IOptions<>-Pattern

```
private static void ConfigureServices(…)
{
    ...
    IConfigurationSection doSomeThingJobConfigurationSection =
        hostBuilderContext.Configuration.GetSection("Jobs:DoSomeThinkJob");
    services.Configure<DoSomeThinkJobSettings>(doSomeThingJobConfigurationSection);
    ...
}


public DoSomeThingJob(IOptions<DoSomeThinkJobSettings> doSomeThingJobSettings) // Konstruktor
{
    _doSomeThingJobSettings = doSomeThingJobSettings.Value;
}
```

Demo

Validierung

Validierung

- Attribute
- IValidateOptions<>-Pattern

- Validierung bei App-Start oder Zugriff auf Konfiguration

Demo

User Secrets

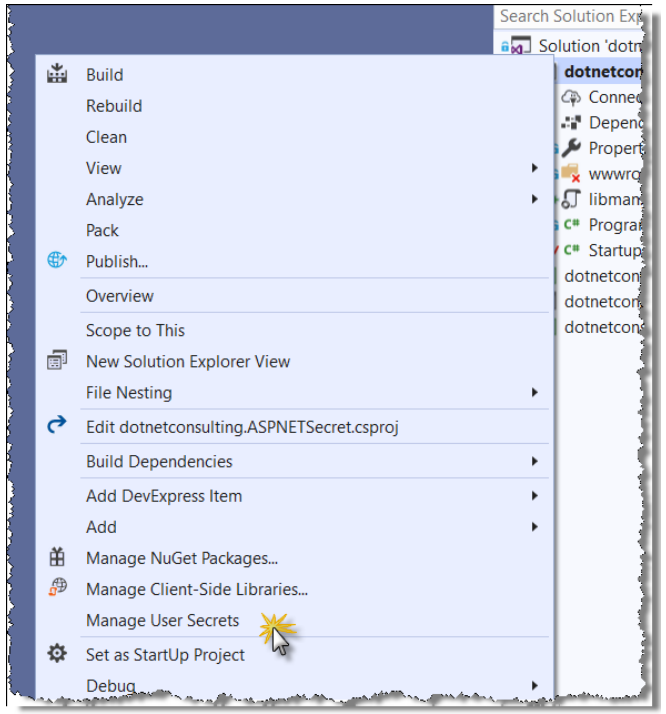
ASP.NET User Secrets

Sensible Einstellungen auslagern

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
  <UserSecretsId>576504d1-3c17-4fc0-b0a0-23ee841be7e6</UserSecretsId>
  </PropertyGroup>
</Project>
```

Betriebssystem	Speicherort
Windows	`%APPDATA%\microsoft\UserSecrets\<userSecretsId>\secrets.json`
Linux	`~/.microsoft/usersecrets/<userSecretsId>/secrets.json`
MacOS	`~/.microsoft/usersecrets/<userSecretsId>/secrets.json`

User Secrets verwalten



Aktion	.NET Core CLI
User Secret setzen	`dotnet user-secrets set Geheimnis "My Name is Bond"`
User Secrets auflisten	`dotnet user-secrets list`
User Secret löschen	`dotnet user-secrets remove Geheimnis`
Alle User Secrets löschen	`dotnet user-secrets clear`

Demo

Custom Provider

Custom Provider

- Eigener File Provider
- Eigener Custom Provider

Eigener File Provider

Code-Klasse	Aufgabe
`FileConfigurationProvider`	Konfigurationprovider für Datei basierte Quellen
`FileConfigurationSource`	Konfigurationsquellen für Datei basierte Quellen

Demo

Eigenen Custom Provider schreiben

Code-Klasse	Aufgabe
`SqlConfigurationExtensions`	Erweiterung der Fluent API
`SqlDatabaseConfigurationSource`	Implementiert `IConfigurationSource`, um eine Konfiguration mittels Fluent API zu ermöglichen.
`SqlDatabaseConfigurationProvider`	Implementiert `IConfigurationProvider` und stellt die eigentliche Programmlogik dar, die die Konfigurationswerte bereitstellt.
`SqlDatabaseChangeToken`	Implementiert `IChangeToken` und bietet die Möglichkeit, Veränderungen bei den Konfigurationswerten zu signalisieren. Hier nicht weiter vertieft, da die Änderungen zwischen dem Lesen zwei zusammenhängender Werte auftreten können.

Demo

Fragen? Jetzt oder später!

Kontakt

E-Mail
tkansy@dotnetconsulting.eu

Telefon
+49 (0) 6187 / 2009090

Microsoft Teams
Meet now

LinkedIn
Link me

XING
Xing me

X (Twitter)
@tkansy

DOTNETCONSULTING

www.dotnetconsulting.eu

SQL Server meets .NET (Core)- professionally!

Ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen mit Microsoft SQL Server- mit Allem, was dazu gehört- und was man vielleicht weglassen sollte.

DOTNETCONSULTING