Programmierung

- Docker
  - Quick And Dirty
- Powershell
- EntityFramework
  - Link-Sammlung
  - Installation
- .Net Upgrade-Assistant
  - Installation
- Zertifikate
  - .Net 6 - https
  - openssl - selfsign - certificate
- T-SQL
  - Datenbankfelder auflisten
- Blazor WASM
  - Update
- Bridge To Kubernetes
  - Debug - launchSettings.json

Docker

Docker

Quick And Dirty

Visual Studio-Containertools unter Windows - Visual Studio | Microsoft Docs

https://docs.microsoft.com/de-de/visualstudio/containers/overview?view=vs-2019

https://melcher.dev/2019/01/til-set-docker-environment-variables-in-visual-studio-for-easier-debugging/

https://docs.microsoft.com/en-us/learn/modules/use-docker-container-dev-env-vs-code/

docker run -it --rm --entrypoint "cmd.exe" counter-image

.NET Install scripts

https://dotnet.microsoft.com/download/dotnet/scripts/v1/dotnet-install.sh

https://dotnet.microsoft.com/download/dotnet/scripts/v1/dotnet-install.ps1

https://docs.microsoft.com/de-de/dotnet/core/install/linux-ubuntu

Ubuntu

wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb

sudo apt update
sudo apt install apt-transport-https
sudo apt install dotnet-sdk-3.1

Powershell

Datenbank - Tools

[offline installs of dbatools – dbatools]

Get-SqlLogin -ServerInstance $serverName
#Get-SqlLogin -ServerInstance "MyServerInstance" -LoginName "Login01" | Remove-SqlLogin
#Add-SqlLogin -ServerInstance "MyServerInstance" -LoginName "MyLogin" -LoginType "SqlLogin" -DefaultDatabase "OtherDatabase"

EntityFramework

Link-Sammlung

http://bekenty.com/use-sqlite-in-net-core-3-with-entity-framework-core/

Installation

Install Entity Framework Core

When working with EF Core, you will want to install the correct package for the Entity Framework Core database provider you want to target in your project. Since this demo project is regarding SQLite, I am going to install the EF Core that supports SQLite.

Add NuGet package reference

```
Install-Package Microsoft.EntityFrameworkCore.Sqlite
```

A new file will be created in the project with the following content

Klassen erstellen

```
Add-Migration InitialCreate
```

```
Update-Database
```

Add-Migration <NAME> //Sample: AddExecuteCounterCreate

Update-Database

.Net Upgrade-Assistant

Installation

dotnet tool install -g --add-source 'https://api.nuget.org/v3/index.json' --ignore-failed-sources upgrade-assistant

https://dotnet.microsoft.com/en-us/platform/upgrade-assistant/tutorial/install-upgrade-assistant

Zertifikate

Zertifikate

.Net 6 - https

Problembehebung beim Starten mit https

1. dotnet dev-certs https --clean
2. dotnet dev-certs https --verbose
3. dotnet dev-certs https --trust

Zertifikate

openssl - selfsign - certificate

To create a self-signed PFX certificate using OpenSSL, you can follow these steps:

1. **Generate a private key**:

You can generate a private key using the `openssl genpkey` command. Here, I'll generate a 2048-bit RSA private key and save it to a file named `private.key`:

```
openssl genpkey -algorithm RSA -out private.key -aes256
```

This command will prompt you to enter a passphrase to protect the private key.

2. **Create a certificate signing request (CSR)**:

You can create a CSR using the private key generated in the previous step. This CSR will be used to generate the self-signed certificate:

```
openssl req -new -key private.key -out certificate.csr
```

Follow the prompts to fill in the information for your certificate.

3. **Generate a self-signed certificate**:

Now, you can generate a self-signed certificate using the CSR and the private key:

```
openssl x509 -req -days 365 -in certificate.csr -signkey private.key -out certificate.crt
```

This command will generate a self-signed certificate valid for 365 days.

4. **Create the PFX file**:

To create a PFX file (also known as PKCS#12), you need both the private key and the certificate:

```
openssl pkcs12 -export -out certificate.pfx -inkey private.key -in certificate.crt
```

This command will prompt you to enter a passphrase to protect the PFX file.

Now, you have a self-signed PFX certificate file named `certificate.pfx`. Make sure to securely store both the private key and the PFX file, as they contain sensitive information.

T-SQL

T-SQL

Datenbankfelder auflisten

```
SELECT TABLE_CATALOG, TABLE_NAME, COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
ORDER BY TABLE_NAME
```

Blazor WASM

Update

Um eine Blazor-Webassembly Anwendung zu aktualisieren

- Im Unterordner die Datei öffnen und unter

  ```
  <script>navigator.serviceWorker.register('service-worker.js');</script>
  ```

  die folgende Zeile einfügen:

  ```
  <script>navigator.serviceWorker.register('version.js');</script>
  ```

- Und erstellen Sie in diesem Ordner eine "version.js"-Datei mit einem einzigen Inhalt:

  ```
  const CACHE_VERSION = 1.0
  ```

- Erhöhen `1.0` Sie den Wert jedes Mal, wenn Ihre Clients die neue Version laden sollen. Sobald es Ihren Clients gelungen ist, ihren Cache zu leeren und die Version zu laden, die diesen Code enthält, wird die neue Version durch einfaches Neuladen oder Herunterfahren und Neustarten des Browsers aktiviert.

Bridge To Kubernetes

Debug - launchSettings.json

Entschuldigung für das Missverständnis. Es scheint, dass die Option zum Erstellen eines Bridge to Kubernetes-Profils direkt in den Projekteigenschaften von Visual Studio 2022 nicht verfügbar ist. Du kannst die Bridge to Kubernetes-Einstellungen jedoch weiterhin über die Datei "launchSettings.json" konfigurieren.

Hier ist ein Beispiel, wie du das Bridge to Kubernetes-Profil manuell in der "launchSettings.json"-Datei hinzufügen kannst:

1. Öffne die Datei "launchSettings.json" für dein Projekt. Wenn sie nicht vorhanden ist, kannst du sie im Stammverzeichnis deines Projekts erstellen.

2. Füge ein neues Profil für Bridge to Kubernetes unter dem Abschnitt "profiles" hinzu. Hier ist ein Beispiel:

```
{
  "$schema": "https://json.schemastore.org/launchsettings.json",
  "profiles": {
    "Bridge to Kubernetes": {
      "commandName": "Project",
      "launchBrowser": true,
      "applicationUrl": "https://localhost:44334;http://localhost:5000",
      "inspectUri": "{wsProtocol}://{url.hostname}:{url.port}/_bridge/"
    }
  }
}
```

Stelle sicher, dass du die Werte für "applicationUrl" und "inspectUri" an die Konfiguration deines Projekts anpasst.

3. Speichere die Datei "launchSettings.json".

Mit dieser Konfiguration solltest du nun das "Bridge to Kubernetes"-Profil aus dem Debug-Dropdown-Menü in Visual Studio 2022 auswählen und dein Projekt mithilfe von Bridge to Kubernetes debuggen können.