https://www.halvorsen.blog

ASP.NET Core

Connection String

Hans-Petter Halvorsen

Introduction

- The Connection string is used to connect to the database
- In this tutorial we will use SQL Server, Visual Studio, C#
- We will show how we use Connection String in an ASP.NET Core Web Application

https://www.halvorsen.blog

SQL Server

Hans-Petter Halvorsen

SQL Server

- SQL Server is a Database System from Microsoft. SQL Server comes in different editions, for basic, personal use
- SQL Server Express is recommended because it is simple to use, and it is free.
- Latest version is SQL Server 2019.

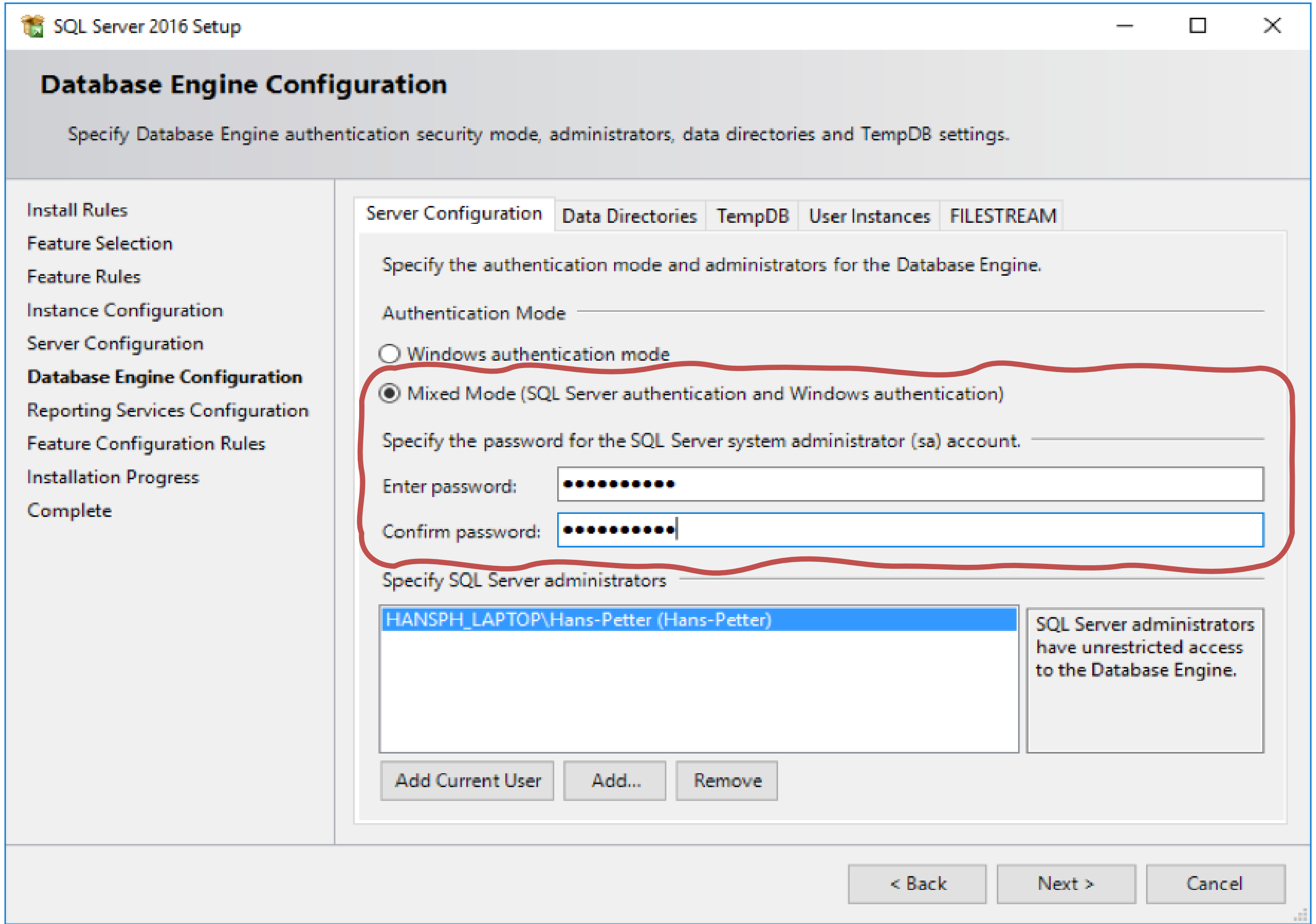
SQL Server Installation

- During the setup of SQL Server you should select "Mixed Mode" (i.e., both "SQL Server Authentication" and "Windows Authentication") and enter the password for your sa user.
- "Windows Authentication" is the default option during installation, so make sure to "Mixed Mode" (i.e., both "SQL Server Authentication" and "Windows Authentication") and enter the password for your sa user
- Make sure to remember the sa password!

SQL Server Installation - Mixed Mode

- During Installation of SQL Server: Select "**Mixed Mode**" (i.e., both SQL Server Authentication and Windows Authentication)
- Make sure to remember the "**sa**" Password!
- "sa" is short for **S**ystem **A**dministrator

SQL Server Installation - Mixed Mode



SQL Server 2016 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories and TempDB settings.

Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configuration
Feature Configuration Rules
Installation Progress
Complete

Server Configuration | Data Directories | TempDB | User Instances | FILESTREAM

Specify the authentication mode and administrators for the Database Engine.

Authentication Mode

○ Windows authentication mode

● Mixed Mode (SQL Server authentication and Windows authentication)

Specify the password for the SQL Server system administrator (sa) account.

Enter password: ●●●●●●●●●●●

Confirm password: ●●●●●●●●●●●

Specify SQL Server administrators

HANSPH_LAPTOP\Hans-Petter (Hans-Petter)

SQL Server administrators have unrestricted access to the Database Engine.

https://www.halvorsen.blog

Authentication

Hans-Petter Halvorsen

Visual Studio

- In WinForm Desktop Applications you should put the Connection String in the App.config file
- While for ASP.NET Core Web Applications the Connection String should be placed in the in the appSettings.json file.

Authentication Methods

SQL offer 2 different Authentication methods:

- SQL Server Authentication
- Windows Authentication

SQL Server Authentication

Connection String - SQL Server Authentication

Using "SQL Server Authentication" the Connection String looks like this:

```
DATA SOURCE=<SQL Server Name>;DATABASE=<Database Name>;UID=sa;PWD=<Your Password>;
```

Replace <SQL Server Name> with the name of your SQL Server, typically "<YourComputerName>\SQLEXPRESS" if you are using SQL Server Express.

UID is a SQL Server user, here you can create your own SQL Server user inside SQL Server Management Studio or use the built-in sa user (sa=System Administrator). During the setup of SQL Server you need to select "Mixed Mode" and enter the password for your sa user.

It may look something like this:

```
DATA SOURCE=DELLPCWORK\\SQLEXPRESS;DATABASE=MEASUREMENTS;UID=sa;PWD=Password123;
```

Localhost

If you don't know the name of your PC or if you use multiple PC, it may be a good idea to use "LOCALHOST" instead of your real computer name (assuming the application and the database in located on the same computer)

```
DATA SOURCE=LOCALHOST\\SQLEXPRESS;DATABASE=MEASUREMENTS;UID=sa;PWD=Password123;
```

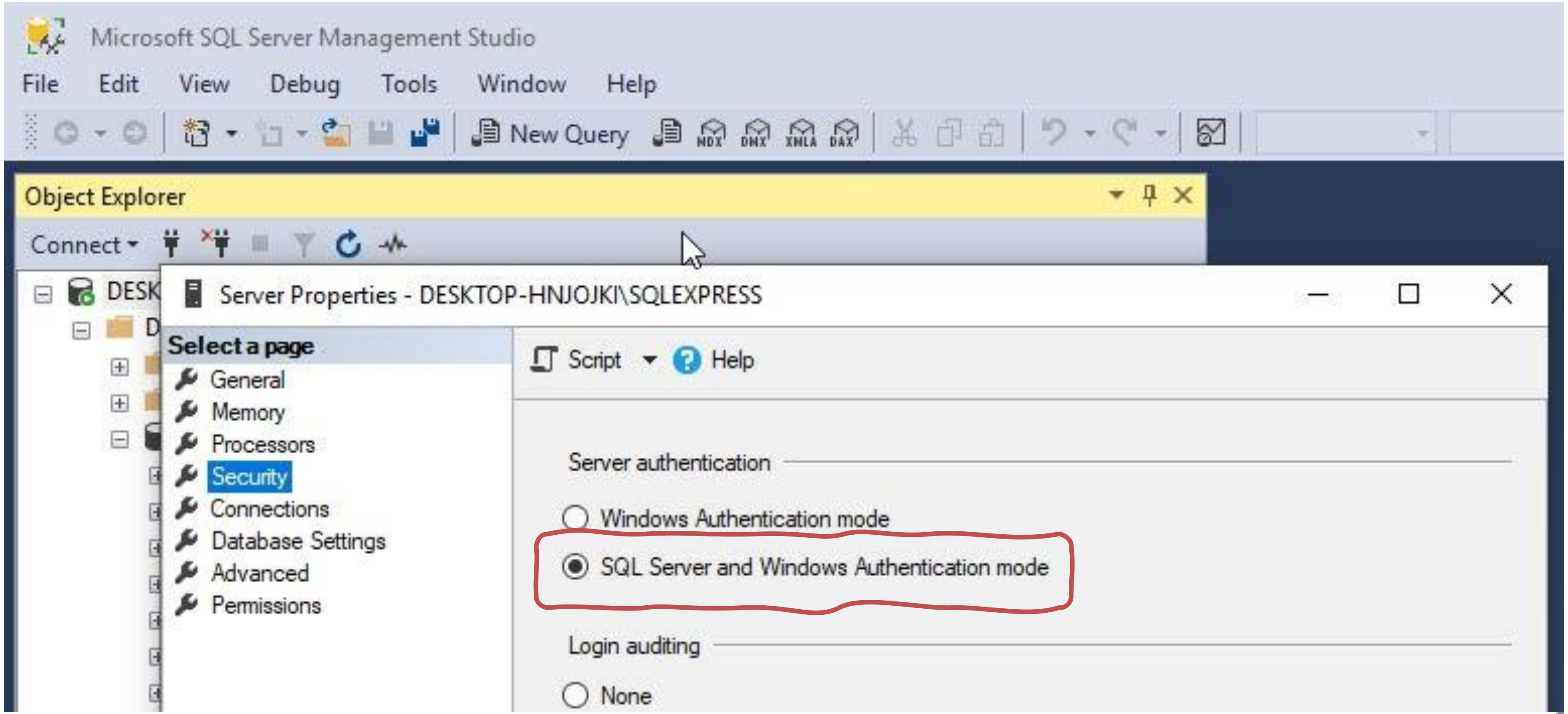
Enable SQL Server Authentication in SSMS

You can also turn on "SQL Server Authentication" in SQL Server Management Studio (SSMS) <u>after installation</u> of SQL Server.

To change security authentication mode, do the following steps:

1. In SQL Server Management Studio Object Explorer, right-click the server, and then click Properties.
2. On the Security page, under Server authentication, select the new server authentication mode, and then click OK.
3. In the SQL Server Management Studio dialog box, click OK to acknowledge the requirement to restart SQL Server.
4. In Object Explorer, right-click your server, and then click Restart. If SQL Server Agent is running, it must also be restarted. Or just restart your computer.

Enable SQL Server Authentication



Server authentication

( ) Windows Authentication mode

(●) SQL Server and Windows Authentication mode

Login auditing

( ) None

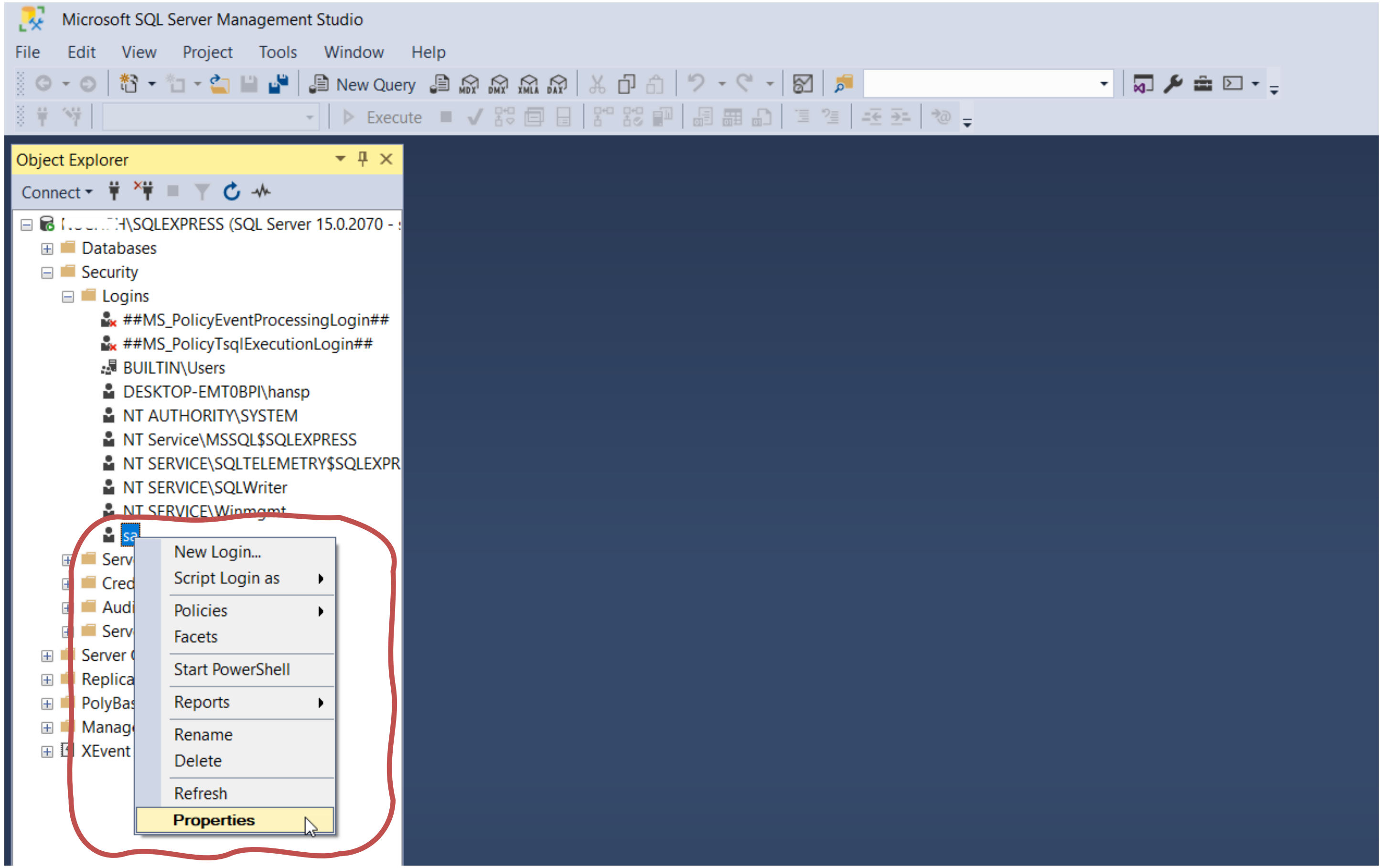
Enable sa login

Then to enable the sa login, do the following steps:

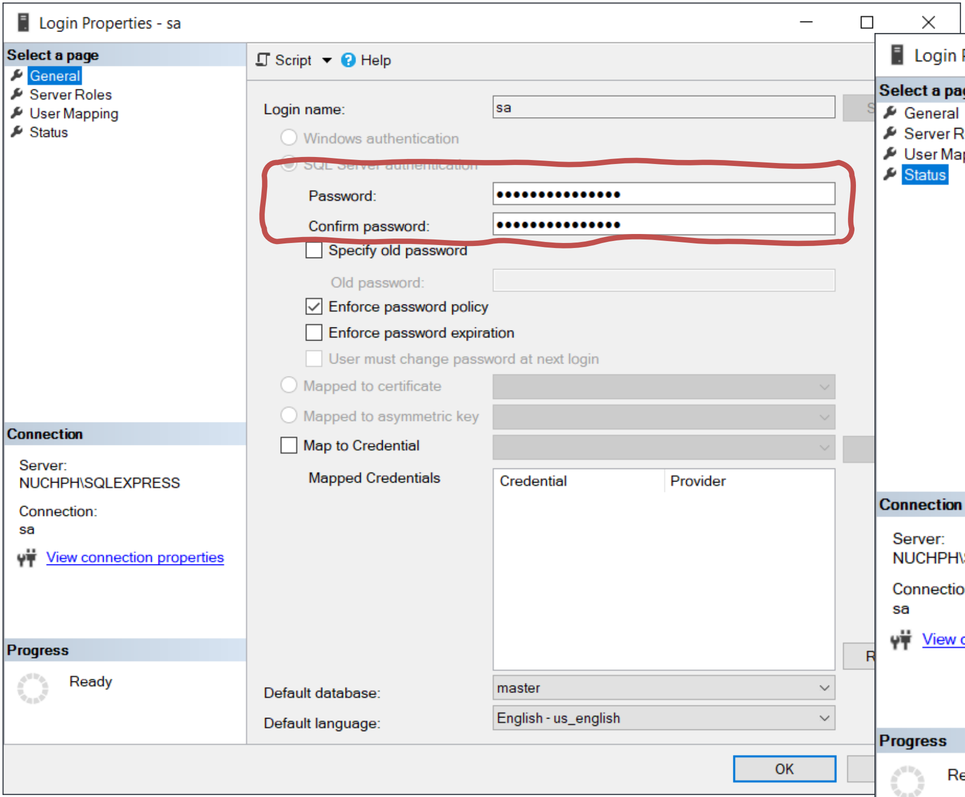
1. In Object Explorer, expand Security, expand Logins, right-click sa, and then click Properties.
2. On the General page, you might have to create and confirm a password for the login.
3. On the Status page, in the Login section, click Enabled, and then click OK.

Note! You must restart your computer afterwards (well, it is enough to restart the “Sql service...”) in order to work.

Enable sa login



Enable sa login



Login Properties - sa

Select a page
- General
- Server Roles
- User Mapping
- Status

Script ▾ Help

Login name: sa

( ) Windows authentication

(●) SQL Server authentication

Password: ●●●●●●●●●●●●●●●

Confirm password: ●●●●●●●●●●●●●●●

[ ] Specify old password

Old password:

[✓] Enforce password policy

[ ] Enforce password expiration

[ ] User must change password at next login

( ) Mapped to certificate

( ) Mapped to asymmetric key

[ ] Map to Credential

Mapped Credentials

Credential	Provider

Default database: master

Default language: English - us_english

Connection

Server:
NUCHPH\SQLEXPRESS

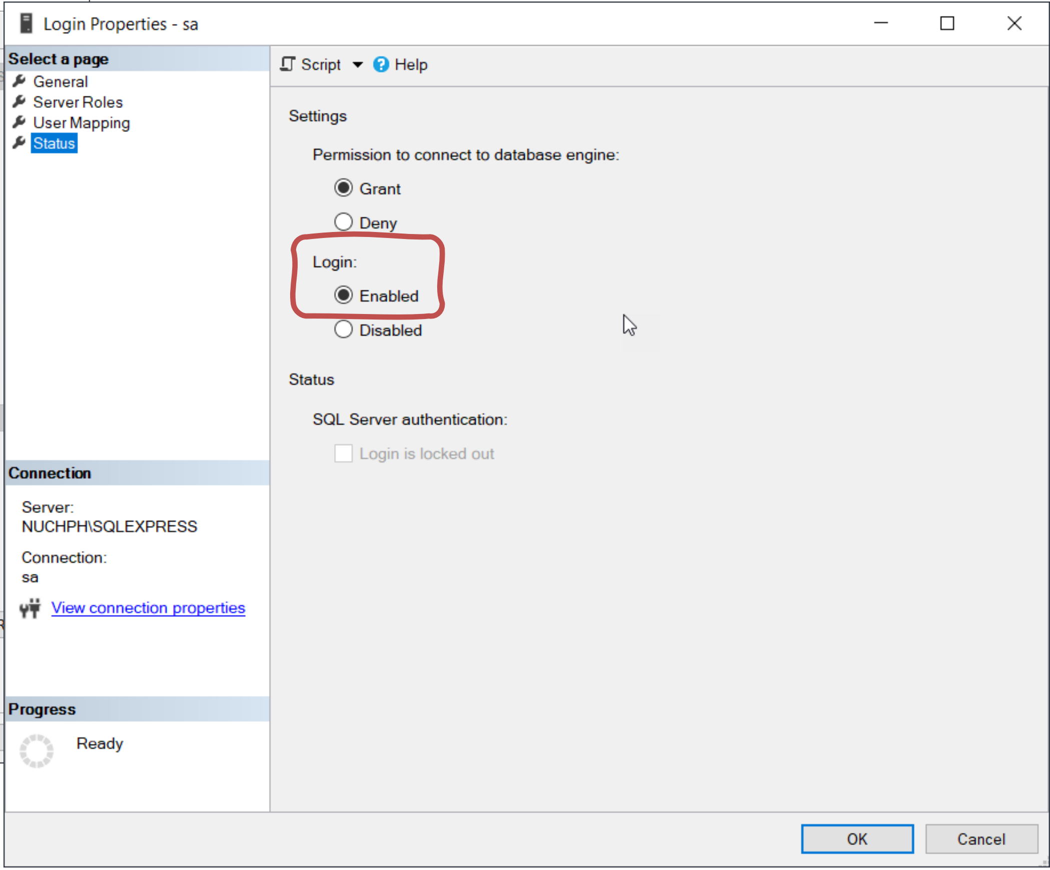
Connection:
sa

View connection properties

Progress

Ready

OK



Login Properties - sa

Select a page
- General
- Server Roles
- User Mapping
- Status

Script ▾ Help

Settings

Permission to connect to database engine:

(●) Grant

( ) Deny

Login:

(●) Enabled

( ) Disabled

Status

SQL Server authentication:

[ ] Login is locked out

Connection

Server:
NUCHPH\SQLEXPRESS

Connection:
sa

View connection properties

Progress

Ready

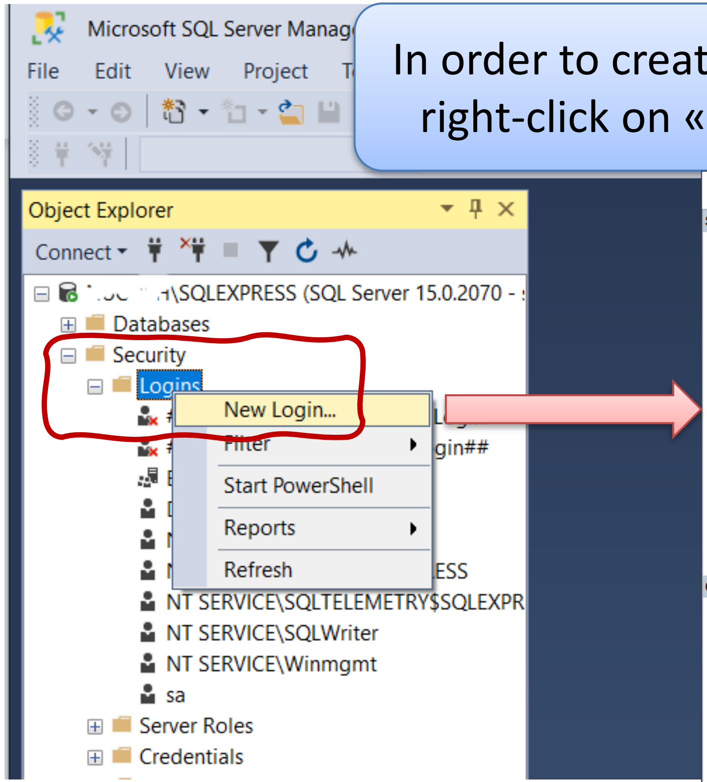
OK Cancel

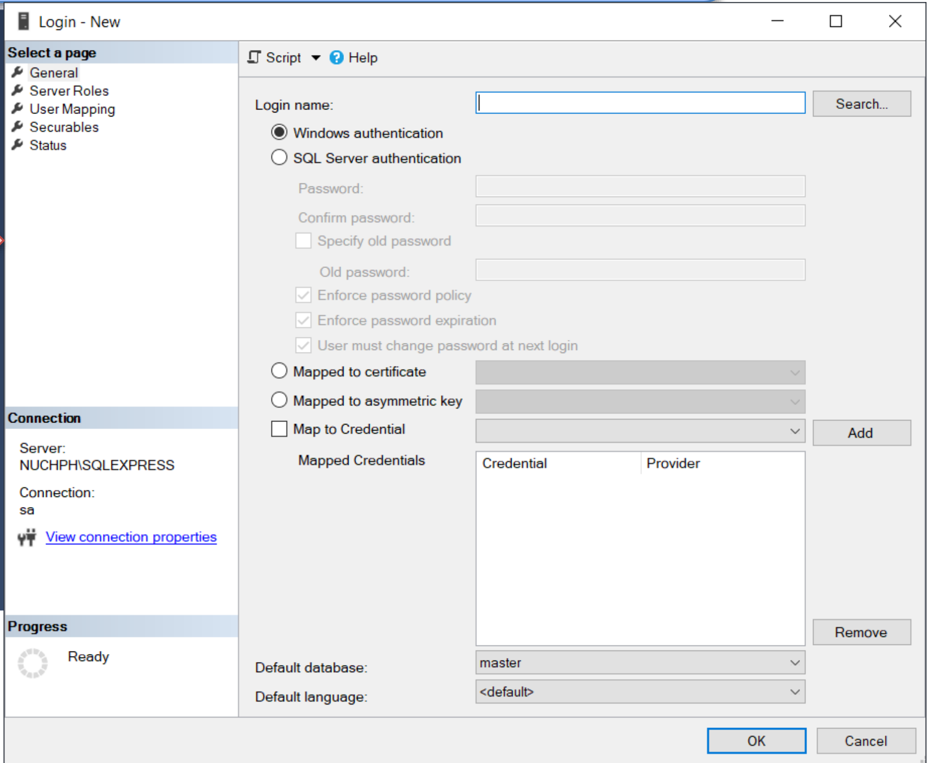
Create Logins in SQL Server

- “sa” is a built-in Login in SQL Server
- You can also create your own SQL Server Logins
- Normally you should do that rather than using the “sa” login
- “sa” have access to “everything” and in context of Data Security that is unfortunate.
- In general, you should make your own Logins that have access to only what's strictly necessary

Create Logins in SQL Server

In order to create a new Login, goto «Security» and right-click on «Logins» and select «New Login...»





Login name:

[✓ Windows authentication]

SQL Server authentication

Password:

Confirm password:

Specify old password

Old password:

[✓ Enforce password policy]

[✓ Enforce password expiration]

[✓ User must change password at next login]

Mapped to certificate

Mapped to asymmetric key

Map to Credential

Mapped Credentials

Credential	Provider

Default database: master

Default language: <default>

Connection

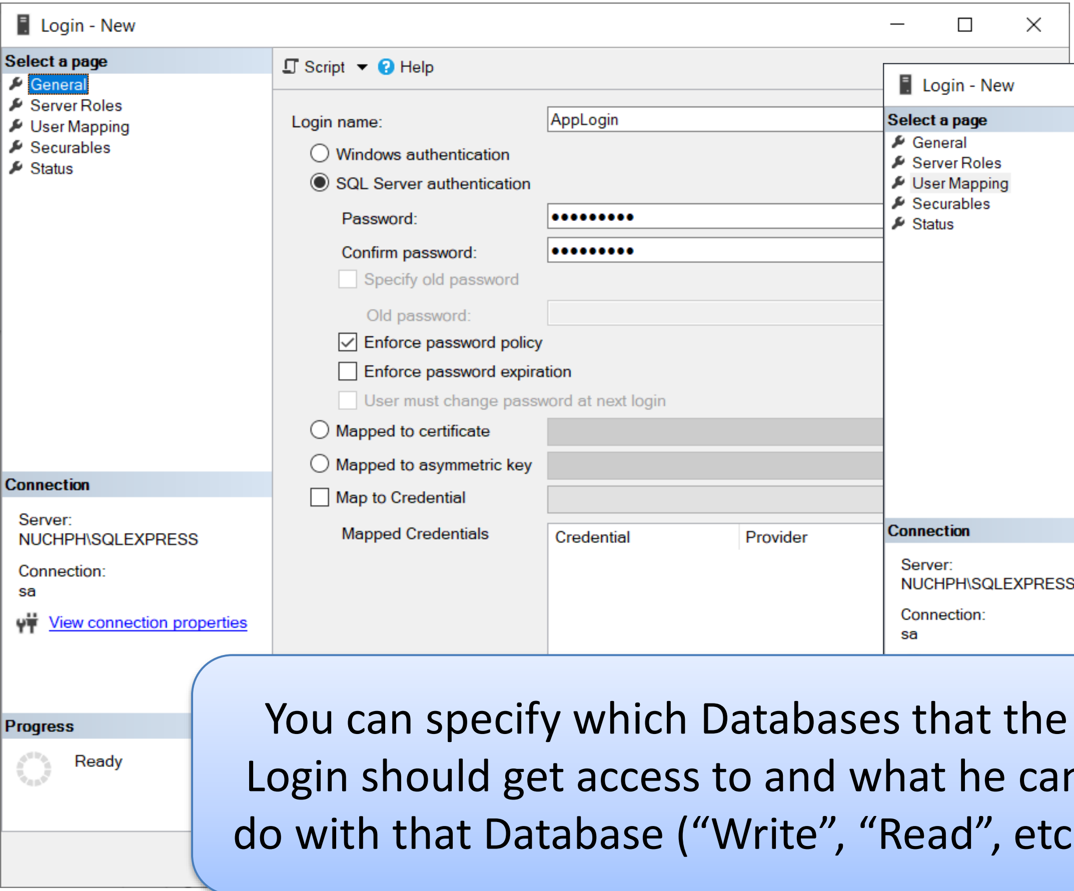
Server:
NUCHPH\SQLEXPRESS

Connection:
sa

Progress

Ready

Create Logins in SQL Server



Login - New

Select a page
- General
- Server Roles
- User Mapping
- Securables
- Status

Script ▾ Help

Login name: AppLogin

( ) Windows authentication
(●) SQL Server authentication

Password: ●●●●●●●●●●
Confirm password: ●●●●●●●●●●

☐ Specify old password
Old password:

☑ Enforce password policy
☐ Enforce password expiration
☐ User must change password at next login

( ) Mapped to certificate
( ) Mapped to asymmetric key
☐ Map to Credential

Mapped Credentials

Credential	Provider

Connection

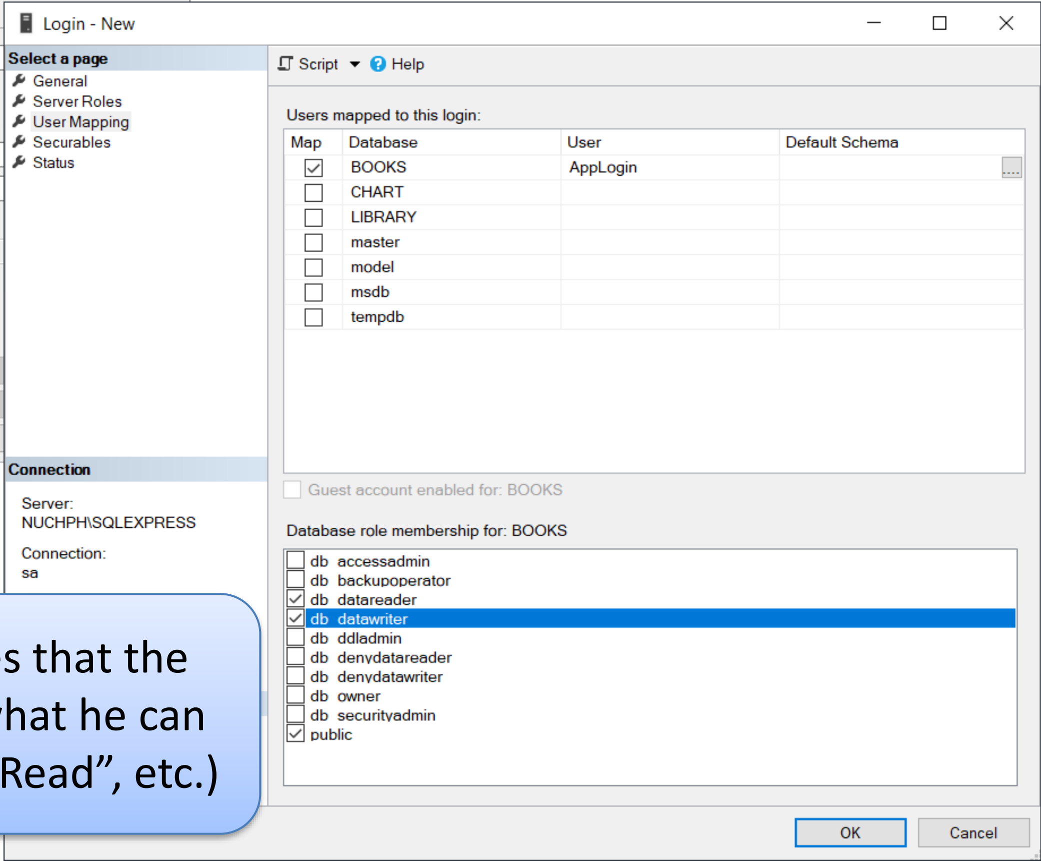
Server:
NUCHPH\SQLEXPRESS

Connection:
sa

View connection properties

Progress

Ready



Login - New

Select a page
- General
- Server Roles
- User Mapping
- Securables
- Status

Script ▾ Help

Users mapped to this login:

Map	Database	User	Default Schema
☑	BOOKS	AppLogin	
☐	CHART		
☐	LIBRARY		
☐	master		
☐	model		
☐	msdb		
☐	tempdb		

☐ Guest account enabled for: BOOKS

Database role membership for: BOOKS

- ☐ db_accessadmin
- ☐ db_backupoperator
- ☑ db_datareader
- ☑ db_datawriter
- ☐ db_ddladmin
- ☐ db_denydatareader
- ☐ db_denydatawriter
- ☐ db_owner
- ☐ db_securityadmin
- ☑ public

Connection

Server:
NUCHPH\SQLEXPRESS

Connection:
sa

OK Cancel

You can specify which Databases that the Login should get access to and what he can do with that Database (“Write”, “Read”, etc.)

Windows Authentication

Windows Authentication

Using "Windows Authentication" the Connection String looks like this:

DATA SOURCE=DELLPCWORK\\SQLEXPRESS;DATABASE=MEASUREMENTS;Integrated Security = True;

Localhost:

If you don't know the name of your PC or if you use multiple PC, it may be a good idea to use "LOCALHOST" instead of your real computer name (assuming the application and the database in located on the same computer).

DATA SOURCE=LOCALHOST\\SQLEXPRESS;DATABASE=MEASUREMENTS;Integrated Security = True;

https://www.halvorsen.blog

ASP.NET Core

Hans-Petter Halvorsen

Introduction

- **appSettings.json** is a configuration file used in ASP.NET Core Web Applications
- It is typically used to store the Connection String to the Database
- But it can be used to store lots of other settings that you need to use in your application

ASP.NET Core

If you have never used ASP.NET Core, I suggest the following Videos:

- ASP.NET Core - Hello World
  https://youtu.be/lcQsWYgQXK4
- ASP.NET Core – Introduction
  https://youtu.be/zkOtiBcwo8s

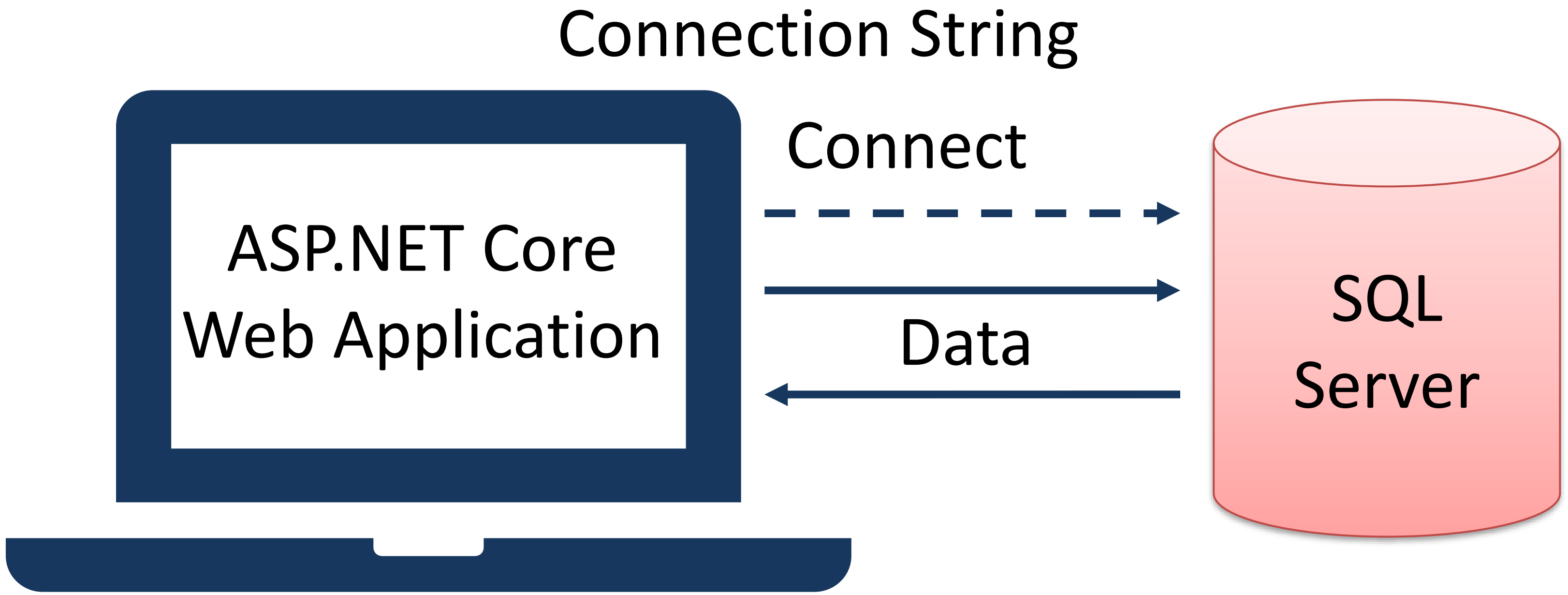
ASP.NET Core Resources:
https://halvorsen.blog/documents/programming/web/aspnet

https://www.halvorsen.blog

Connection String in appSettings.json

Hans-Petter Halvorsen

Connection String



```
ConnectionString":"DATA SOURCE=xxx; DATABASE=xxx;UID=xxx;PWD=xxx
```

appSettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",

  "ConnectionStrings": {
    "ConnectionString": "DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=xxx"
  }

 }
```

Startup.cs

We need to add something to the "**Startup.cs**" file:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();

    services.AddSingleton<IConfiguration>(Configuration);
}
```

We have added:

services.AddSingleton<IConfiguration>(Configuration);

Example

Example

SQL Server

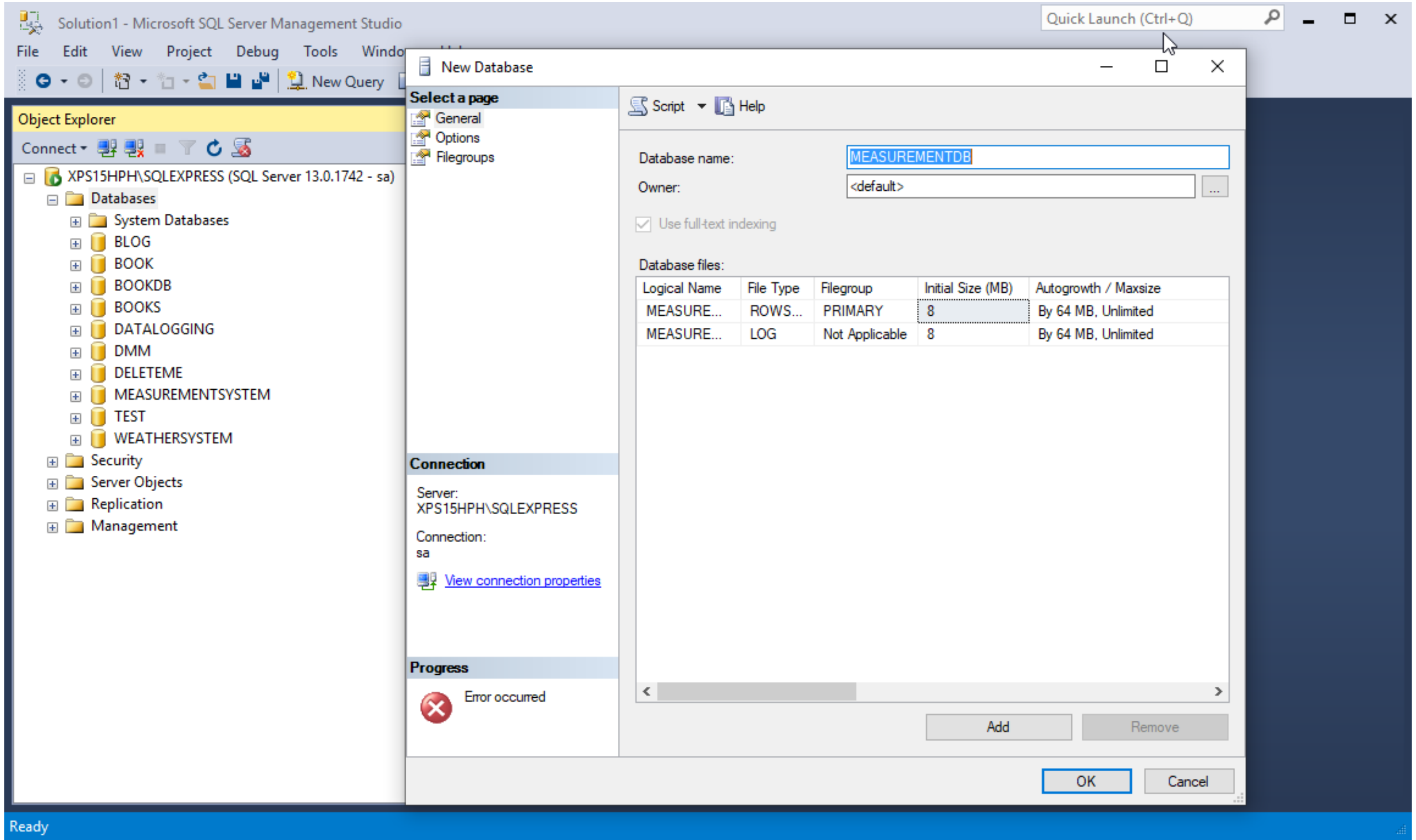
SQL Server

- We will use SQL Server in this example as our database.
- You should have SQL Server locally installed on your computer
- SQL Server Express is recommended.

Example

Database

SQL Server - Create Database



Database name: MEASUREMENTDB

Owner: <default>

Use full-text indexing

Server: XPS15HPH\SQLEXPRESS

Connection: sa

Database files:

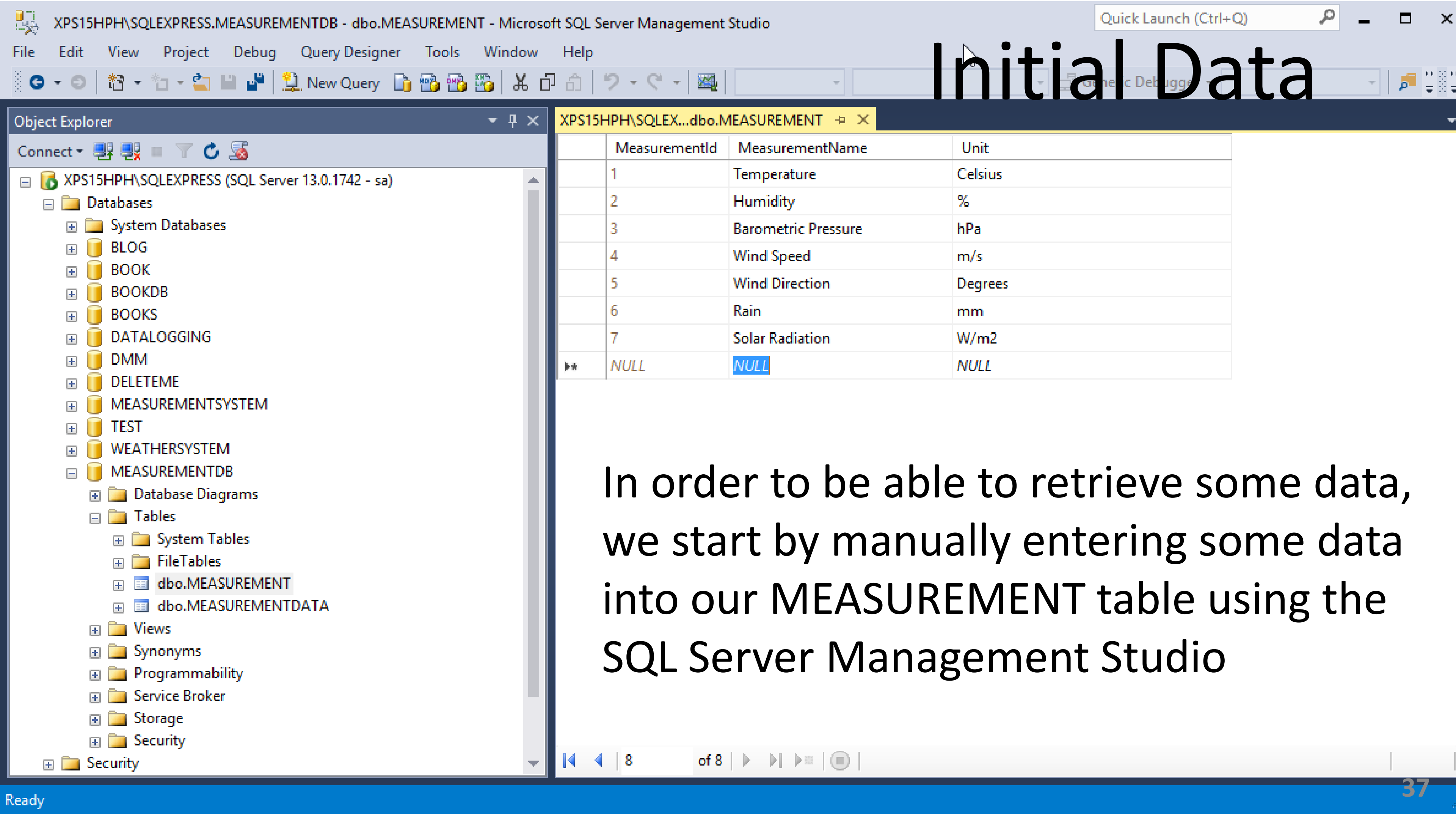
Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize
MEASURE...	ROWS...	PRIMARY	8	By 64 MB, Unlimited
MEASURE...	LOG	Not Applicable	8	By 64 MB, Unlimited

Database Table

```
CREATE TABLE [MEASUREMENT]
(
      [MeasurementId]    int   NOT NULL  IDENTITY ( 1,1 ) Primary Key,
      [MeasurementName] varchar(100) NOT NULL UNIQUE,
      [Unit]            varchar(50)  NULL
)
go
```

You can use SQL Server Management Studio in order to run this SQL Script

Initial Data



MeasurementId	MeasurementName	Unit
1	Temperature	Celsius
2	Humidity	%
3	Barometric Pressure	hPa
4	Wind Speed	m/s
5	Wind Direction	Degrees
6	Rain	mm
7	Solar Radiation	W/m2
NULL	NULL	NULL

In order to be able to retrieve some data, we start by manually entering some data into our MEASUREMENT table using the SQL Server Management Studio

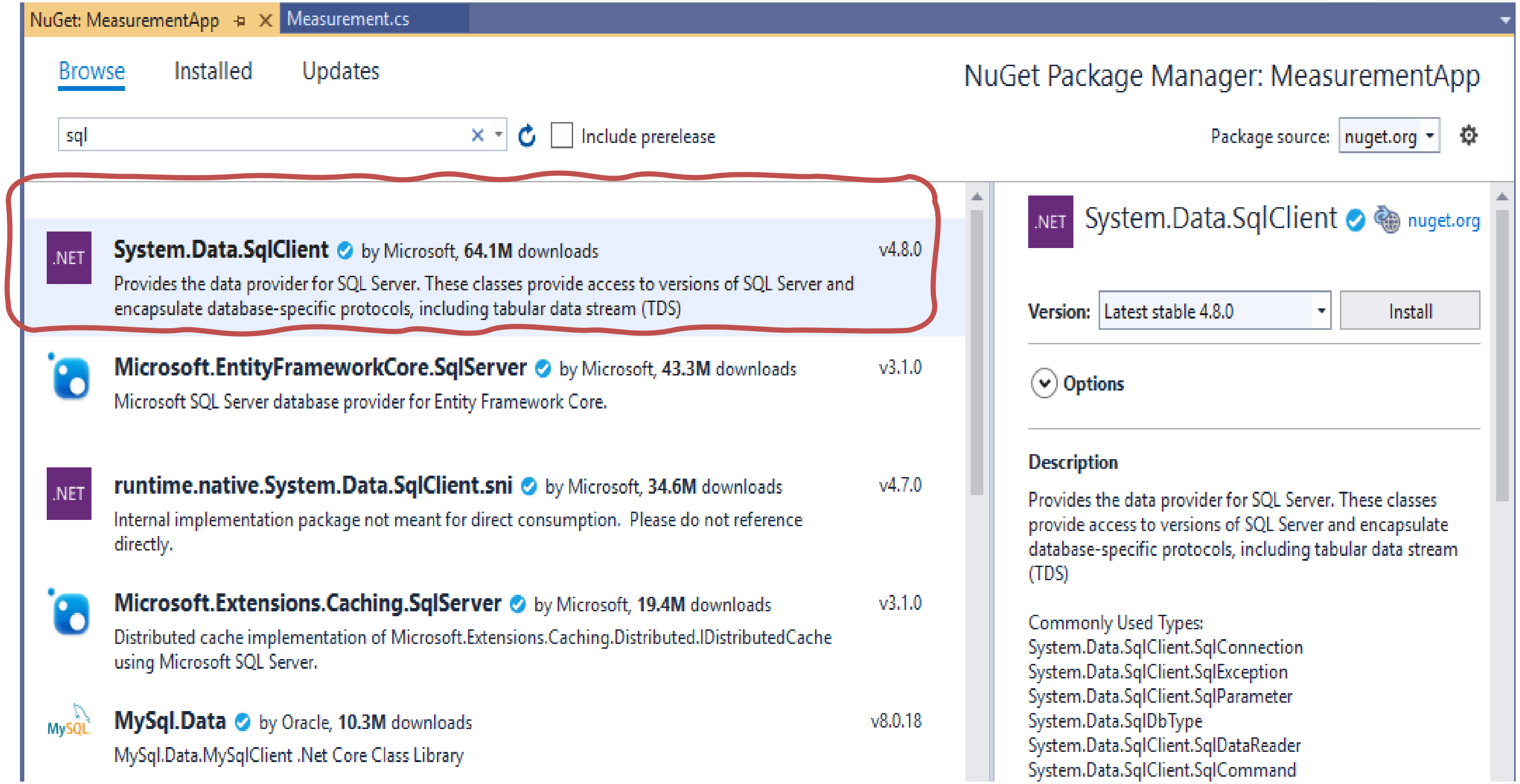
Example

Visual Studio

ASP.NET Core Web Application

NuGet

Make sure to install the necessary NuGet package(s). We will use the **System.Data.SqlClient**



System.Data.SqlClient by Microsoft, 64.1M downloads v4.8.0
Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, including tabular data stream (TDS)

Microsoft.EntityFrameworkCore.SqlServer by Microsoft, 43.3M downloads v3.1.0
Microsoft SQL Server database provider for Entity Framework Core.

runtime.native.System.Data.SqlClient.sni by Microsoft, 34.6M downloads v4.7.0
Internal implementation package not meant for direct consumption. Please do not reference directly.

Microsoft.Extensions.Caching.SqlServer by Microsoft, 19.4M downloads v3.1.0
Distributed cache implementation of Microsoft.Extensions.Caching.Distributed.IDistributedCache using Microsoft SQL Server.

MySql.Data by Oracle, 10.3M downloads v8.0.18
MySql.Data.MySqlClient .Net Core Class Library

System.Data.SqlClient nuget.org

Version: Latest stable 4.8.0 Install

Options

Description

Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, including tabular data stream (TDS)

Commonly Used Types:
System.Data.SqlClient.SqlConnection
System.Data.SqlClient.SqlException
System.Data.SqlClient.SqlParameter
System.Data.SqlDbType
System.Data.SqlClient.SqlDataReader
System.Data.SqlClient.SqlCommand

appSettings.json

```
{
 "Logging": {
  "LogLevel": {
   "Default": "Information",
   "Microsoft": "Warning",
   "Microsoft.Hosting.Lifetime": "Information"
  }
 },
 "AllowedHosts": "*",

 "ConnectionStrings": {
  "ConnectionString": "DATA SOURCE=xxx\\SQLEXPRESS;DATABASE=xxx;UID=sa;PWD=xxx"
 }

}
```

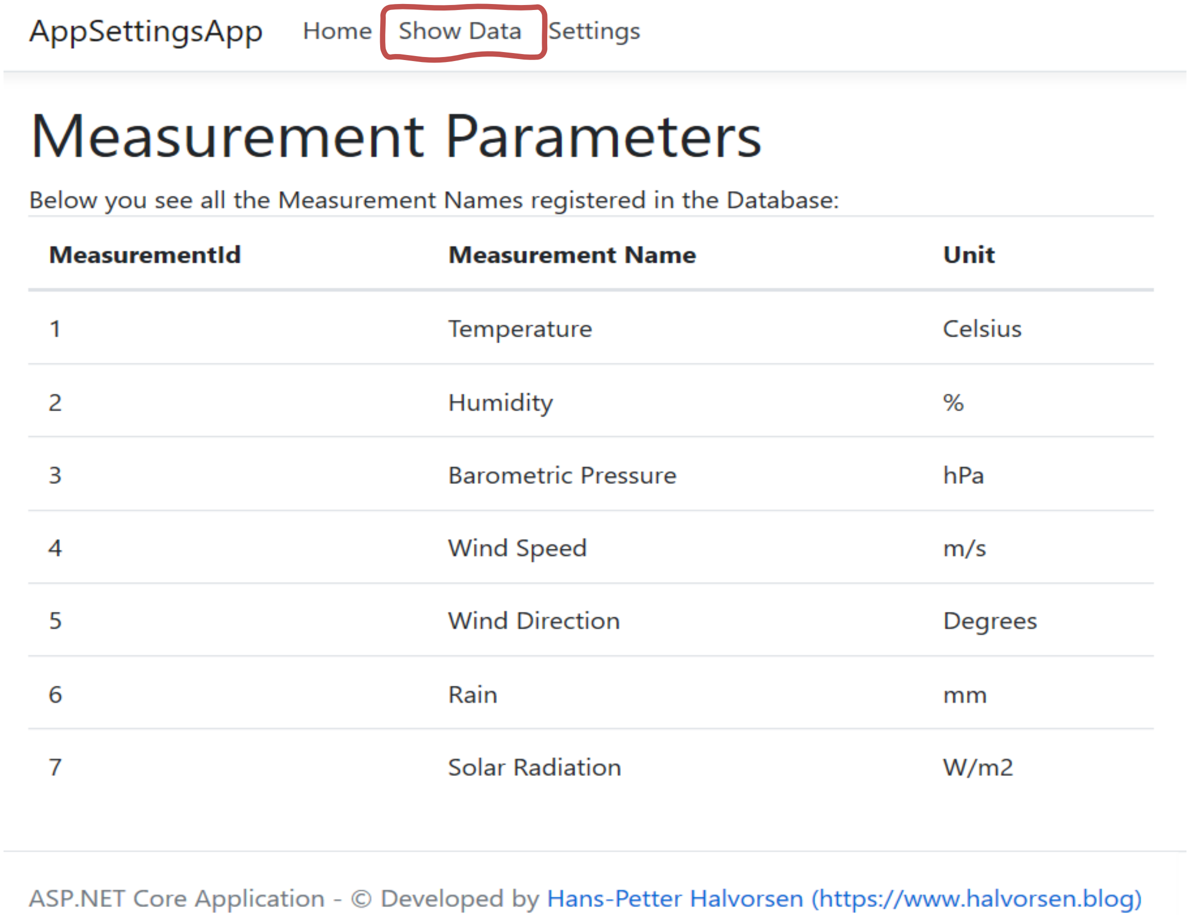
C# Code

```
…
using Microsoft.Extensions.Configuration;
public class xxxModel : PageModel
{
   readonly IConfiguration _configuration;

   private string connectionString;

   public xxxModel(IConfiguration configuration)
   {
       _configuration = configuration;
   }
   …
   connectionString =
   _configuration.GetConnectionString("ConnectionString");

}
```

The Constructor

ASP.NET Core Web Application

The following Application will be demonstrated here:



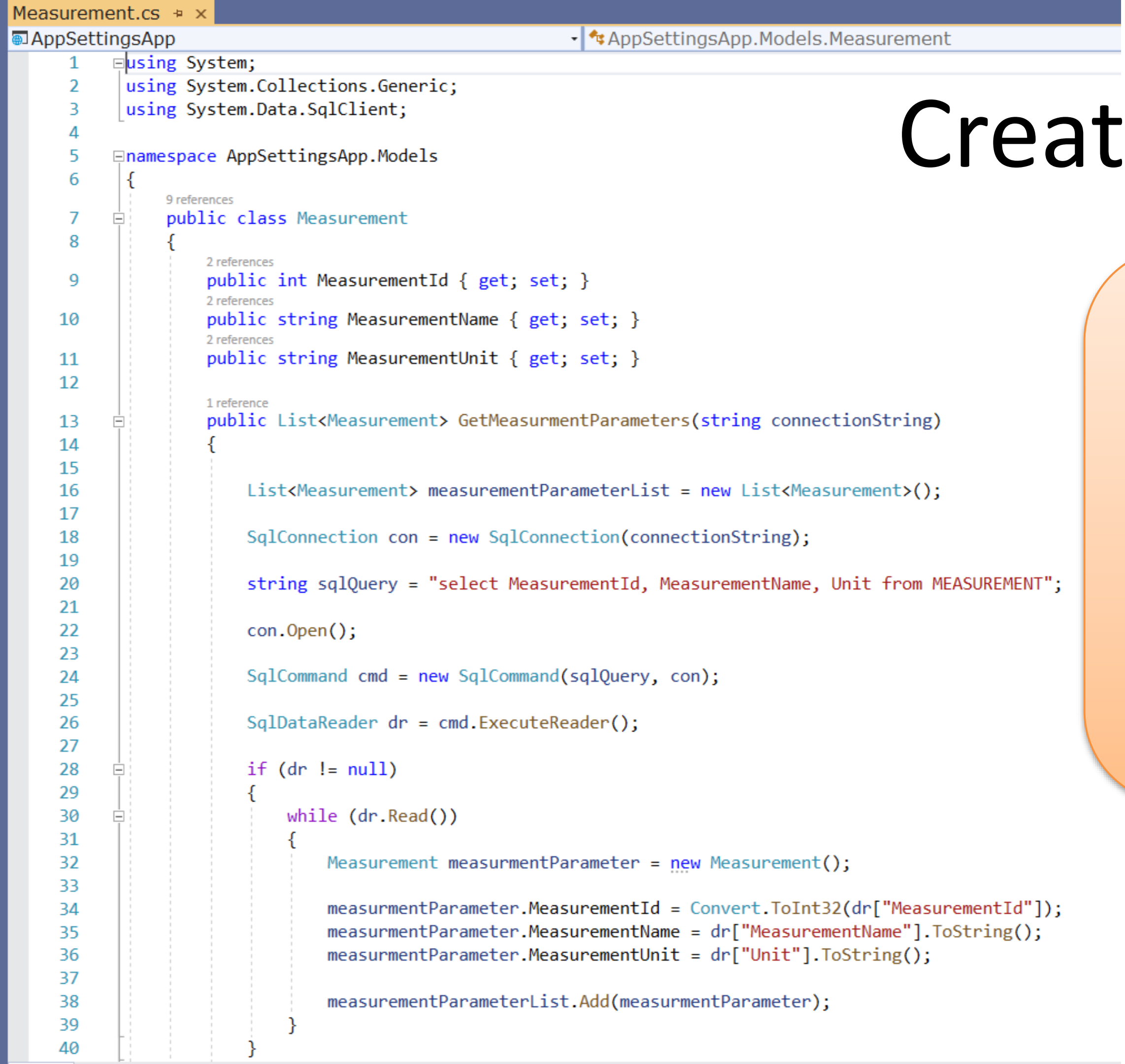
Measurement Parameters

Below you see all the Measurement Names registered in the Database:

MeasurementId	Measurement Name	Unit
1	Temperature	Celsius
2	Humidity	%
3	Barometric Pressure	hPa
4	Wind Speed	m/s
5	Wind Direction	Degrees
6	Rain	mm
7	Solar Radiation	W/m2

ASP.NET Core Application - © Developed by Hans-Petter Halvorsen (https://www.halvorsen.blog)

We will retrieve these data from a SQL Server Database



```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;

namespace AppSettingsApp.Models
{
    public class Measurement
    {
        public int MeasurementId { get; set; }
        public string MeasurementName { get; set; }
        public string MeasurementUnit { get; set; }

        public List<Measurement> GetMeasurmentParameters(string connectionString)
        {

            List<Measurement> measurementParameterList = new List<Measurement>();

            SqlConnection con = new SqlConnection(connectionString);

            string sqlQuery = "select MeasurementId, MeasurementName, Unit from MEASUREMENT";

            con.Open();

            SqlCommand cmd = new SqlCommand(sqlQuery, con);

            SqlDataReader dr = cmd.ExecuteReader();

            if (dr != null)
            {
                while (dr.Read())
                {
                    Measurement measurmentParameter = new Measurement();

                    measurmentParameter.MeasurementId = Convert.ToInt32(dr["MeasurementId"]);
                    measurmentParameter.MeasurementName = dr["MeasurementName"].ToString();
                    measurmentParameter.MeasurementUnit = dr["Unit"].ToString();

                    measurementParameterList.Add(measurmentParameter);
                }
            }
```

Create Database Class

- We start by creating a **Models** folder in our project using the Solutions Explorer
- Then we create a new Class ("**Measurement.cs**")
- Then we create C# Code for retrieving data from the Database

“Measurement.cs”

```
using System.Data.SqlClient;

namespace MeasurementApp.Model
{
    public class Measurement
    {
        public int MeasurementId { get; set; }
        public string MeasurementName { get; set; }
        public string MeasurementUnit { get; set; }

        public List<Measurement> GetMeasurmentParameters(string connectionString)
        {
            List<Measurement> measurementParameterList = new List<Measurement>();

            SqlConnection con = new SqlConnection(connectionString);

            string sqlQuery = "select MeasurementId, MeasurementName, Unit from MEASUREMENT";

            con.Open();

            SqlCommand cmd = new SqlCommand(sqlQuery, con);

            SqlDataReader dr = cmd.ExecuteReader();

            if (dr != null)
            {
                while (dr.Read())
                {
                    Measurement measurmentParameter = new Measurement();

                    measurmentParameter.MeasurementId = Convert.ToInt32(dr["MeasurementId"]);
                    measurmentParameter.MeasurementName = dr["MeasurementName"].ToString();
                    measurmentParameter.MeasurementUnit = dr["Unit"].ToString();

                    measurementParameterList.Add(measurmentParameter);
                }
            }
            return measurementParameterList;
        }
    }
}
```

An ASP.NET Core Web Page consist of the following:

- “Database.**cshtml”** - HTML/Razor code
- “Database.**cshtml.cs”** - Page Model (Code behind C# File)

“Database.cshtml.cs”

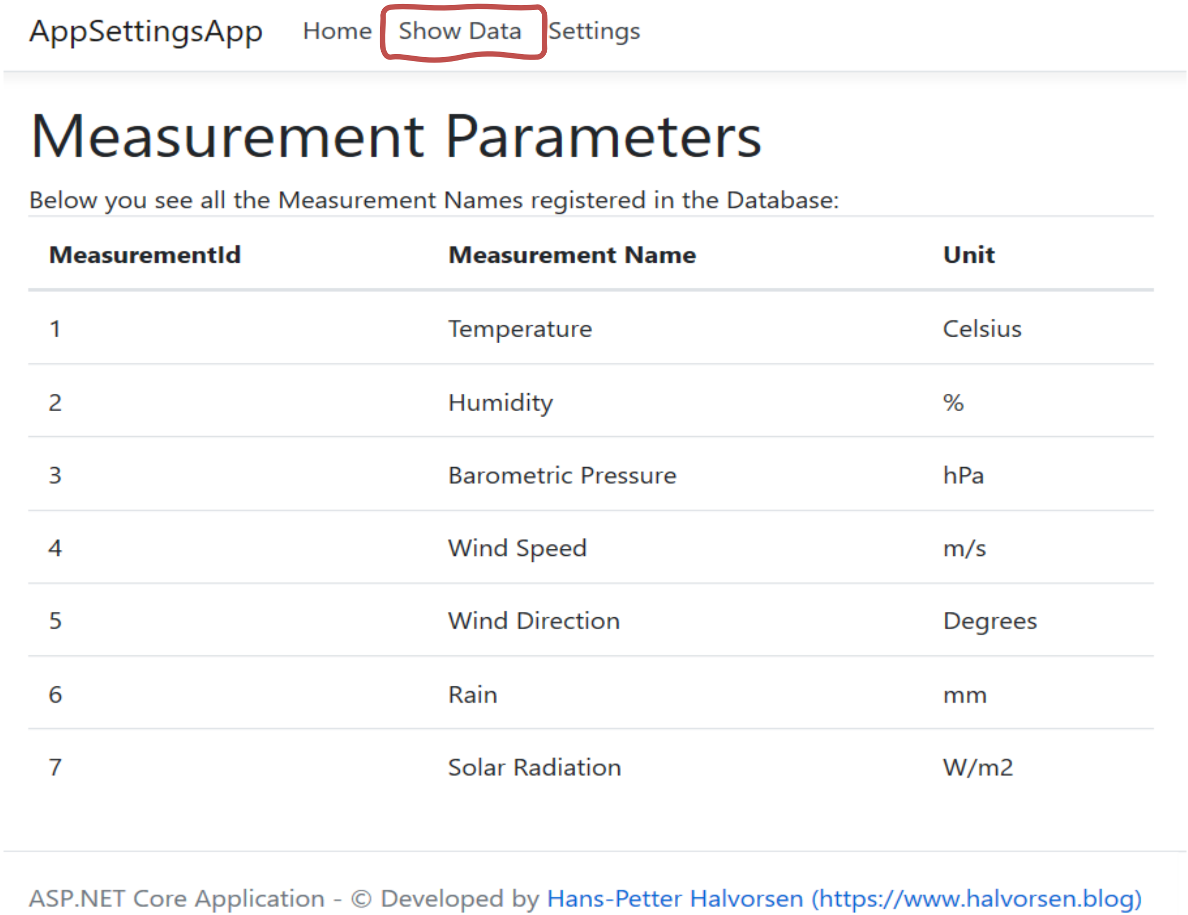
```
...
using Microsoft.Extensions.Configuration;
using AppSettingsApp.Models;

namespace AppSettingsApp.Pages
{
    public class DatabaseModel : PageModel
    {
        readonly IConfiguration _configuration;

        public List<Measurement> measurementParameterList = new List<Measurement>();

        public string connectionString;

        public DatabaseModel(IConfiguration configuration)
        {
            _configuration = configuration;
        }
        public void OnGet()
        {
            GetData();
        }

        void GetData()
        {
                Measurement measurement = new Measurement();

                connectionString = _configuration.GetConnectionString("ConnectionString");

                measurementParameterList = measurement.GetMeasurmentParameters(connectionString);
        }
    }
}
```

“Database.cshtml”

```
…

<div>

    <h1>Measurement Parameters</h1>

    Below you see all the Measurement Names registered in the Database:

    <table class="table">
        <thead>
            <tr>
                <th>MeasurementId</th>
                <th>Measurement Name</th>
                <th>Unit</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var measurement in Model.measurementParameterList)
            {
                <tr>
                    <td> @measurement.MeasurementId</td>
                    <td> @measurement.MeasurementName</td>
                    <td> @measurement.MeasurementUnit</td>
                </tr>
            }
        </tbody>
    </table>

</div>
```

Run the Application

Now we can run the Application



Measurement Parameters

Below you see all the Measurement Names registered in the Database:

MeasurementId	Measurement Name	Unit
1	Temperature	Celsius
2	Humidity	%
3	Barometric Pressure	hPa
4	Wind Speed	m/s
5	Wind Direction	Degrees
6	Rain	mm
7	Solar Radiation	W/m2

ASP.NET Core Application - © Developed by Hans-Petter Halvorsen (https://www.halvorsen.blog)

Resources

- https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/connection-string-syntax
- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog