https://www.halvorsen.blog

ASP.NET Core

Database Communication

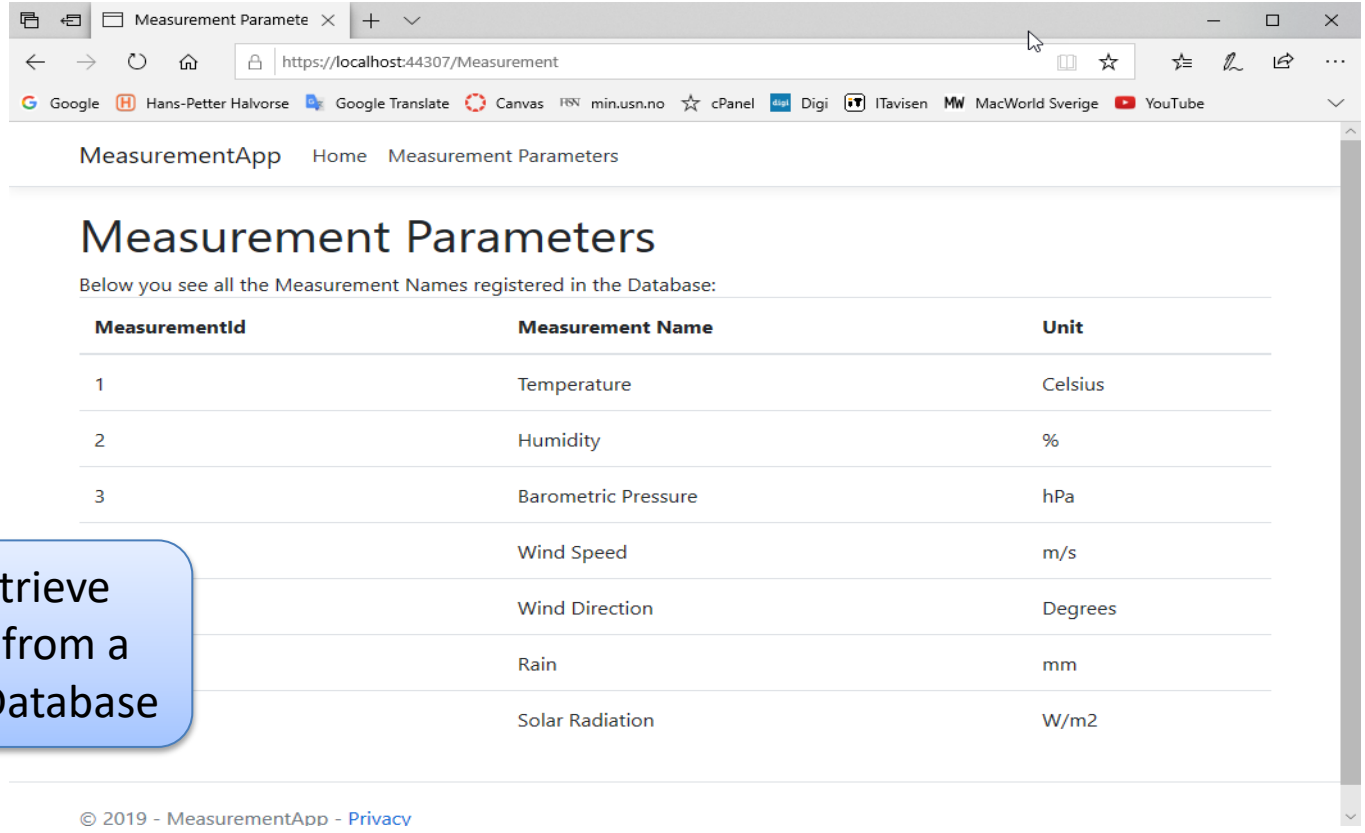
Hans-Petter Halvorsen

Introduction

- A basic **Database Communication** example will be given using **ASP.NET Core**.
- If you have never used AP.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 Web Application

The following Application will be demonstrated here:



We will retrieve these data from a SQL Server Database

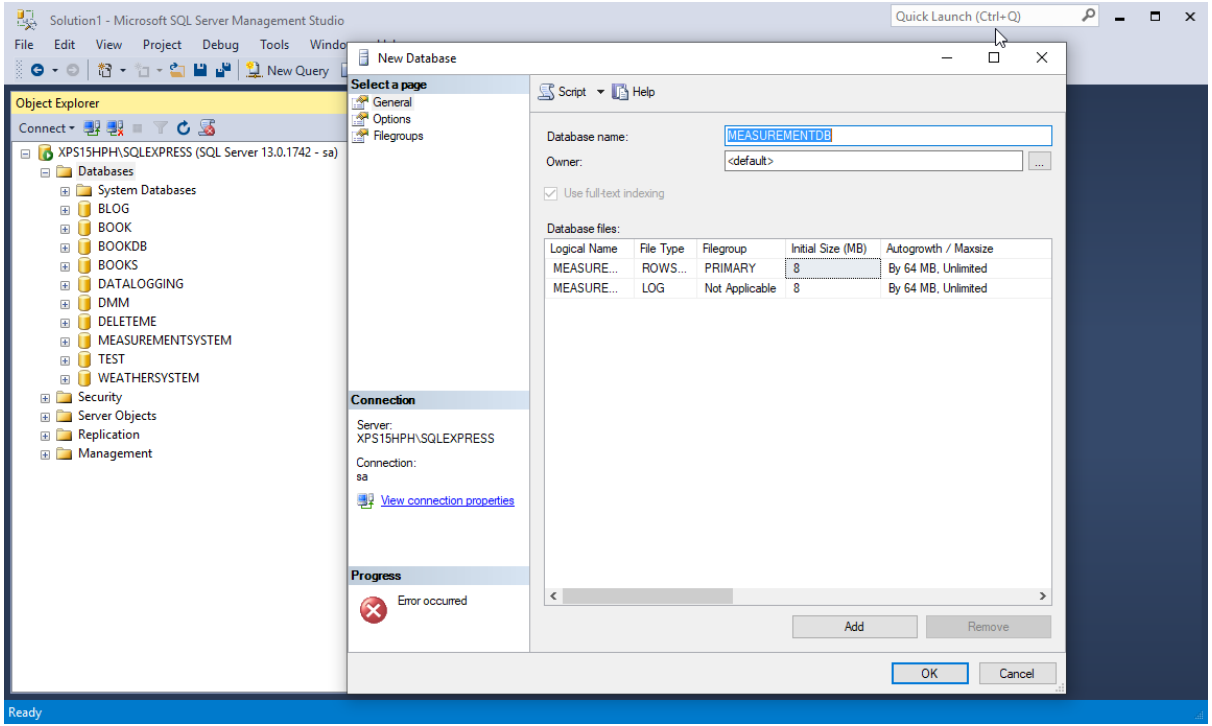
SQL Server

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.

Database

SQL Server - Create Database



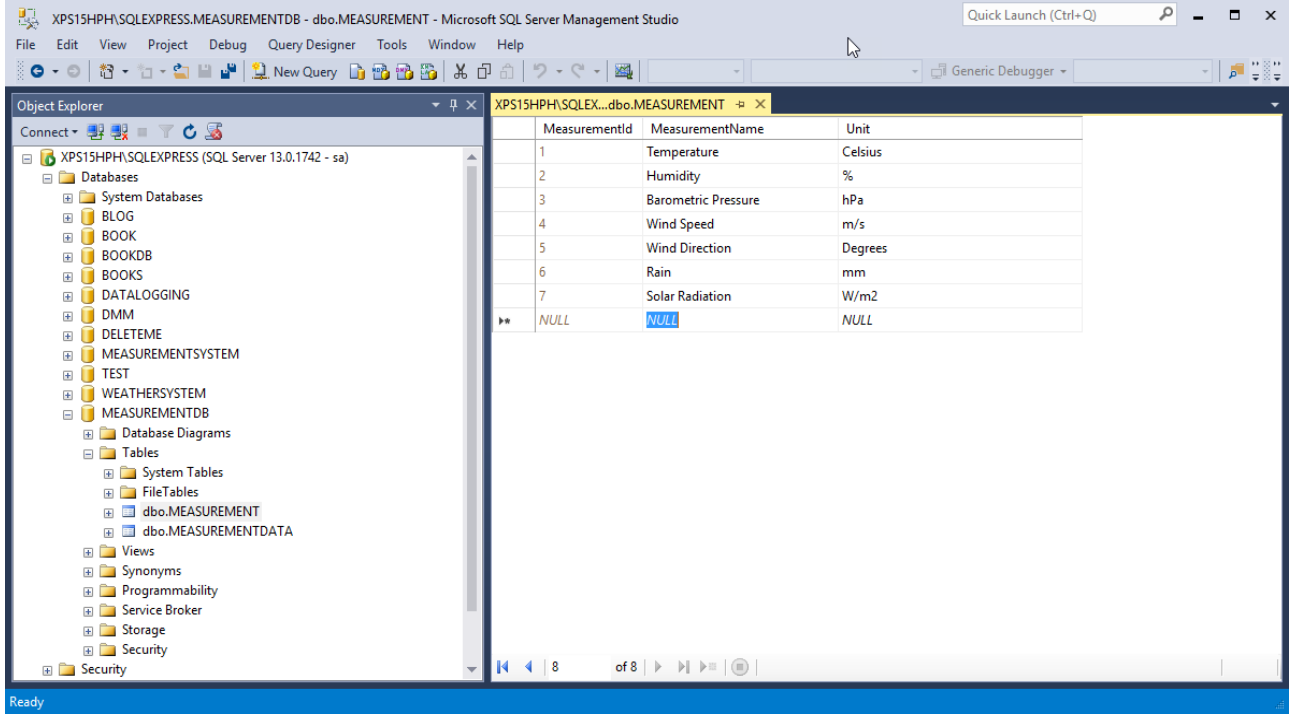
Database Table

```
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

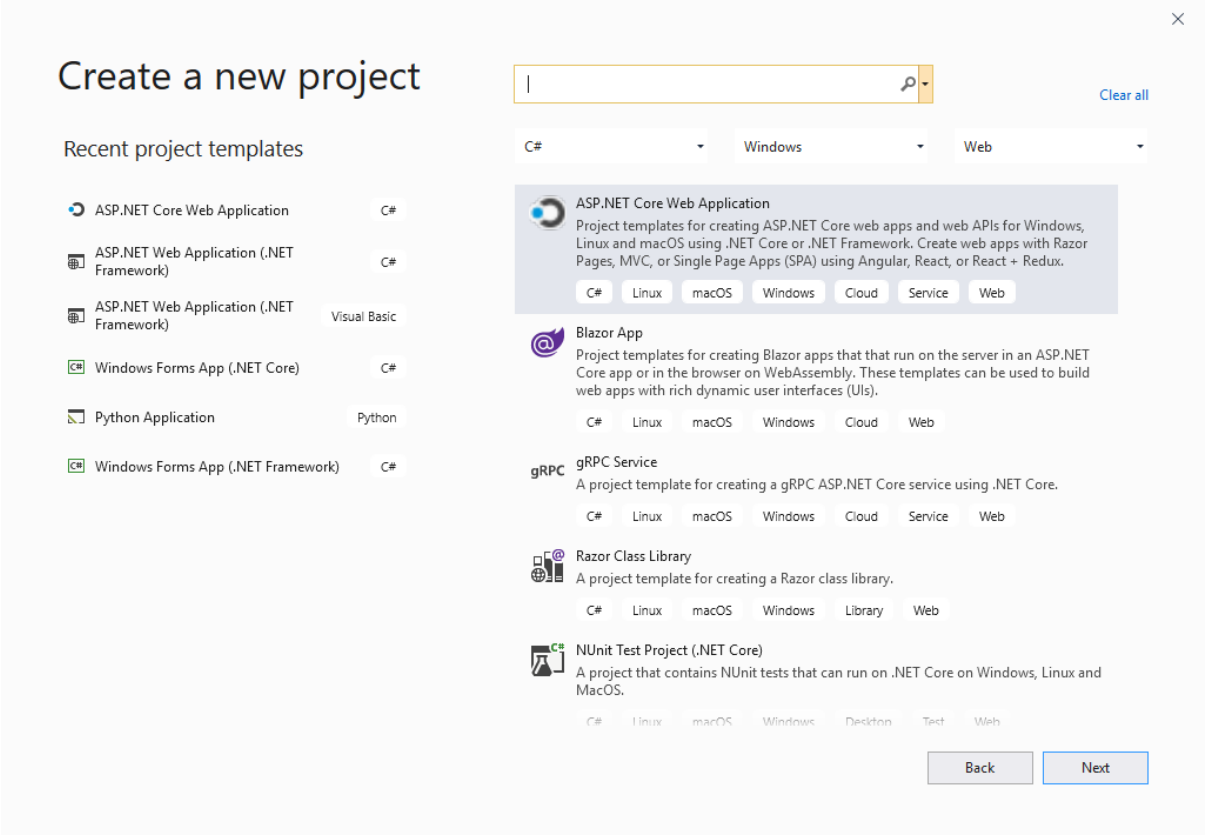
Initial Data

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:

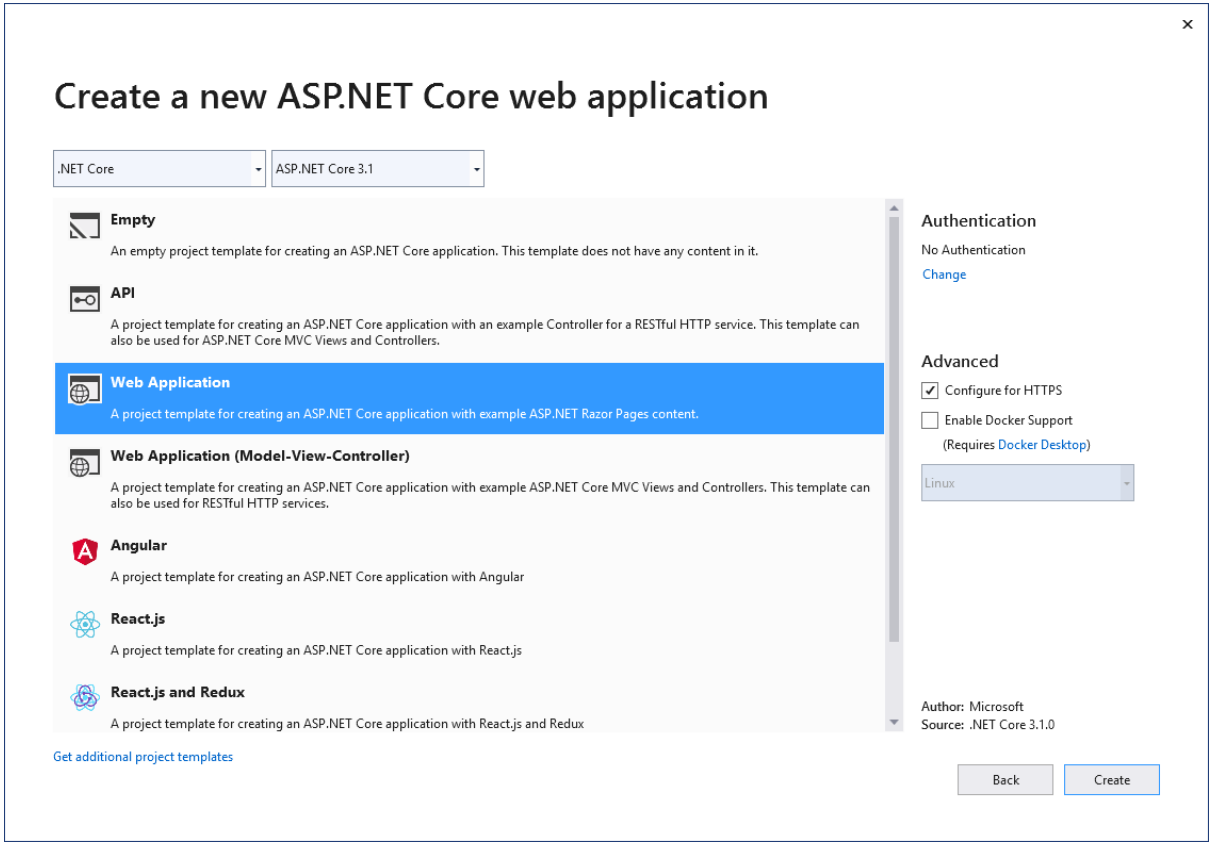


Visual Studio

Create New Project



Select ASP.NET Core Web Application



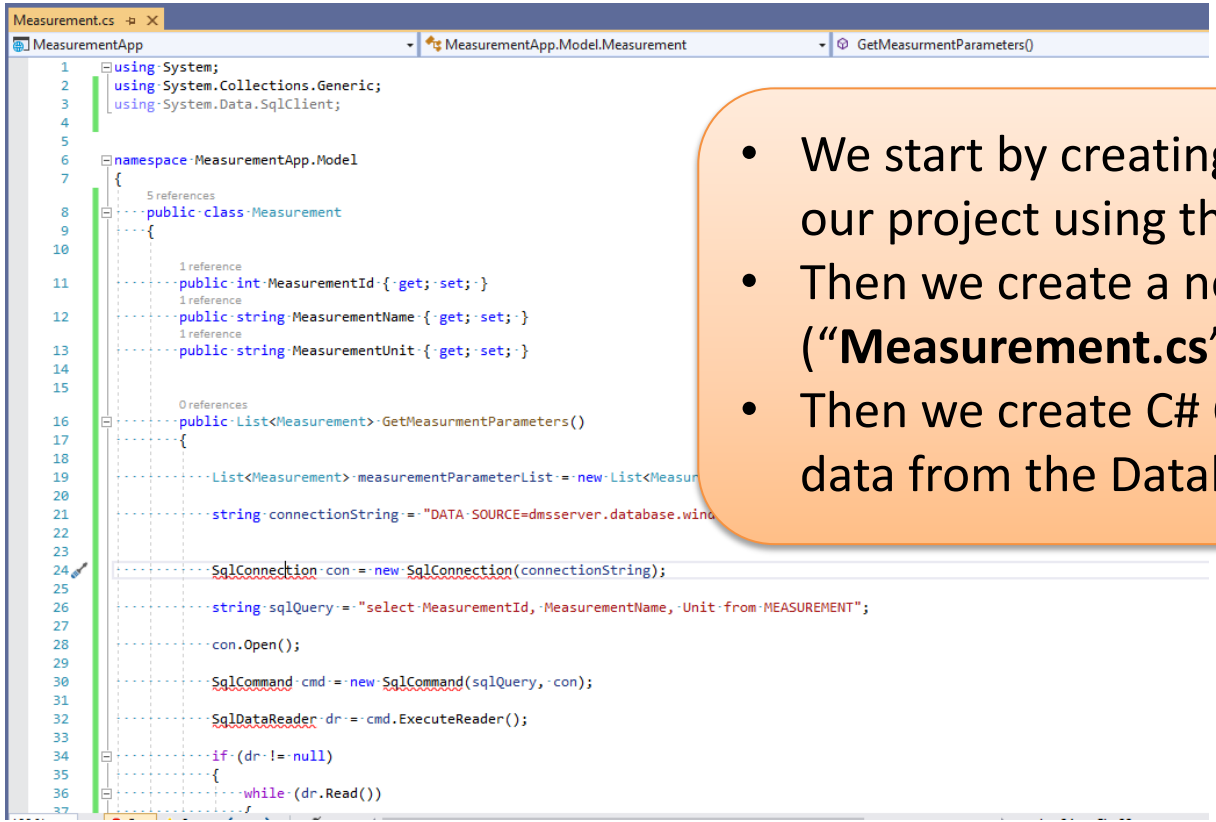
ADO.NET

ADO.NET

- ADO.NET is the core data access technology for .NET languages.
- **System.Data.SqlClient** (or the newer Microsoft.Data.SqlClient) is the provider or namespace you typically use to connect to an SQL Server.
- Typically we need to add the necessary **NuGet** package for that.

NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages.

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;
using System.Collections.Generic;
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()
        {
            List<Measurement> measurementParameterList = new List<Measurement>();

            string connectionString = "DATA SOURCE=xxx;UID=sa;PWD=xxx;DATABASE=MEASUREMENTDB";

            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;
        }
    }
}
```

Connection String

As you see the connection string to the database is hardcoded inside the "Measurement"class (A better approach will be presented later):

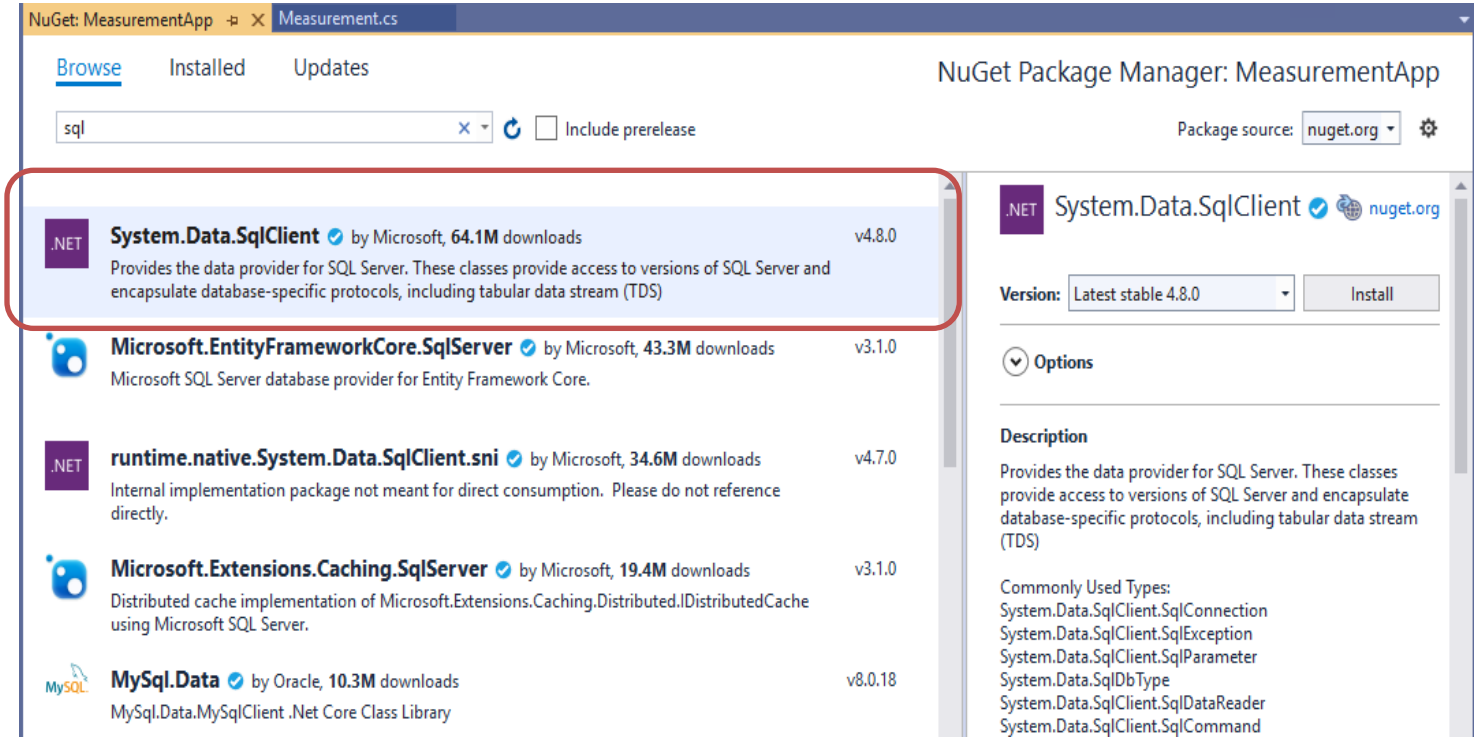
```
string connectionString = "DATA SOURCE=xxx;UID=sa;PWD=xxx;DATABASE=MEASUREMENTDB";
```

Just replace the "xxx" with the settings for your database.

- DATA SOURCE – The name of your SQL Server
- UID – your User Name that you need to connect to the SQL Server
- PWD – Your Password
- DATABASE – The Name of your Dataabase

NuGet

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

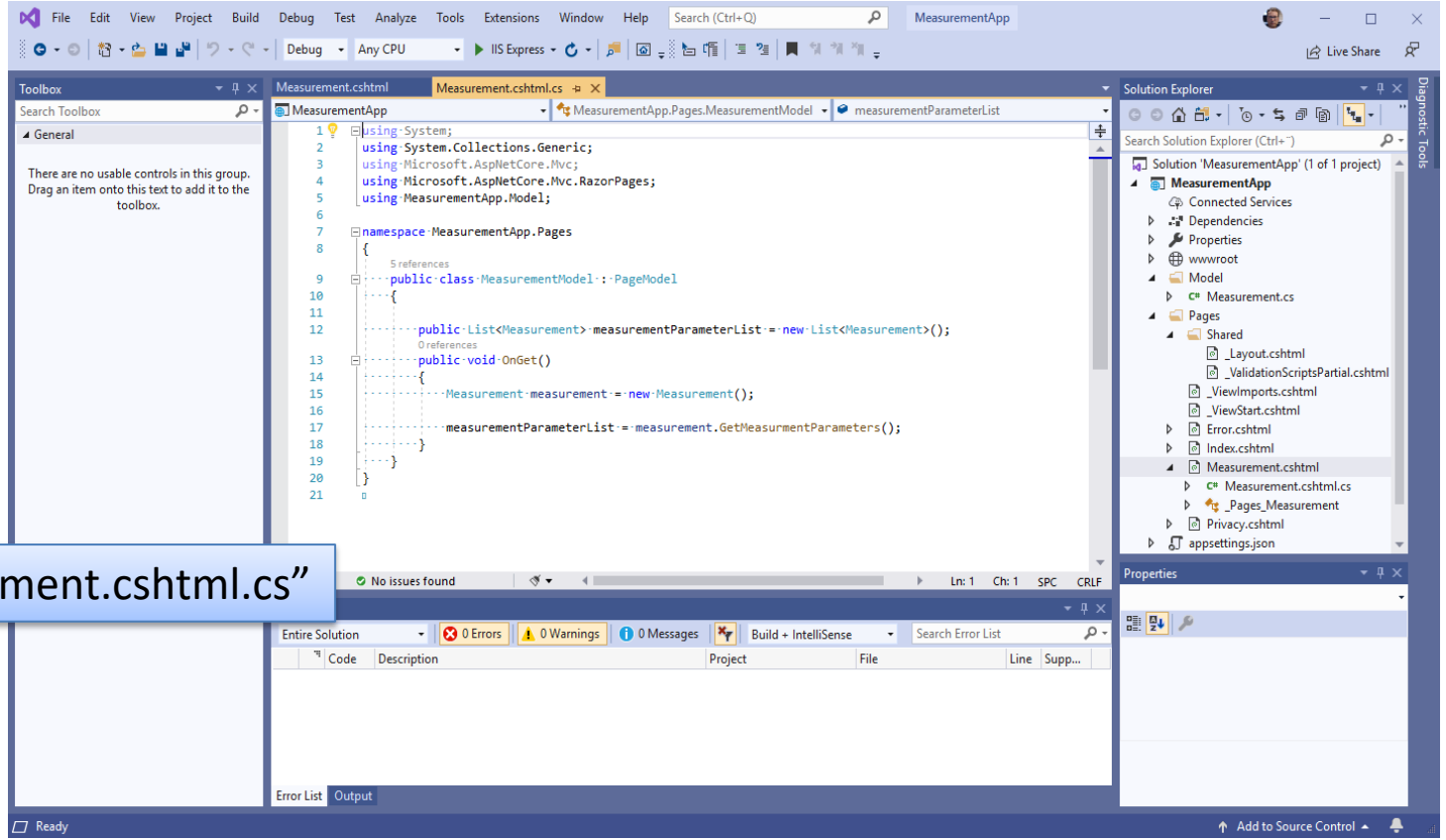


Razor

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

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

Page Model (Code behind C# File)

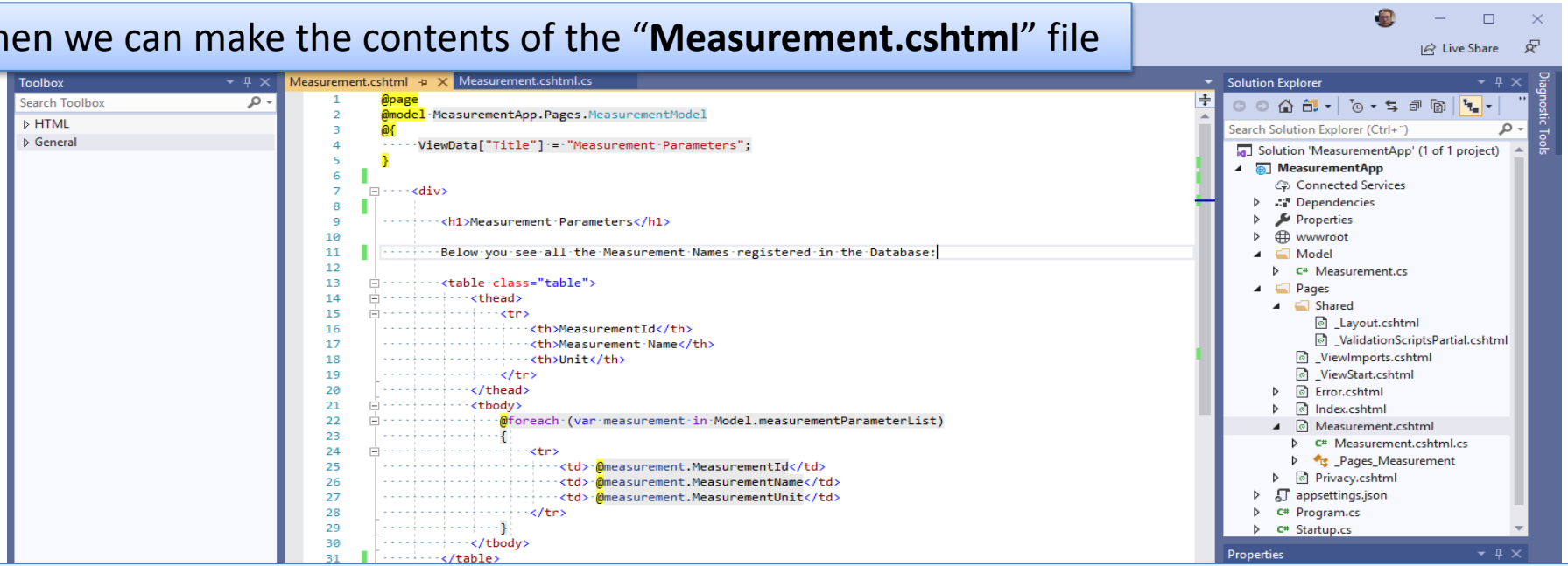


"Measurement.cshtml.cs"

```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using MeasurementApp.Model;

namespace MeasurementApp.Pages
{
    public class MeasurementModel : PageModel
    {

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

        public void OnGet()
        {
            Measurement measurement = new Measurement();

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

Razor Page

Then we can make the contents of the “**Measurement.cshtml**” file



@xxx is the Razor code. The Razor code is executed on the server before the web page is sent to the client (web browser).

We use a “foreach” to create the contents inside a HTML table.

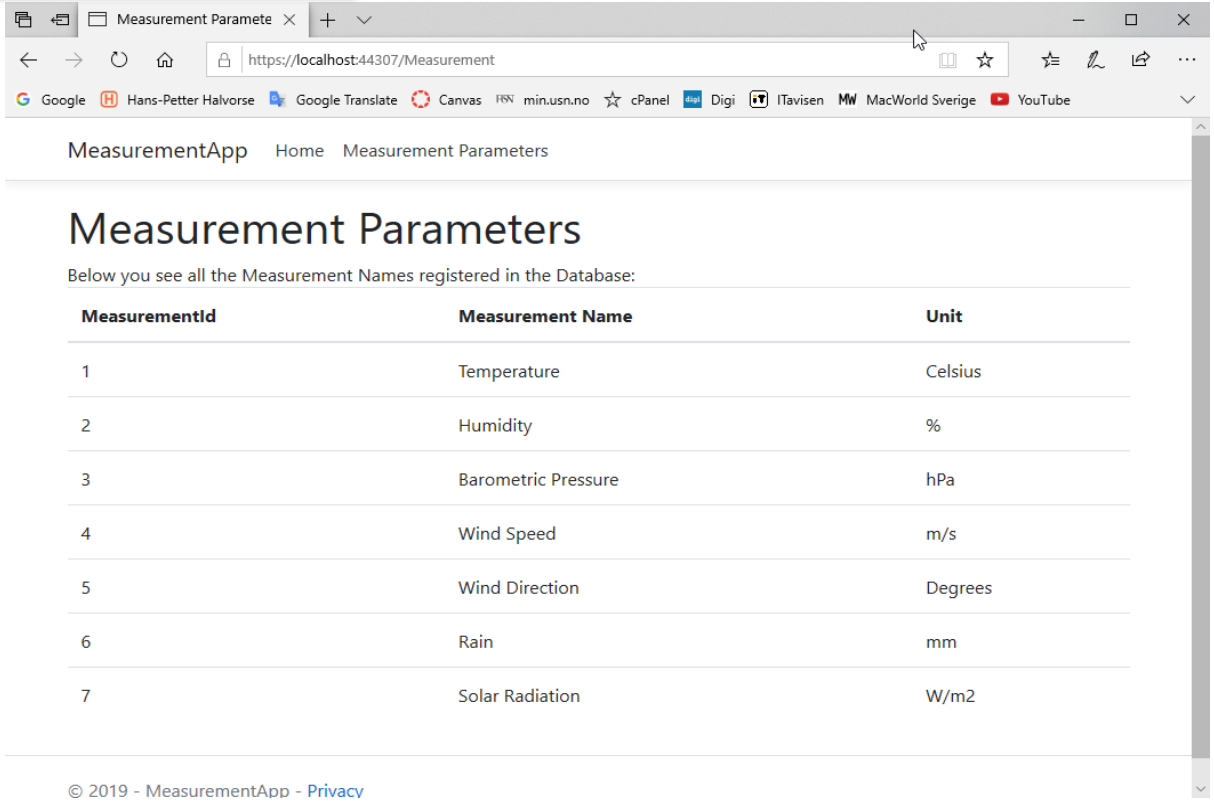
The “Model.” variable is used to retrieve data from the Page Model file (“Measurement.cshtml.cs”).
All public variables that are created in the Measurement.cshtml .cs file are available in the Measurement.cshtml file by using @Model.<variablename>.

“Measurement.cshtml.cs”

```
@page
@model MeasurementApp.Pages.MeasurementModel
@{
     ViewData["Title"] = "Measurement Parameters";
}

    <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



Connection String in appSettings.json

appSettings.json

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

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

}
```

Updated “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;
        }
    }
}
```

Startup.cs

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

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

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

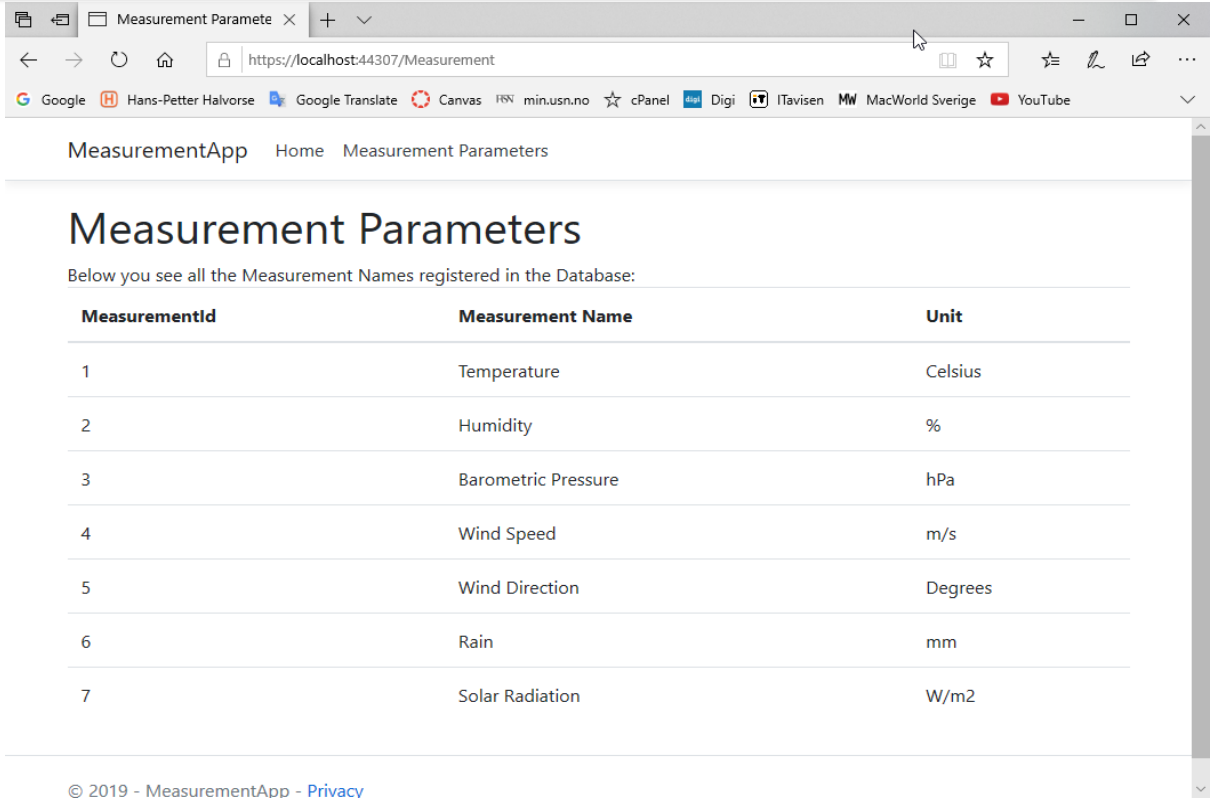
We have added:
services.AddSingleton<IConfiguration>(Configuration);

We need to update “Measurement.cshtml.cs ”

```
using System;
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Configuration;
using MeasurementApp.Model;

namespace MeasurementApp.Pages
{
    public class MeasurementModel : PageModel
    {
        readonly IConfiguration _configuration;

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

        public string connectionString;

        public MeasurementModel(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        public void OnGet()
        {
            Measurement measurement = new Measurement();

            connectionString = _configuration.GetConnectionString("ConnectionString");

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

Run the Application

Now we can run the application. The result should be the same as before



Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog