

<https://www.halvorsen.blog>



# ASP.NET Core

# Database Communication

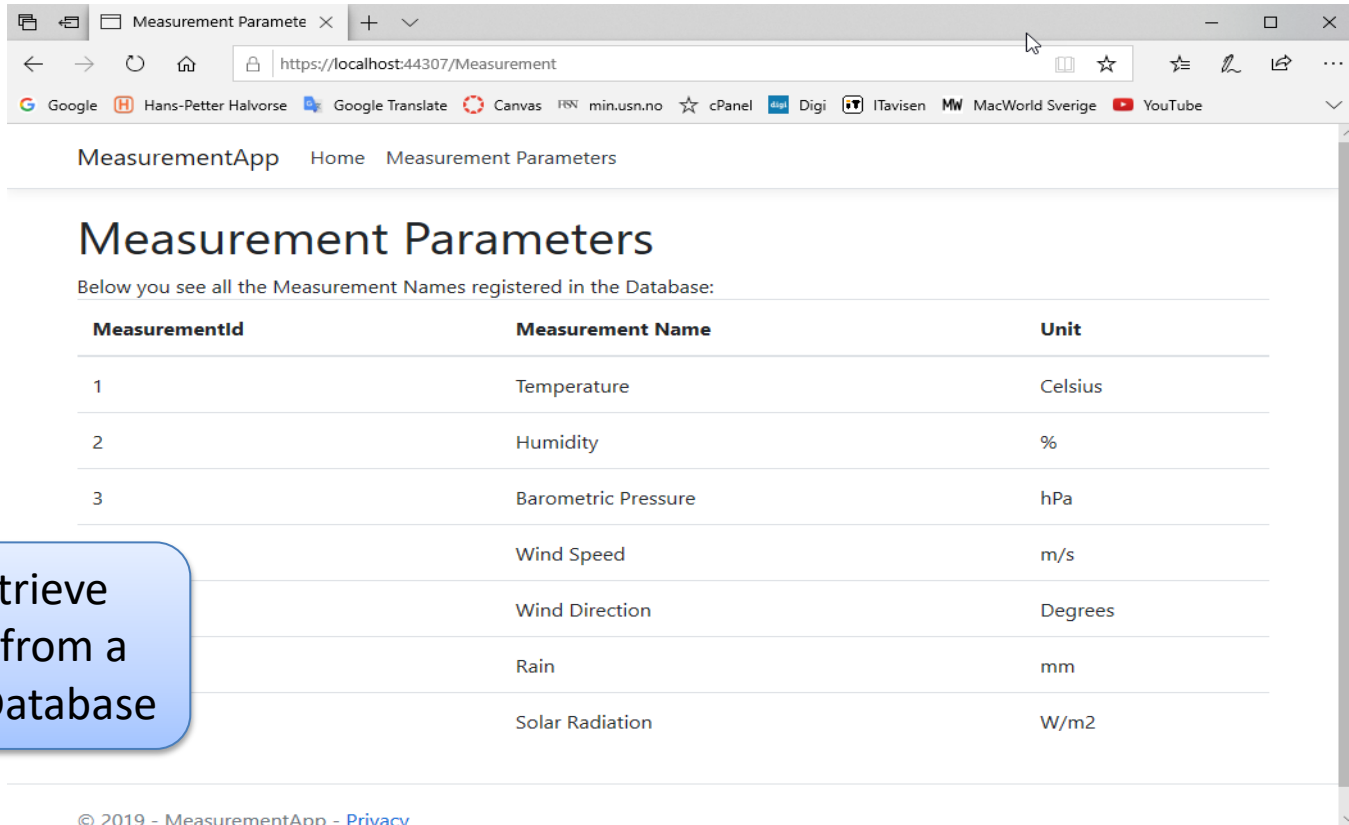
Hans-Petter Halvorsen

# 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:



MeasurementApp Home Measurement Parameters

## 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
	Wind Speed	m/s
	Wind Direction	Degrees
	Rain	mm
	Solar Radiation	W/m2

© 2019 - MeasurementApp - Privacy

We will retrieve these data from a SQL Server Database

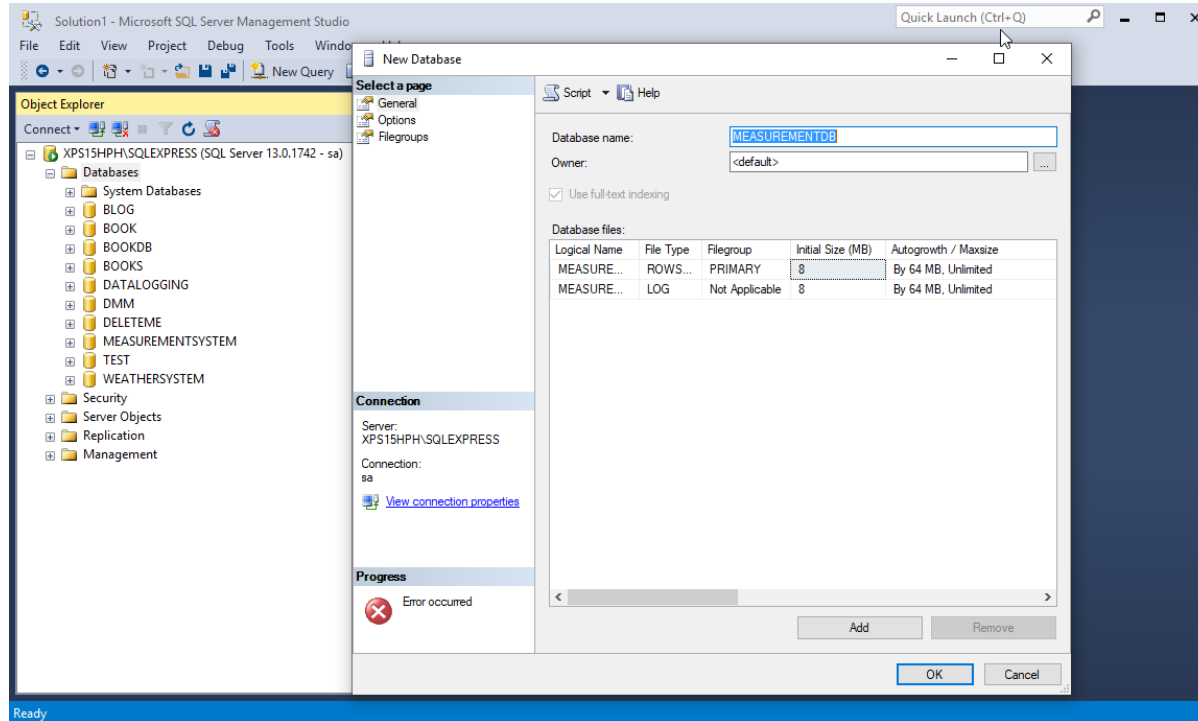
# SQL Server

# 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

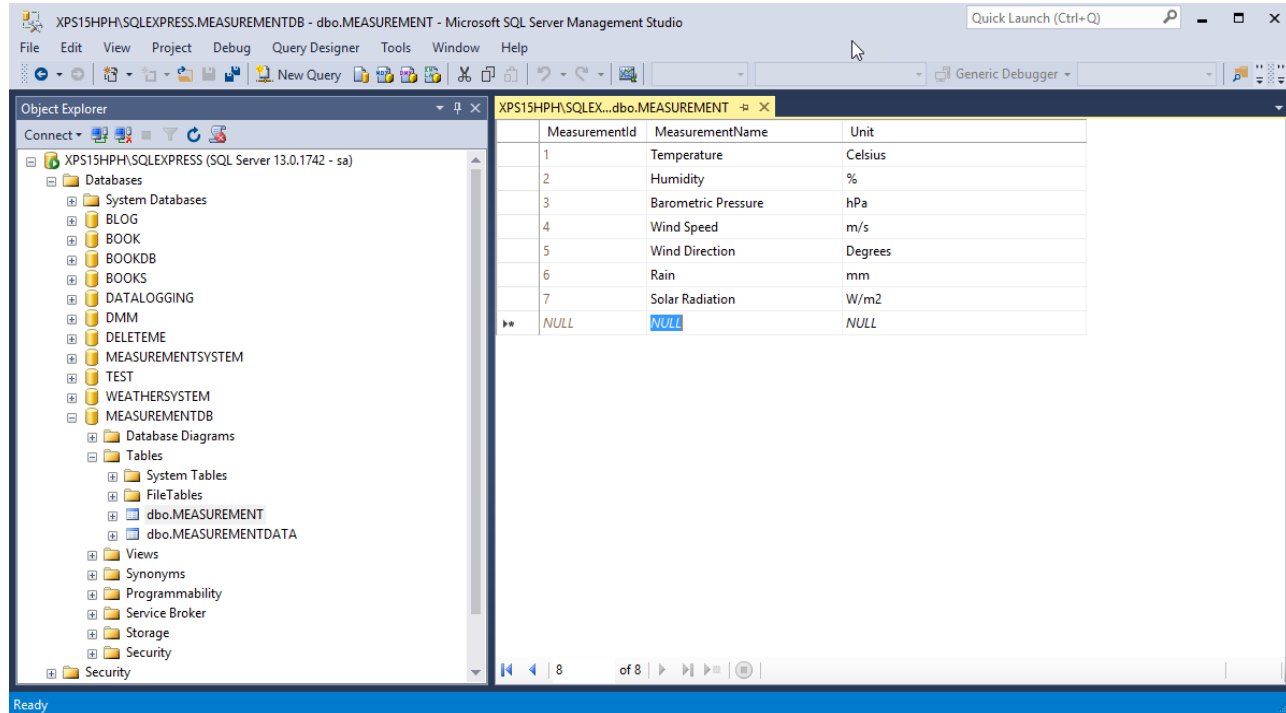
```
CREATE TABLE [MEASUREMENT]
(
    [MeasurementId]    int NOT NULL IDENTITY ( 1,1 ) Primary Key,
    [MeasurementName]  varchar(100) NOT NULL UNIQUE,
    [Unit]              varchar(50)  NULL
)
go
```

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:



The screenshot displays the Microsoft SQL Server Management Studio interface. The title bar indicates the current context is 'XPS15HPH\SQLEXPRESS.MEASUREMENTDB - dbo.MEASUREMENT'. The Object Explorer on the left shows the database structure, with 'dbo.MEASUREMENT' selected. The main window displays the data in the 'MEASUREMENT' table, which has three columns: 'MeasurementId', 'MeasurementName', and 'Unit'. The data is as follows:

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

Visual Studio

# Create New Project

Create a new project

Recent project templates

- ASP.NET Core Web Application C#
- ASP.NET Web Application (.NET Framework) C#
- ASP.NET Web Application (.NET Framework) Visual Basic
- Windows Forms App (.NET Core) C#
- Python Application Python
- Windows Forms App (.NET Framework) C#

Search: | Clear all

C# Windows Web

**ASP.NET Core Web Application**  
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.  
C# Linux macOS Windows Cloud Service Web

**Blazor App**  
Project templates for creating Blazor apps that that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).  
C# Linux macOS Windows Cloud Web

**gRPC Service**  
A project template for creating a gRPC ASP.NET Core service using .NET Core.  
C# Linux macOS Windows Cloud Service Web

**Razor Class Library**  
A project template for creating a Razor class library.  
C# Linux macOS Windows Library Web

**NUnit Test Project (.NET Core)**  
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.  
C# Linux macOS Windows Desktop Test Web

Back Next

# Select ASP.NET Core Web Application

Create a new ASP.NET Core web application

.NET Core ASP.NET Core 3.1

- Empty**  
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
- API**  
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
- Web Application**  
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- Web Application (Model-View-Controller)**  
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- Angular**  
A project template for creating an ASP.NET Core application with Angular
- React.js**  
A project template for creating an ASP.NET Core application with React.js
- React.js and Redux**  
A project template for creating an ASP.NET Core application with React.js and Redux

[Get additional project templates](#)

**Authentication**  
No Authentication  
[Change](#)

**Advanced**  
 Configure for HTTPS  
 Enable Docker Support  
(Requires [Docker Desktop](#))  
Linux

Author: Microsoft  
Source: .NET Core 3.1.0

Back Create

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.

# Create Database Class

```
Measurement.cs -> X
MeasurementApp
  MeasurementApp.Model.Measurement
    GetMeasurementParameters()
1  using System;
2  using System.Collections.Generic;
3  using System.Data.SqlClient;
4
5
6  namespace MeasurementApp.Model
7  {
8      public class Measurement
9      {
10
11         public int MeasurementId { get; set; }
12         public string MeasurementName { get; set; }
13         public string MeasurementUnit { get; set; }
14
15
16         public List<Measurement> GetMeasurementParameters()
17         {
18
19             List<Measurement> measurementParameterList = new List<Measurement>();
20
21             string connectionString = "DATA SOURCE=dmsserver.database.windows.net;
22             USER ID=dmsserver; PASSWORD=dmsserver;";
23
24             SqlConnection con = new SqlConnection(connectionString);
25
26             string sqlQuery = "select MeasurementId, MeasurementName, Unit from MEASUREMENT";
27
28             con.Open();
29
30             SqlCommand cmd = new SqlCommand(sqlQuery, con);
31
32             SqlDataReader dr = cmd.ExecuteReader();
33
34             if (dr != null)
35             {
36                 while (dr.Read())
37             }
```

- 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> GetMeasurementParameters ()
        {
            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 measurementParameter = new Measurement();

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

                    measurementParameterList.Add(measurementParameter);
                }
            }
            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 Database

# NuGet

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

The screenshot shows the NuGet Package Manager interface for a project named 'MeasurementApp'. The search bar contains 'sql'. The results list several packages, with 'System.Data.SqlClient' highlighted by a red box. The details panel on the right shows the selected package, its version (4.8.0), and an 'Install' button. The description and commonly used types are also visible.

Package Name	Author	Downloads	Version
<b>System.Data.SqlClient</b>	Microsoft	64.1M	v4.8.0
<b>Microsoft.EntityFrameworkCore.SqlServer</b>	Microsoft	43.3M	v3.1.0
<b>runtime.native.System.Data.SqlClient.sni</b>	Microsoft	34.6M	v4.7.0
<b>Microsoft.Extensions.Caching.SqlServer</b>	Microsoft	19.4M	v3.1.0
<b>MySql.Data</b>	Oracle	10.3M	v8.0.18

**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** 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)

**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

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)

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Shows the code for `Measurement.cshtml.cs`. The code includes using statements for `System`, `System.Collections.Generic`, `Microsoft.AspNetCore.Mvc`, and `MeasurementApp.Model`. It defines a `MeasurementApp.Pages` namespace containing a `PageModel` class with a `measurementParameterList` property and an `OnGet` method.
- Solution Explorer:** Shows the project structure for `MeasurementApp`, including `Pages` and `Measurement.cshtml`.
- Properties Window:** Shows the properties of the selected file.
- Output Window:** Shows the error list with 0 errors, 0 warnings, and 0 messages.

```
1 using System;
2 using System.Collections.Generic;
3 using Microsoft.AspNetCore.Mvc;
4 using Microsoft.AspNetCore.Mvc.RazorPages;
5 using MeasurementApp.Model;
6
7 namespace MeasurementApp.Pages
8 {
9     public class MeasurementModel : PageModel
10    {
11    }
12    public List<Measurement> measurementParameterList = new List<Measurement>();
13    public void OnGet()
14    {
15        Measurement measurement = new Measurement();
16        measurementParameterList = measurement.GetMeasurementParameters();
17    }
18 }
19
20
21
```

“Measurement.cshtml.cs”

Code	Description	Project	File	Line	Supp...
------	-------------	---------	------	------	---------

```
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.GetMeasurementParameters();
        }
    }
}
```

# Razor Page

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

The screenshot displays the Visual Studio IDE with the following components:

- Toolbox:** Shows HTML and General controls.
- Code Editor:** Contains the following Razor code:

```
1 @page
2 @model MeasurementApp.Pages.MeasurementModel
3 @{
4     ViewData["Title"] = "Measurement Parameters";
5 }
6
7 <div>
8
9     <h1>Measurement Parameters</h1>
10
11     <p>Below you see all the Measurement Names registered in the Database:</p>
12
13     <table class="table">
14         <thead>
15             <tr>
16                 <th>MeasurementId</th>
17                 <th>Measurement Name</th>
18                 <th>Unit</th>
19             </tr>
20         </thead>
21         <tbody>
22             @foreach (var measurement in Model.measurementParameterList)
23             {
24                 <tr>
25                     <td>@measurement.MeasurementId</td>
26                     <td>@measurement.MeasurementName</td>
27                     <td>@measurement.MeasurementUnit</td>
28                 </tr>
29             }
30         </tbody>
31     </table>
```
- Solution Explorer:** Shows the project structure for 'Solution 'MeasurementApp' (1 of 1 project)'. The 'Pages' folder is expanded, showing 'Measurement.cshtml' selected.

@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:

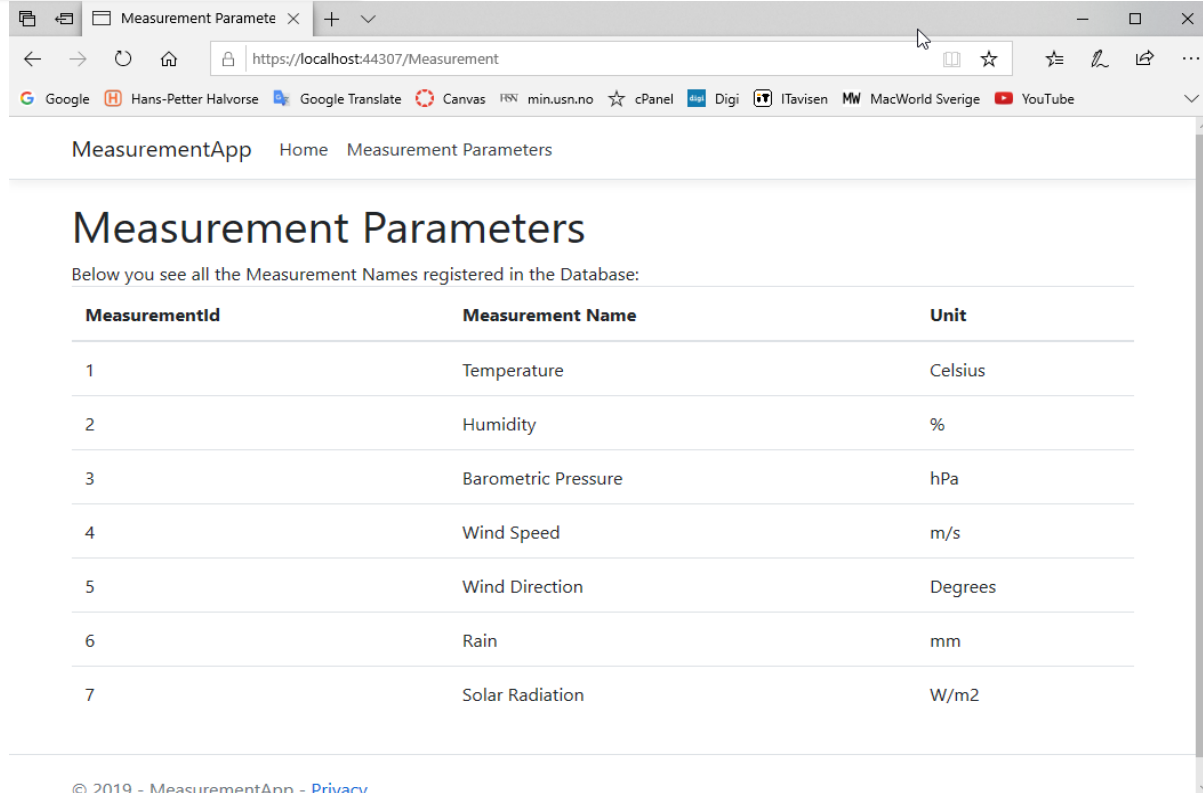
```
<table class="table">
    <thead>
        <tr>
            <th>MeasurementId</th>
            <th>Measurement Name</th>
            <th>Unit</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var measurement in Model.measurementParameterList)
        {
            <tr>
               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



MeasurementApp Home Measurement Parameters

## 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

© 2019 - MeasurementApp - [Privacy](#)

# 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;
        }

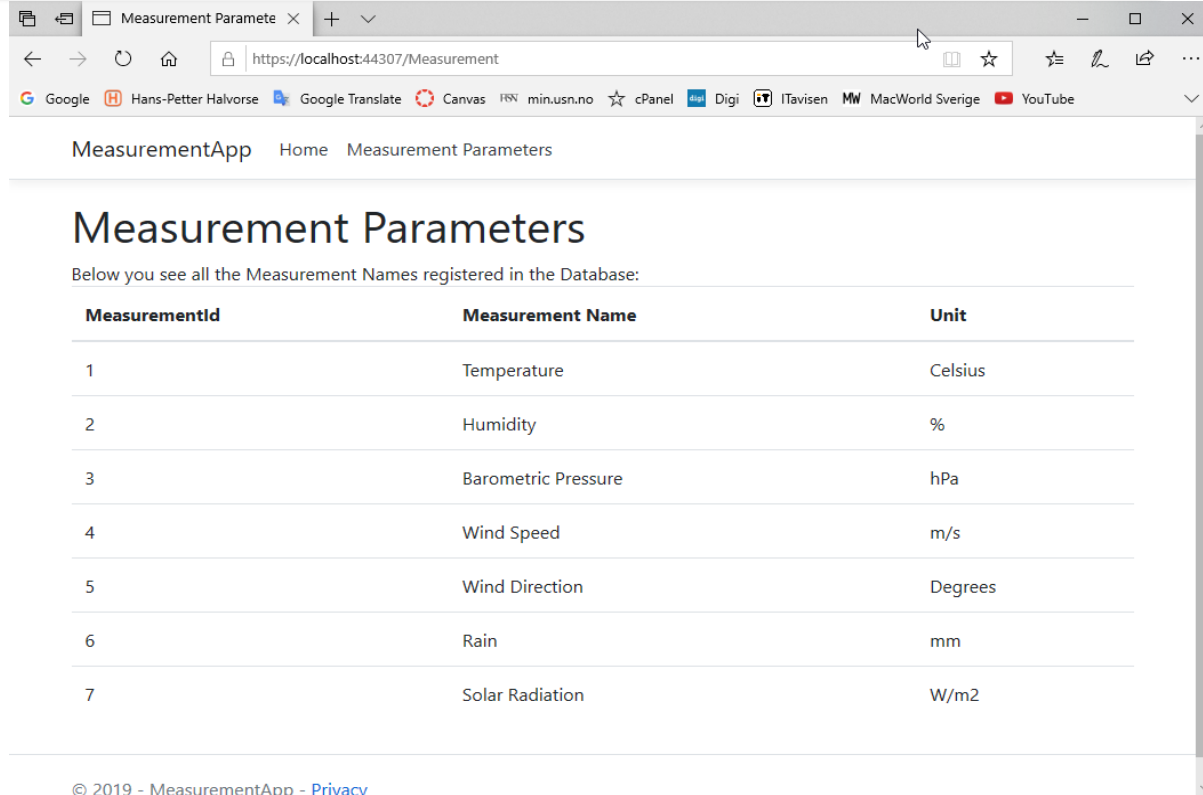
        public void OnGet()
        {
            Measurement measurement = new Measurement();

            connectionString = _configuration.GetConnectionString("ConnectionString");

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

# Run the Application

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



MeasurementApp Home Measurement Parameters

## 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

© 2019 - MeasurementApp - [Privacy](#)

# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

