

<https://www.halvorsen.blog>



ASP.NET Core appSettings.json

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>

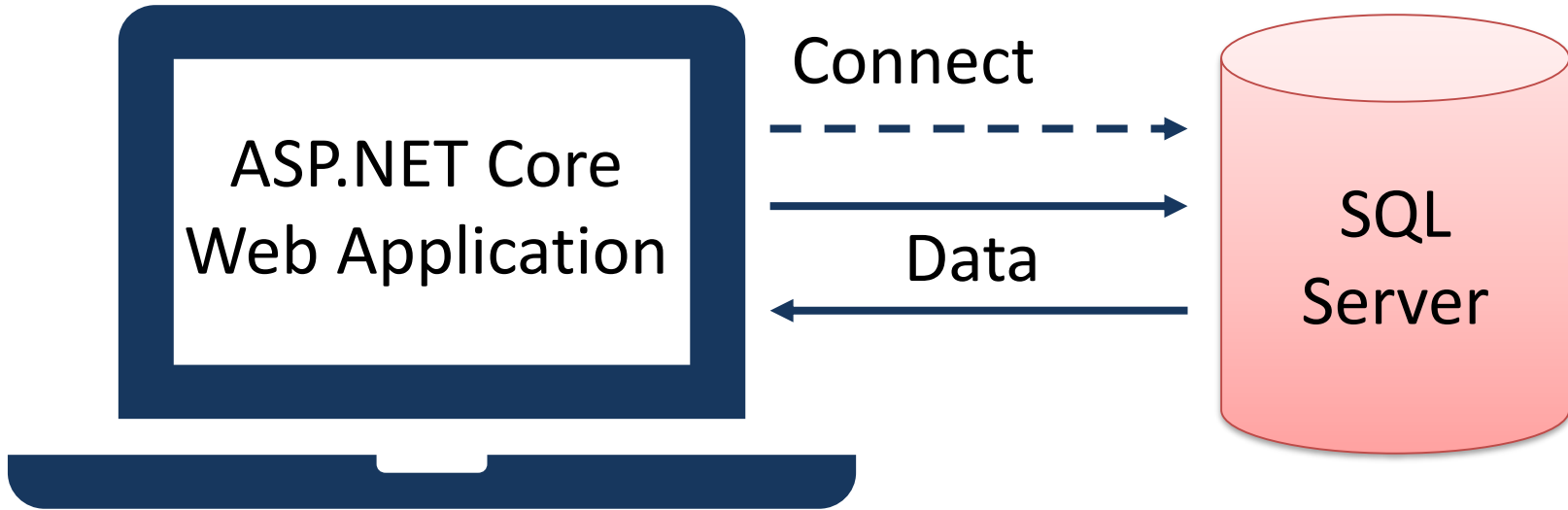


Connection String

Hans-Petter Halvorsen

Connection String

Connection String



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

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

Object Explorer

Connect

- XPS15HPH\SQLEXPRESS (SQL Server 13.0.1742 - sa)
 - Databases
 - System Databases
 - BLOG
 - BOOK
 - BOOKDB
 - BOOKS
 - DATALOGGING
 - DMM
 - DELETEME
 - MEASUREMENTSYSTEM
 - TEST
 - WEATHERSYSTEM
 - Security
 - Server Objects
 - Replication
 - Management

New Database

Select a page

- General
- Options
- Filegroups

Script Help

Database name: MEASUREMENTDB

Owner: <default>

Use full-text indexing

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

Connection

Server: XPS15HPH\SQLEXPRESS

Connection: sa

[View connection properties](#)

Progress

Error occurred

Add Remove

OK Cancel

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

Object Explorer

Connect

- XPS15HPH\SQLEXPRESS (SQL Server 13.0.1742 - sa)
 - Databases
 - System Databases
 - BLOG
 - BOOK
 - BOOKDB
 - BOOKS
 - DATALOGGING
 - DMM
 - DELETEME
 - MEASUREMENTSYSTEM
 - TEST
 - WEATHERSYSTEM
 - MEASUREMENTDB
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - dbo.MEASUREMENT
 - dbo.MEASUREMENTDATA
 - Views
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security

XPS15HPH\SQLEX...dbo.MEASUREMENT

	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

Visual Studio

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' and the package source is set to 'nuget.org'. The search results list several packages, with 'System.Data.SqlClient' highlighted by a red circle. The details panel on the right shows the package name, version (4.8.0), and an 'Install' button. The description and commonly used types are also visible.

Package Name	Author	Downloads	Version
System.Data.SqlClient	Microsoft	64.1M	v4.8.0
Microsoft.EntityFrameworkCore.SqlServer	Microsoft	43.3M	v3.1.0
runtime.native.System.Data.SqlClient.sni	Microsoft	34.6M	v4.7.0
Microsoft.Extensions.Caching.SqlServer	Microsoft	19.4M	v3.1.0
MySql.Data	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)

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;UID=sa;PWD=xxx;DATABASE=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

Demo

Connection String in
appSettings.json

ASP.NET Core Web Application

AppSettingsApp Home **Show Data** Settings

The following Application will be demonstrated here:

We will retrieve these data from a SQL Server Database

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

```
Measurement.cs → x
AppSettingsApp - | AppSettingsApp.Models.Measurement
1 using System;
2 using System.Collections.Generic;
3 using System.Data.SqlClient;
4
5 namespace AppSettingsApp.Models
6 {
7     9 references
8     public class Measurement
9     {
10         2 references
11         public int MeasurementId { get; set; }
12         2 references
13         public string MeasurementName { get; set; }
14         2 references
15         public string MeasurementUnit { get; set; }
16
17         1 reference
18         public List<Measurement> GetMeasurementParameters(string connectionString)
19         {
20
21             List<Measurement> measurementParameterList = new List<Measurement>();
22
23             SqlConnection con = new SqlConnection(connectionString);
24
25             string sqlQuery = "select MeasurementId, MeasurementName, Unit from MEASUREMENT";
26
27             con.Open();
28
29             SqlCommand cmd = new SqlCommand(sqlQuery, con);
30
31             SqlDataReader dr = cmd.ExecuteReader();
32
33             if (dr != null)
34             {
35                 while (dr.Read())
36                 {
37                     Measurement measurementParameter = new Measurement();
38
39                     measurementParameter.MeasurementId = Convert.ToInt32(dr["MeasurementId"]);
40                     measurementParameter.MeasurementName = dr["MeasurementName"].ToString();
41                     measurementParameter.MeasurementUnit = dr["Unit"].ToString();
42
43                     measurementParameterList.Add(measurementParameter);
44                 }
45             }
46         }
47     }
48 }
```

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

```
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;  
    }  
}
```

“Measurement.cs”

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

```
...  
<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

AppSettingsApp Home **Show Data** Settings

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

<https://www.halvorsen.blog>



Get Configuration Data from appSettings.json in your C# Code

Hans-Petter Halvorsen

appSettings.json

```
{  
  ...  
  "ConnectionStrings": {  
    "ConnectionString": "DATA SOURCE=xxx\\SQLEXPRESS;UID=sa;PWD=xxx;DATABASE=VOTINGSYSTEM"  
  },  
  "Company": {  
    "CompanyName": "University of South-Eastern Norway",  
    "WebSite": "https://www.usn.no/english/"  
  },  
  "Appearance": {  
    "BackColor": "warning"  
  }  
}
```

C# Code

```
string companyName;
```

```
...
```

```
companyName = _configuration.GetSection(" Company").GetValue<string>(" CompanyName ");
```

```
...
```

Demo

ASP.NET Core Web Application

AppSettingsApp Home Show Data **Settings**

Settings

Company Name: University of South-Eastern Norway

[Company Web Site](#)

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

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

“Settings.cshtml.cs”

```
...
using Microsoft.Extensions.Configuration;

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

        public string companyName;
        public string webSite;
        public string backColor;

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

        public void OnGet()
        {
            GetAppSettings();
        }

        void GetAppSettings()
        {
            companyName = _configuration.GetSection("Company").GetValue<string>("CompanyName");

            webSite = _configuration.GetSection("Company").GetValue<string>("WebSite");

            backColor = _configuration.GetSection("Appearance").GetValue<string>("BackColor");
        }
    }
}
```



```
@page
```

```
@model AppSettingsApp.Pages.SettingsModel
```

```
@{
```

```
    ViewData["Title"] = "Settings";
```

```
}
```

```
<div>
```

```
    <h1 class="text-@Model.backColor">Settings</h1>
```

```
    Company Name: @Model.companyName
```

```
    <br />
```

```
    <a href="@Model.webSite" target="_blank">Company Web Site</a>
```

```
    <br />
```

```
</div>
```

Run the Application

AppSettingsApp

Home

Show Data

Settings

Settings

Company Name: University of South-Eastern Norway

[Company Web Site](#)

Now we can run the Application

appSetting.json:

```
"CompanyName": "University of South-Eastern Norway"
```

```
"WebSite": "https://www.usn.no/english/"
```

```
"BackColor": "warning"
```

Resources

- <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>

