https://www.halvorsen.blog

SQL Server and Visual Studio

Hans-Petter Halvorsen

Contents

SQL Server and Visual Studio

Introduction

Hans-Petter Halvorsen

Introduction

- We will use **SQL Server**, which is a Database System from Microsoft.
- We will create Applications in **Visual Studio** and **C#** that communicates with the SQL Server Database.
- We will create **Windows Forms** Desktop Applications.
  - Applications that **Writes Data** to the SQL Server.
  - Applications that **Reads Data** from the SQL Server.

What is a Database?

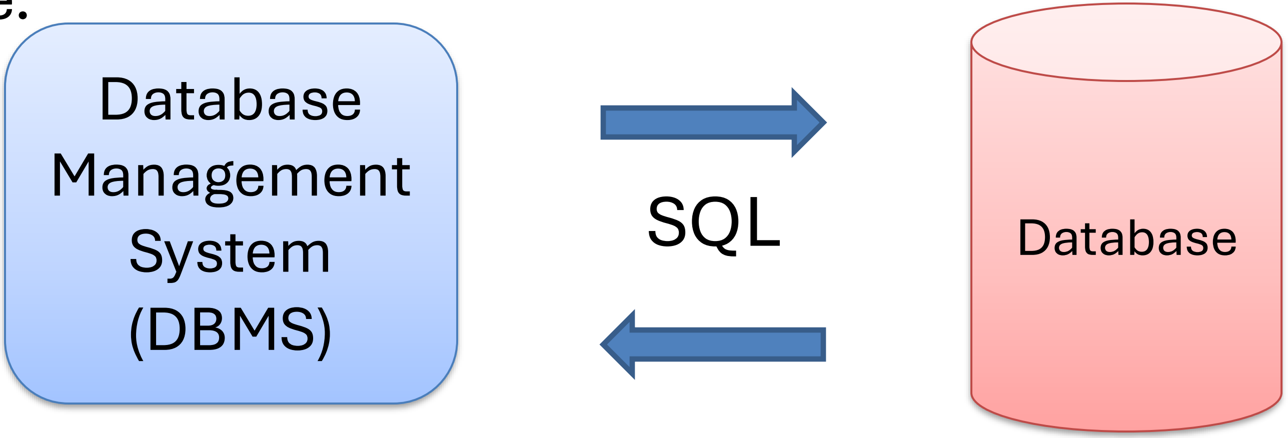
- A Database is a structured way to store lots of information.
- The information inside the database is stored in different tables.
- - "Everything" today is stored in databases!

Examples:

- Bank/Account systems
- Information in Web pages such as Facebook, Wikipedia, YouTube, etc.
- Online Web Shops
- … lots of other examples!

Database Systems

We communicate with the Database using a Database Management System (DBMS). We use the **Structured Query Language (SQL)** to communicate with the Database, i.e., Insert Data, Retrieve Data, Update Data and Delete Data from the Database.



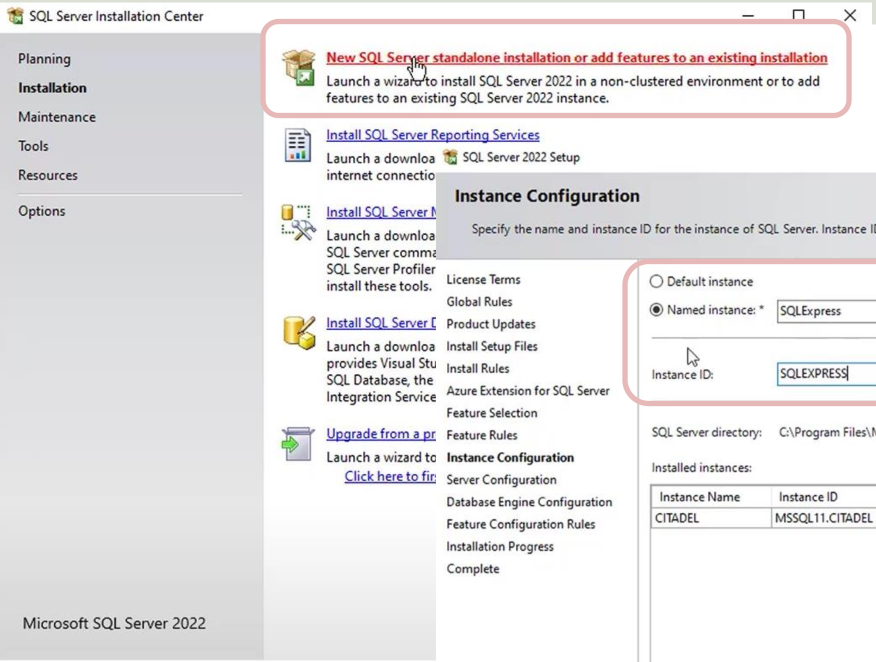
SQL – Structured Query Language

SQL Server and Visual Studio

SQL Server

Hans-Petter Halvorsen

Install **SQL Server Express**



SQL Server Installation Center

Planning
Installation
Maintenance
Tools
Resources
Options

Microsoft SQL Server 2022

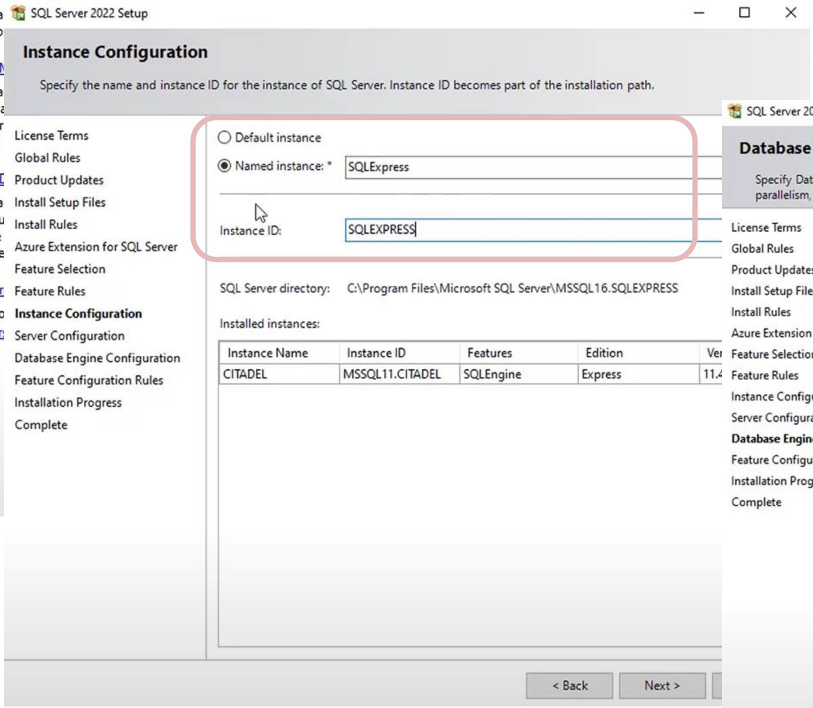
New SQL Server standalone installation or add features to an existing installation
Launch a wizard to install SQL Server 2022 in a non-clustered environment or to add features to an existing SQL Server 2022 instance.

Install SQL Server Reporting Services
Launch a downloa
internet connectio

Install SQL Server M
Launch a downloa
SQL Server comma
SQL Server Profiler
install these tools.

Install SQL Server D
Launch a downloa
provides Visual Stu
SQL Database, the
Integration Service

Upgrade from a pr
Launch a wizard to
Click here to fir



SQL Server 2022 Setup

Instance Configuration

Specify the name and instance ID for the instance of SQL Server. Instance ID becomes part of the installation path.

License Terms
Global Rules
Product Updates
Install Setup Files
Install Rules
Azure Extension for SQL Server
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Installation Progress
Complete

Default instance [✓ Named instance: *] SQLExpress

Instance ID: SQLEXPRESS

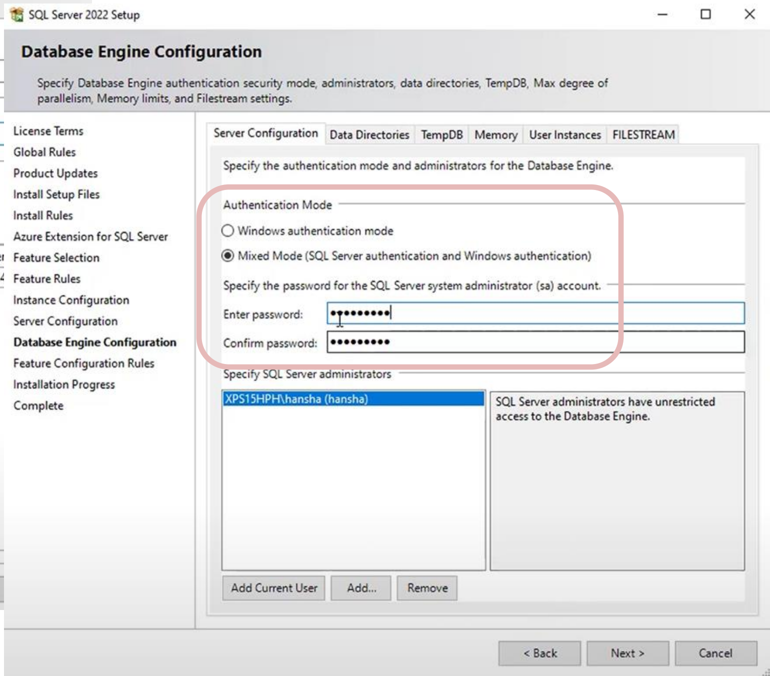
SQL Server directory: C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS

Installed instances:

Instance Name	Instance ID	Features	Edition	Ver
CITADEL	MSSQL11.CITADEL	SQLEngine	Express	11.4

< Back | Next >

Mixed Mode is recommended



SQL Server 2022 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories, TempDB, Max degree of parallelism, Memory limits, and Filestream settings.

License Terms
Global Rules
Product Updates
Install Setup Files
Install Rules
Azure Extension for SQL Server
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Feature Configuration Rules
Installation Progress
Complete

Server Configuration | Data Directories | TempDB | Memory | 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

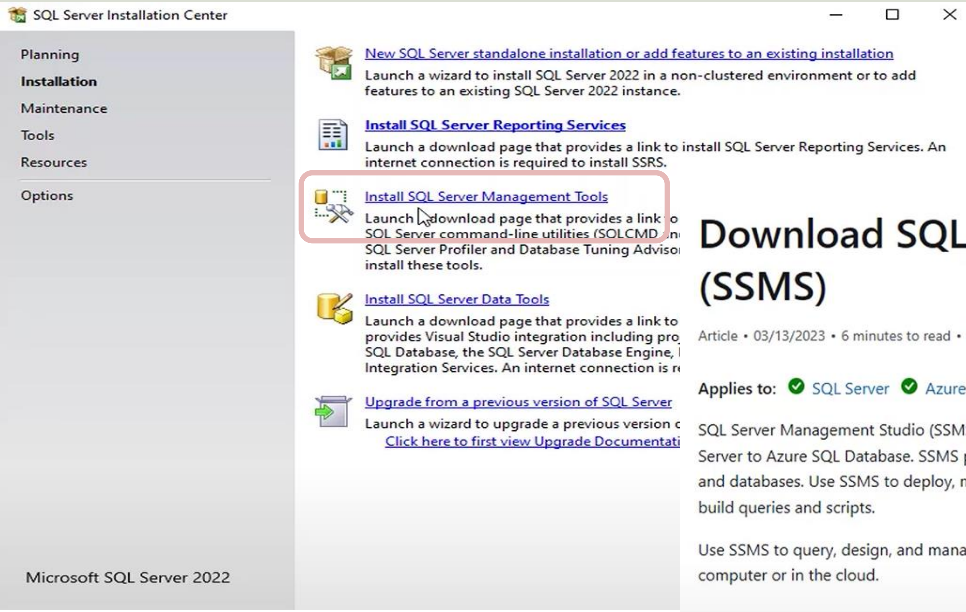
XPS15HPH\hansha (hansha)

SQL Server administrators have unrestricted access to the Database Engine.

Add Current User | Add... | Remove

< Back | Next > | Cancel

Install **SQL Server Management Studio**



SQL Server Installation Center

New SQL Server standalone installation or add features to an existing installation
Launch a wizard to install SQL Server 2022 in a non-clustered environment or to add features to an existing SQL Server 2022 instance.

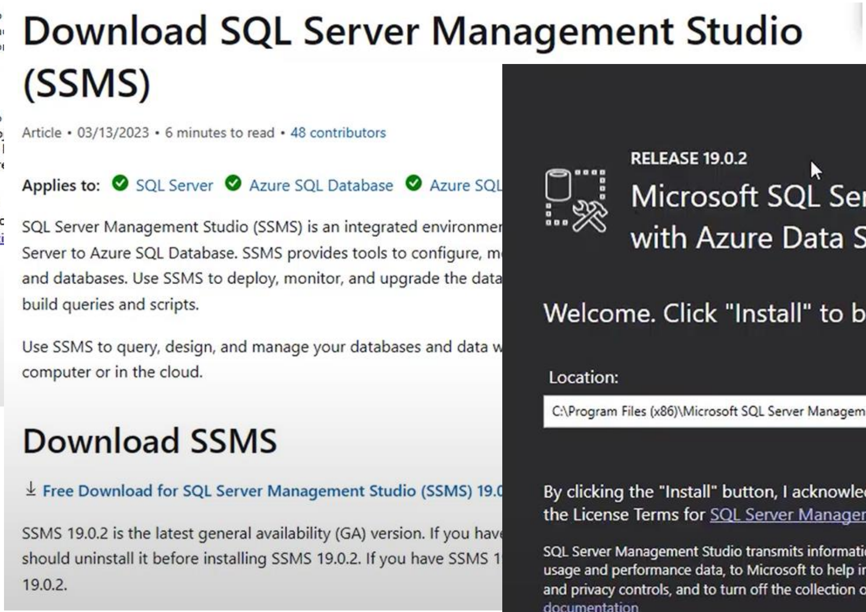
Install SQL Server Reporting Services
Launch a download page that provides a link to install SQL Server Reporting Services. An internet connection is required to install SSRS.

Install SQL Server Management Tools
Launch a download page that provides a link to
SQL Server command-line utilities (SQLCMD an
SQL Server Profiler and Database Tuning Adviso
install these tools.

Install SQL Server Data Tools
Launch a download page that provides a link to
provides Visual Studio integration including pro
SQL Database, the SQL Server Database Engine,
Integration Services. An internet connection is r

Upgrade from a previous version of SQL Server
Launch a wizard to upgrade a previous version c
Click here to first view Upgrade Documentati

Microsoft SQL Server 2022



Download SQL Server Management Studio (SSMS)

Article • 03/13/2023 • 6 minutes to read • 48 contributors

Applies to: ✓ SQL Server ✓ Azure SQL Database ✓ Azure SQL

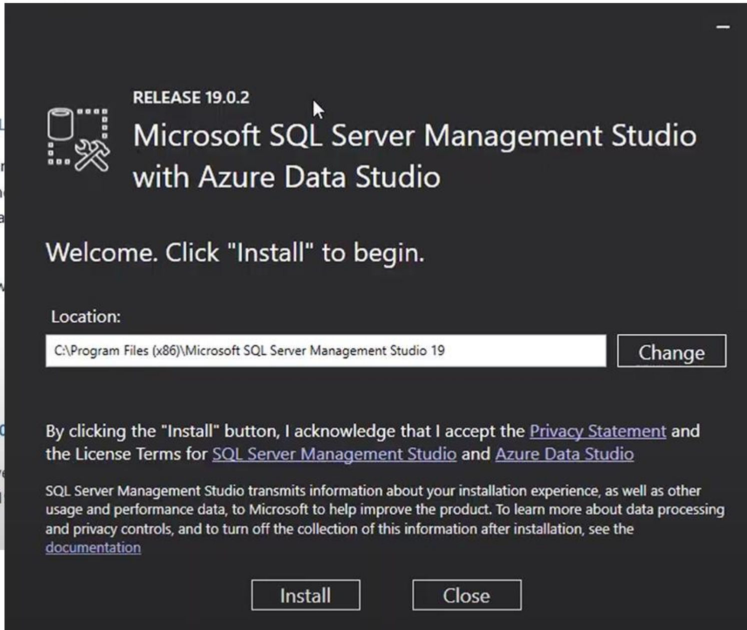
SQL Server Management Studio (SSMS) is an integrated environmer
Server to Azure SQL Database. SSMS provides tools to configure, m
and databases. Use SSMS to deploy, monitor, and upgrade the data
build queries and scripts.

Use SSMS to query, design, and manage your databases and data w
computer or in the cloud.

Download SSMS

⤓ Free Download for SQL Server Management Studio (SSMS) 19.0

SSMS 19.0.2 is the latest general availability (GA) version. If you have
should uninstall it before installing SSMS 19.0.2. If you have SSMS 1
19.0.2.



RELEASE 19.0.2

Microsoft SQL Server Management Studio with Azure Data Studio

Welcome. Click "Install" to begin.

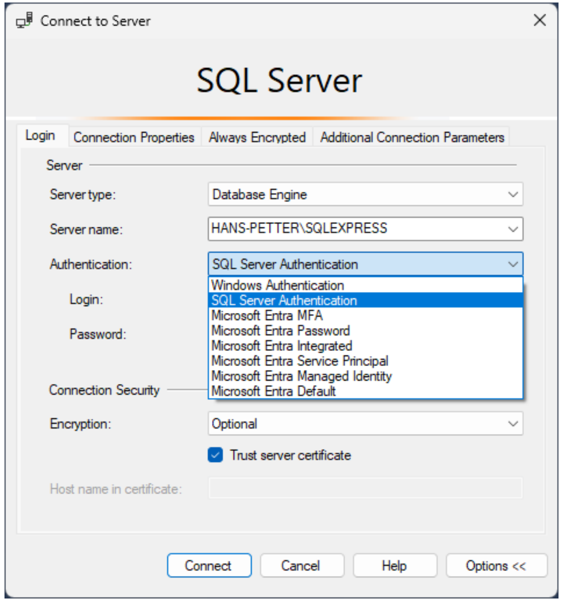
Location:

C:\Program Files (x86)\Microsoft SQL Server Management Studio 19

By clicking the "Install" button, I acknowledge that I accept the Privacy Statement and the License Terms for SQL Server Management Studio and Azure Data Studio

SQL Server Management Studio transmits information about your installation experience, as well as other usage and performance data, to Microsoft to help improve the product. To learn more about data processing and privacy controls, and to turn off the collection of this information after installation, see the documentation

Login to **SQL Server Management Studio**



Connect to Server

SQL Server

Login | Connection Properties | Always Encrypted | Additional Connection Parameters

Server

Server type: Database Engine

Server name: HANS-PETTER\SQLEXPRESS

Authentication: SQL Server Authentication

- Windows Authentication
- SQL Server Authentication
- Microsoft Entra MFA
- Microsoft Entra Password
- Microsoft Entra Integrated
- Microsoft Entra Service Principal
- Microsoft Entra Managed Identity
- Microsoft Entra Default

Login:

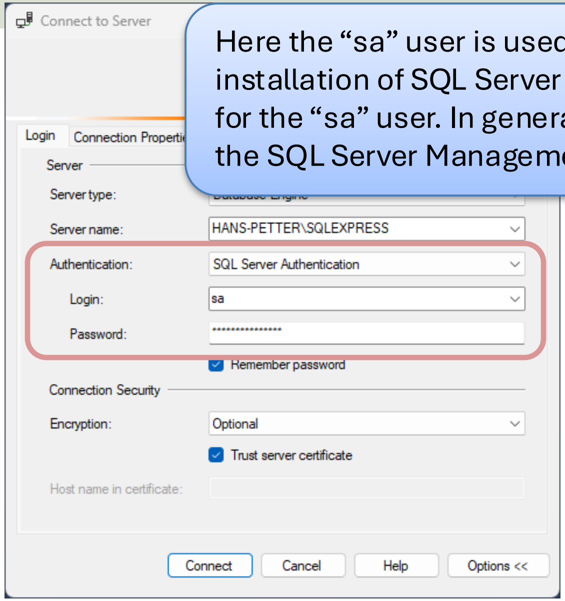
Password:

Connection Security

Encryption: Optional

☑ Trust server certificate

Host name in certificate:



Connect to Server

Login | Connection Properti

Server

Server type: Database Engine

Server name: HANS-PETTER\SQLEXPRESS

Authentication: SQL Server Authentication

Login: sa

Password: ***************

☑ Remember password

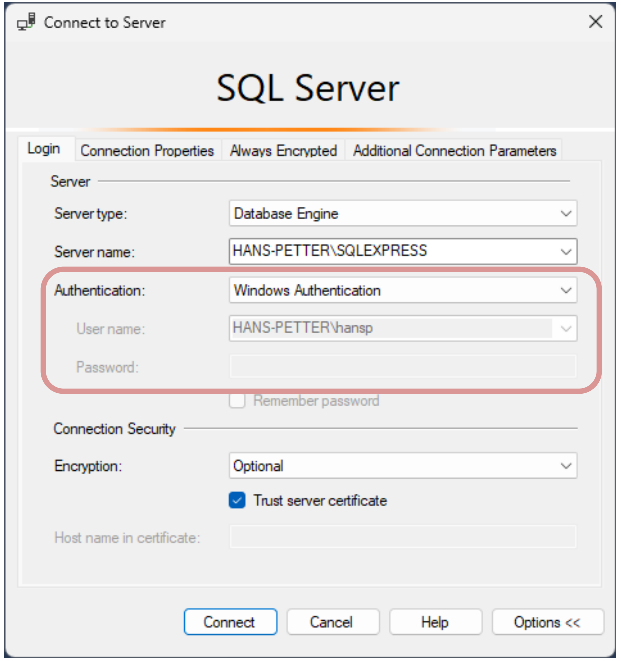
Connection Security

Encryption: Optional

☑ Trust server certificate

Host name in certificate:

Here the "sa" user is used. If you select "Mixed mode" during installation of SQL Server Express, you need to specify the Password for the "sa" user. In general, better to create additional SQL users in the SQL Server Management Studio and use that instead of "sa".



Connect to Server

SQL Server

Login | Connection Properties | Always Encrypted | Additional Connection Parameters

Server

Server type: Database Engine

Server name: HANS-PETTER\SQLEXPRESS

Authentication: Windows Authentication

User name: HANS-PETTER\hansp

Password:

☐ Remember password

Connection Security

Encryption: Optional

☑ Trust server certificate

Host name in certificate:

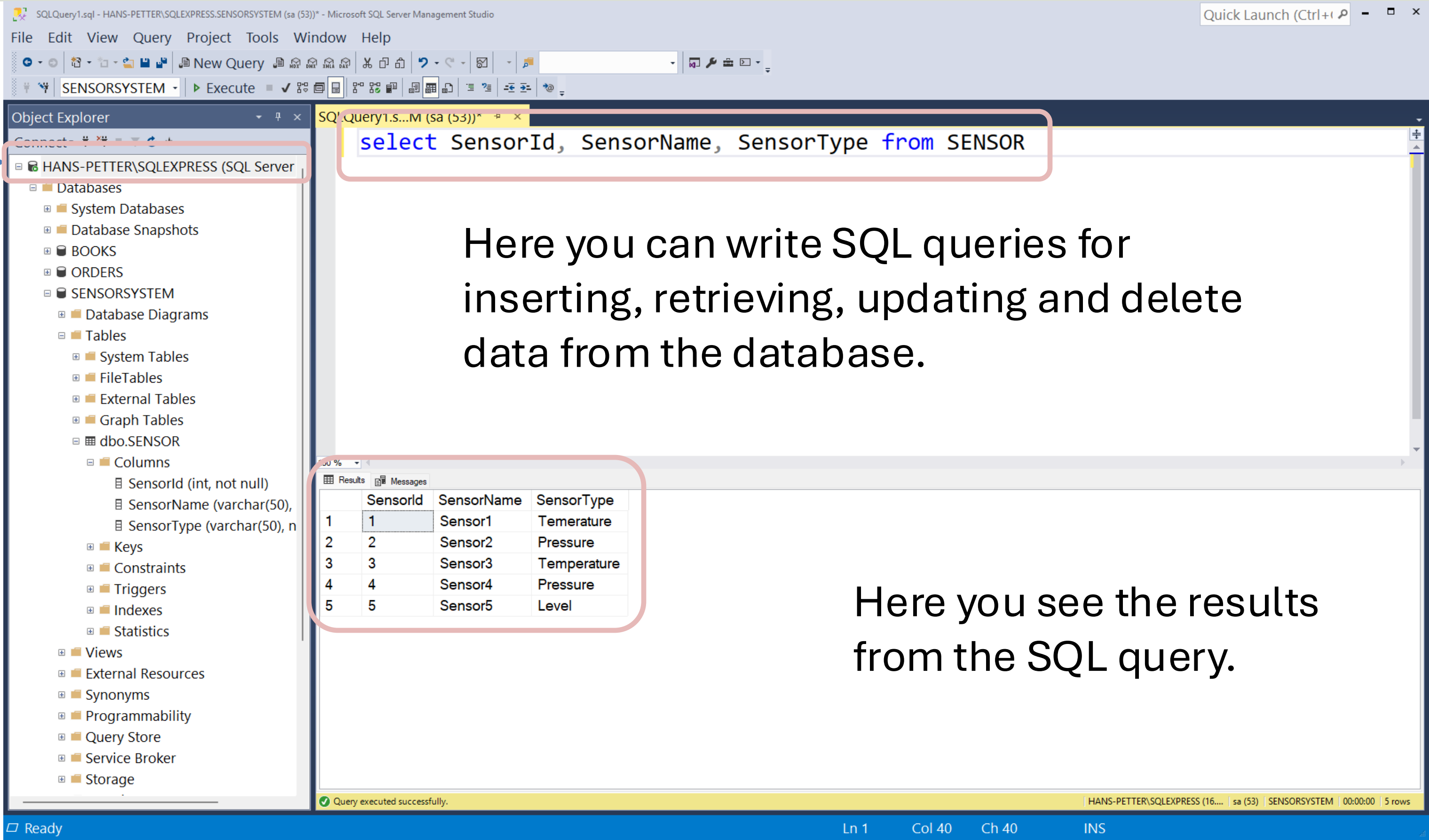
Choose between:

- **SQL Server Authentication**
- **Windows Authentication** (the current Windows account credentials are used for authentication)

SQL Server Management Studio

Server Name

Databases


SQLQuery1.sql - HANS-PETTER\SQLEXPRESS.SENSORSYSTEM (sa (53))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
SENSORSYSTEM
Execute
Object Explorer
SQLQuery1.s...M (sa (53))*
Results
Messages
Ready
Ln 1
Col 40
Ch 40
INS

- HANS-PETTER\SQLEXPRESS (SQL Server
  - Databases
    - System Databases
    - Database Snapshots
    - BOOKS
    - ORDERS
    - SENSORSYSTEM
      - Database Diagrams
      - Tables
        - System Tables
        - FileTables
        - External Tables
        - Graph Tables
        - dbo.SENSOR
          - Columns
            - SensorId (int, not null)
            - SensorName (varchar(50),
            - SensorType (varchar(50), n
          - Keys
          - Constraints
          - Triggers
          - Indexes
          - Statistics
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Query Store
      - Service Broker
      - Storage

```sql
select SensorId, SensorName, SensorType from SENSOR
```

	SensorId	SensorName	SensorType
1	1	Sensor1	Temerature
2	2	Sensor2	Pressure
3	3	Sensor3	Temperature
4	4	Sensor4	Pressure
5	5	Sensor5	Level

Query executed successfully. HANS-PETTER\SQLEXPRESS (16.... sa (53) SENSORSYSTEM 00:00:00 5 rows

Here you can write SQL queries for inserting, retrieving, updating and delete data from the database.

Here you see the results from the SQL query.

Structured Query Language (SQL)

- Structured Query Language (SQL) is used to write, read and update data from the Database System
- You can use SQL inside the “SQL Server Management Studio” or inside your Visual Studio C# App.
- SQL Example: “select * from SCHOOL”

SQL Examples

Query Examples:

- **insert** into STUDENT (Name , Number, SchoolId)
  values ('John Smith', '100005', 1)
- **select** SchoolId, Name from SCHOOL
- **select** * from SCHOOL where SchoolId > 100
- **update** STUDENT set Name='John Wayne' **where** StudentId=2
- **delete** from STUDENT **where** SchoolId=3

We have 4 different Query Types: **INSERT**, **SELECT**, **UPDATE** and **DELETE**

CRUD: **C** – Create or Insert Data, **R** – Retrieve (Select) Data, **U** – Update Data, **D** – Delete Data

Database

We will create a Database called “SENSORSYSTEM” and create the following Table:

```
CREATE TABLE SENSOR
(
   SensorId int   NOT NULL  IDENTITY (1,1),
   SensorName   varchar(50)  NOT NULL,
   SensorType   varchar(50)  NOT NULL
)
```

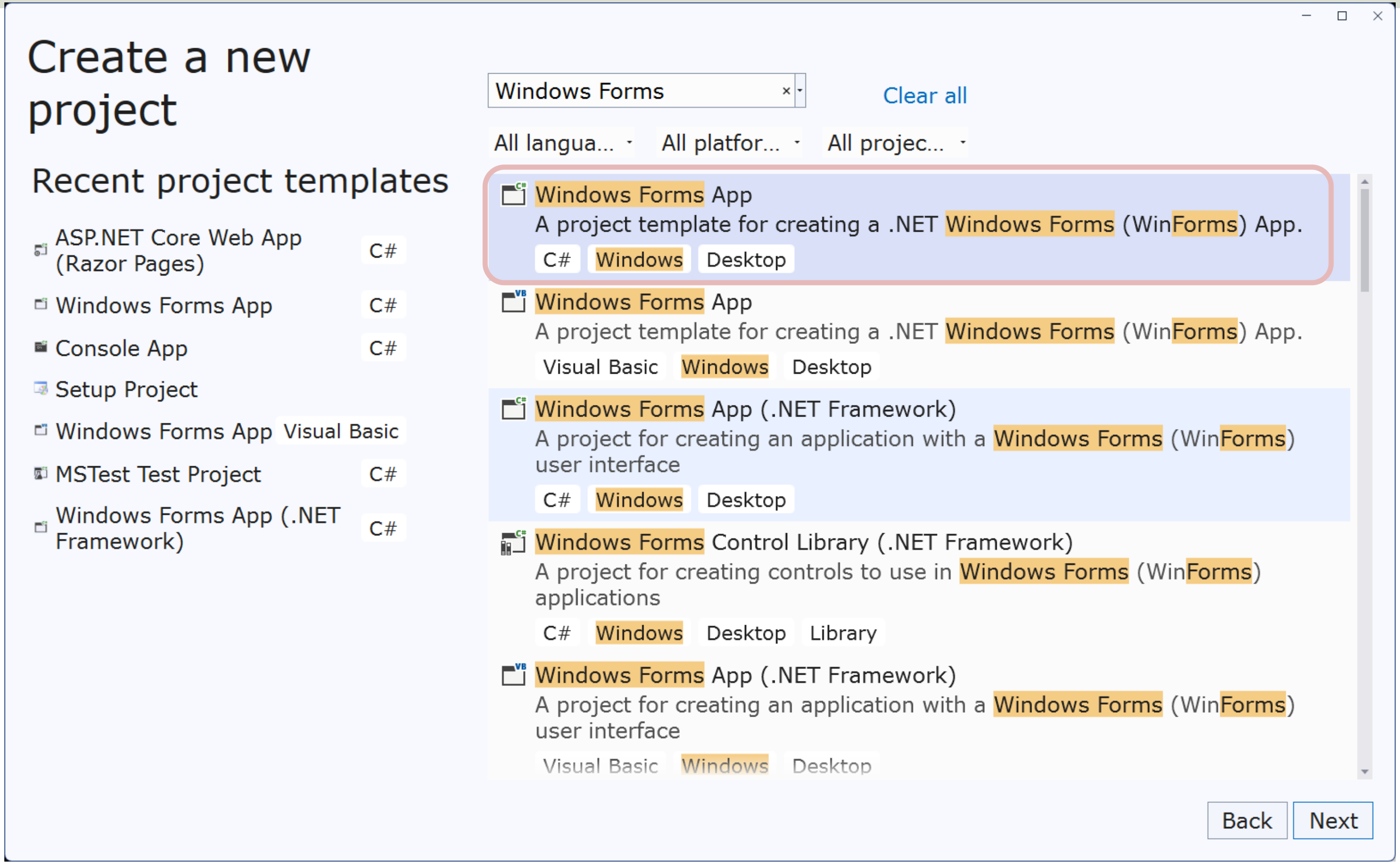
Note! This is a very simplified example, typically we create multiple tables.

SQL Server and Visual Studio

Visual Studio

Hans-Petter Halvorsen

Windows Forms Application



Create a new project

Recent project templates

ASP.NET Core Web App (Razor Pages)	C#
Windows Forms App	C#
Console App	C#
Setup Project	
Windows Forms App	Visual Basic
MSTest Test Project	C#
Windows Forms App (.NET Framework)	C#

Windows Forms

Clear all

All langua... All platfor... All projec...

Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

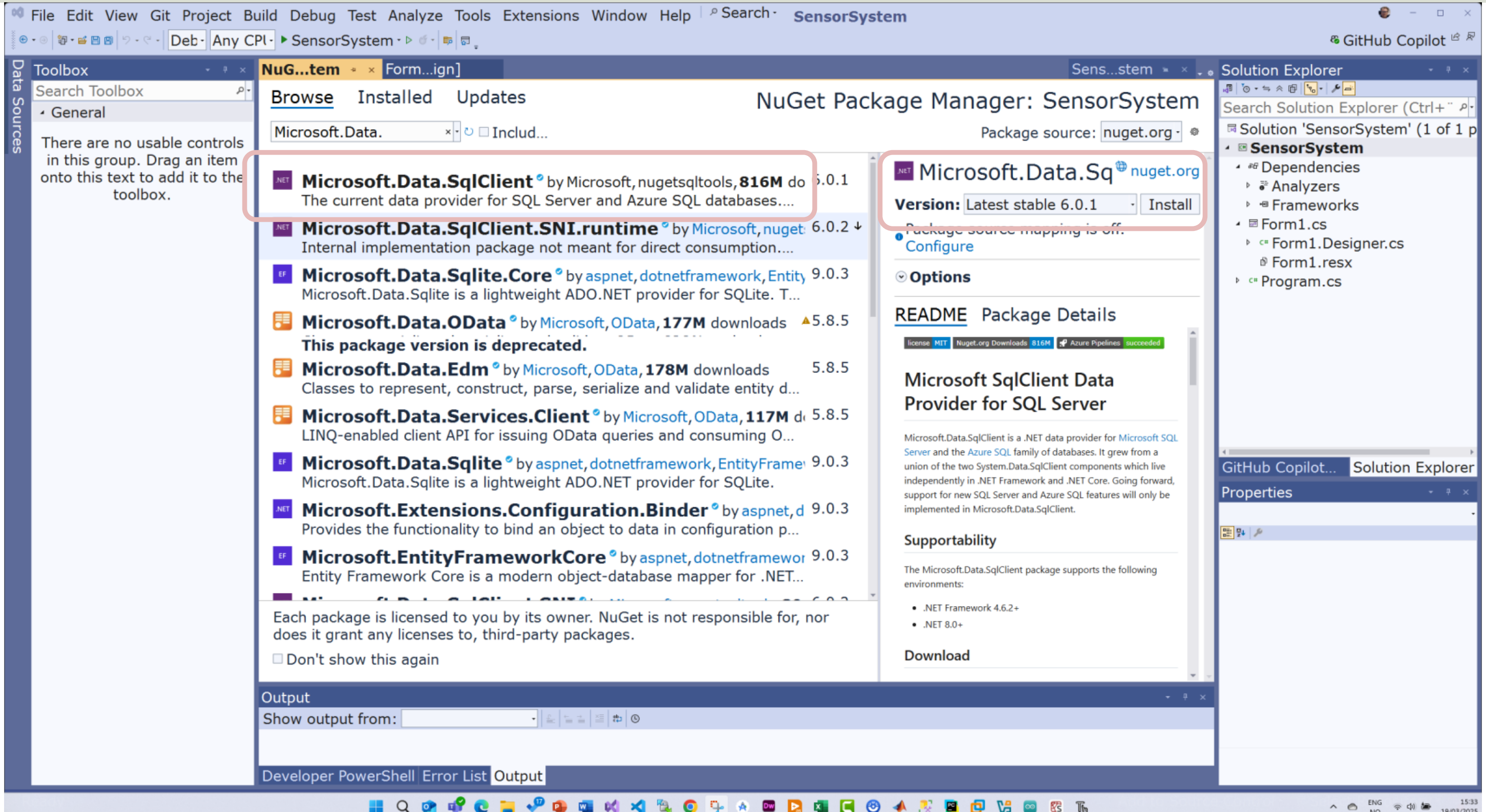
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
Visual Basic Windows Desktop

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

Windows Forms Control Library (.NET Framework)
A project for creating controls to use in Windows Forms (WinForms) applications
C# Windows Desktop Library

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
Visual Basic Windows Desktop

Microsoft.Data.SqlClient



NuGet Package Manager: SensorSystem

Browse Installed Updates

Microsoft.Data. Includ... Package source: nuget.org

Microsoft.Data.SqlClient by Microsoft, nugetsqltools, **816M** do 6.0.1
The current data provider for SQL Server and Azure SQL databases....

Microsoft.Data.SqlClient.SNI.runtime by Microsoft, nuget 6.0.2
Internal implementation package not meant for direct consumption....

Microsoft.Data.Sqlite.Core by aspnet, dotnetfoundation, Entity 9.0.3
Microsoft.Data.Sqlite is a lightweight ADO.NET provider for SQLite. T...

Microsoft.Data.OData by Microsoft, OData, **177M** downloads 5.8.5
This package version is deprecated.

Microsoft.Data.Edm by Microsoft, OData, **178M** downloads 5.8.5
Classes to represent, construct, parse, serialize and validate entity d...

Microsoft.Data.Services.Client by Microsoft, OData, **117M** d 5.8.5
LINQ-enabled client API for issuing OData queries and consuming O...

Microsoft.Data.Sqlite by aspnet, dotnetfoundation, EntityFrame 9.0.3
Microsoft.Data.Sqlite is a lightweight ADO.NET provider for SQLite.

Microsoft.Extensions.Configuration.Binder by aspnet, d 9.0.3
Provides the functionality to bind an object to data in configuration p...

Microsoft.EntityFrameworkCore by aspnet, dotnetframewor 9.0.3
Entity Framework Core is a modern object-database mapper for .NET...

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Don't show this again

Microsoft.Data.Sq nuget.org

Version: Latest stable 6.0.1 Install

Package source mapping is off. Configure

Options

README Package Details

license MIT | Nuget.org Downloads 816M | Azure Pipelines succeeded

Microsoft SqlClient Data Provider for SQL Server

Microsoft.Data.SqlClient is a .NET data provider for Microsoft SQL Server and the Azure SQL family of databases. It grew from a union of the two System.Data.SqlClient components which live independently in .NET Framework and .NET Core. Going forward, support for new SQL Server and Azure SQL features will only be implemented in Microsoft.Data.SqlClient.

Supportability

The Microsoft.Data.SqlClient package supports the following environments:

- .NET Framework 4.6.2+
- .NET 8.0+

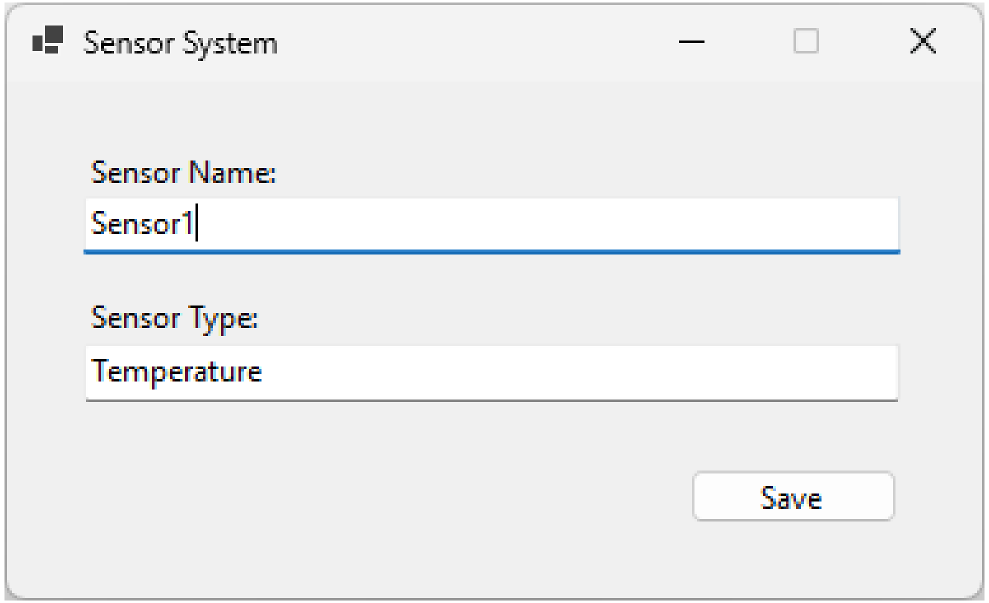
Download

SQL Server and Visual Studio

Insert Data into Database

Hans-Petter Halvorsen

“Write Sensor Data” App



Sensor Name: Sensor1

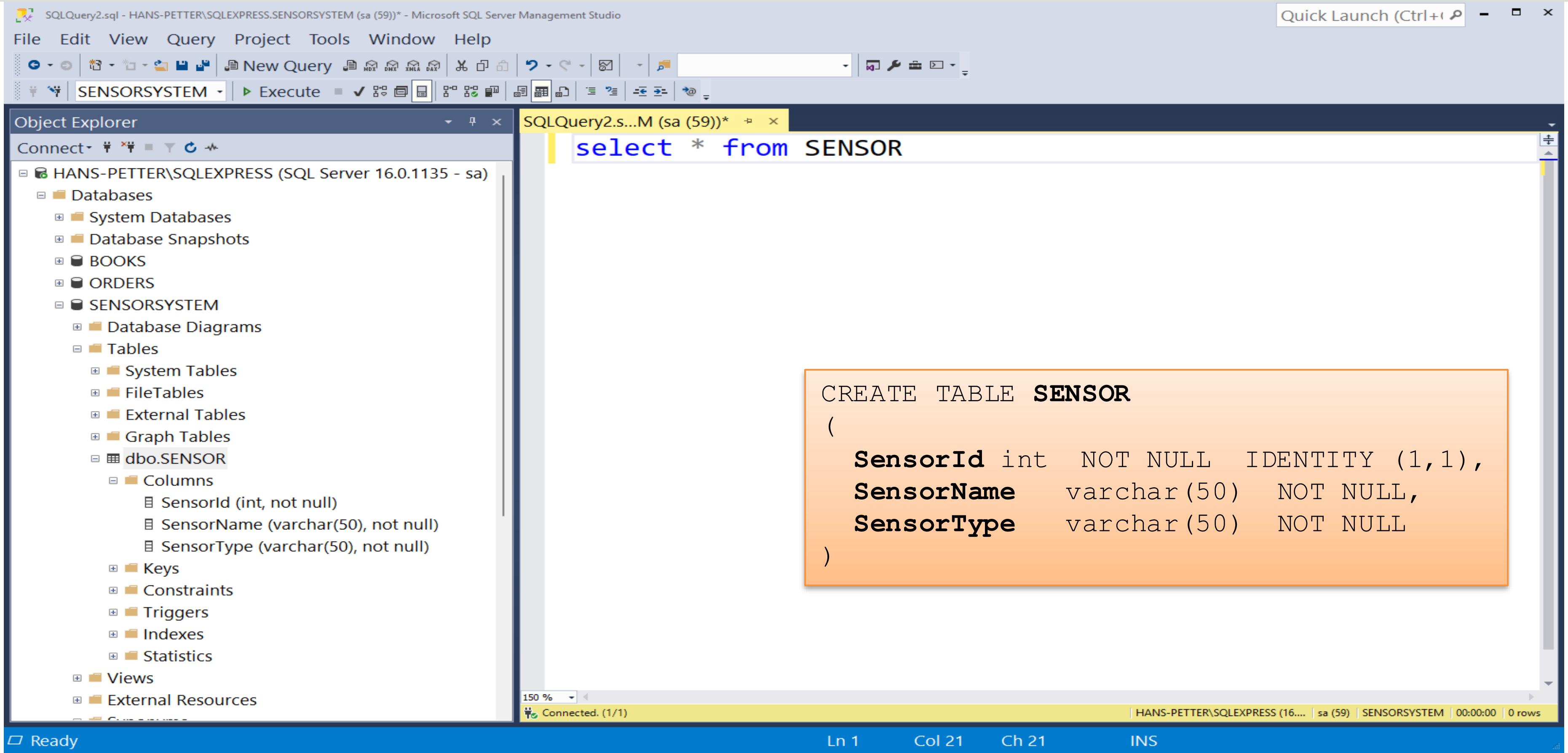
Sensor Type: Temperature

When clicking “Save”, data entered in the TextBoxes will be saved in the SQL Server Database.

The following SQL query will be executed in the C# code:

```
INSERT INTO SENSOR (SensorName, SensorType) VALUES ('Sensor1', 'Temperature')
```

Database



SQLQuery2.sql - HANS-PETTER\SQLEXPRESS.SENSORSYSTEM (sa (59))* - Microsoft SQL Server Management Studio

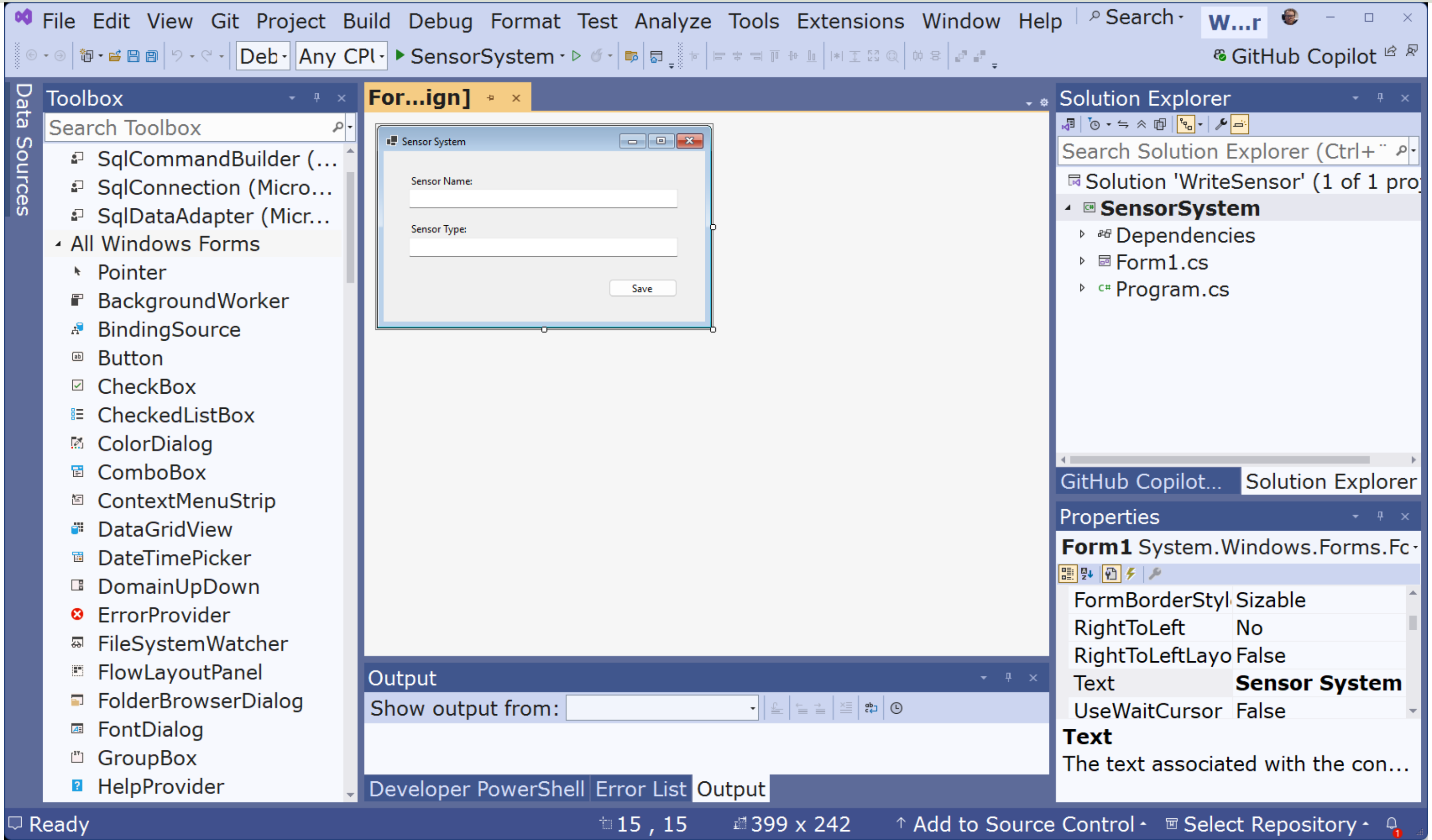
```
HANS-PETTER\SQLEXPRESS (SQL Server 16.0.1135 - sa)
  Databases
    System Databases
    Database Snapshots
    BOOKS
    ORDERS
    SENSORSYSTEM
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.SENSOR
          Columns
            SensorId (int, not null)
            SensorName (varchar(50), not null)
            SensorType (varchar(50), not null)
          Keys
          Constraints
          Triggers
          Indexes
          Statistics
      Views
      External Resources
```

```sql
select * from SENSOR
```

```sql
CREATE TABLE SENSOR
(
  SensorId int  NOT NULL  IDENTITY (1,1),
  SensorName   varchar(50)  NOT NULL,
  SensorType   varchar(50)  NOT NULL
)
```

150 %
Connected. (1/1)
HANS-PETTER\SQLEXPRESS (16.... | sa (59) | SENSORSYSTEM | 00:00:00 | 0 rows
Ready
Ln 1
Col 21
Ch 21
INS

Visual Studio



FormBorderStyl	Sizable
RightToLeft	No
RightToLeftLayo	False
Text	Sensor System
UseWaitCursor	False

Text
The text associated with the con...

Connection String

The Connection String to connect to the Database can be written in many ways. Here is some examples (There are many other ways also):

Windows Authentication (the current Windows account credentials are used for authentication):

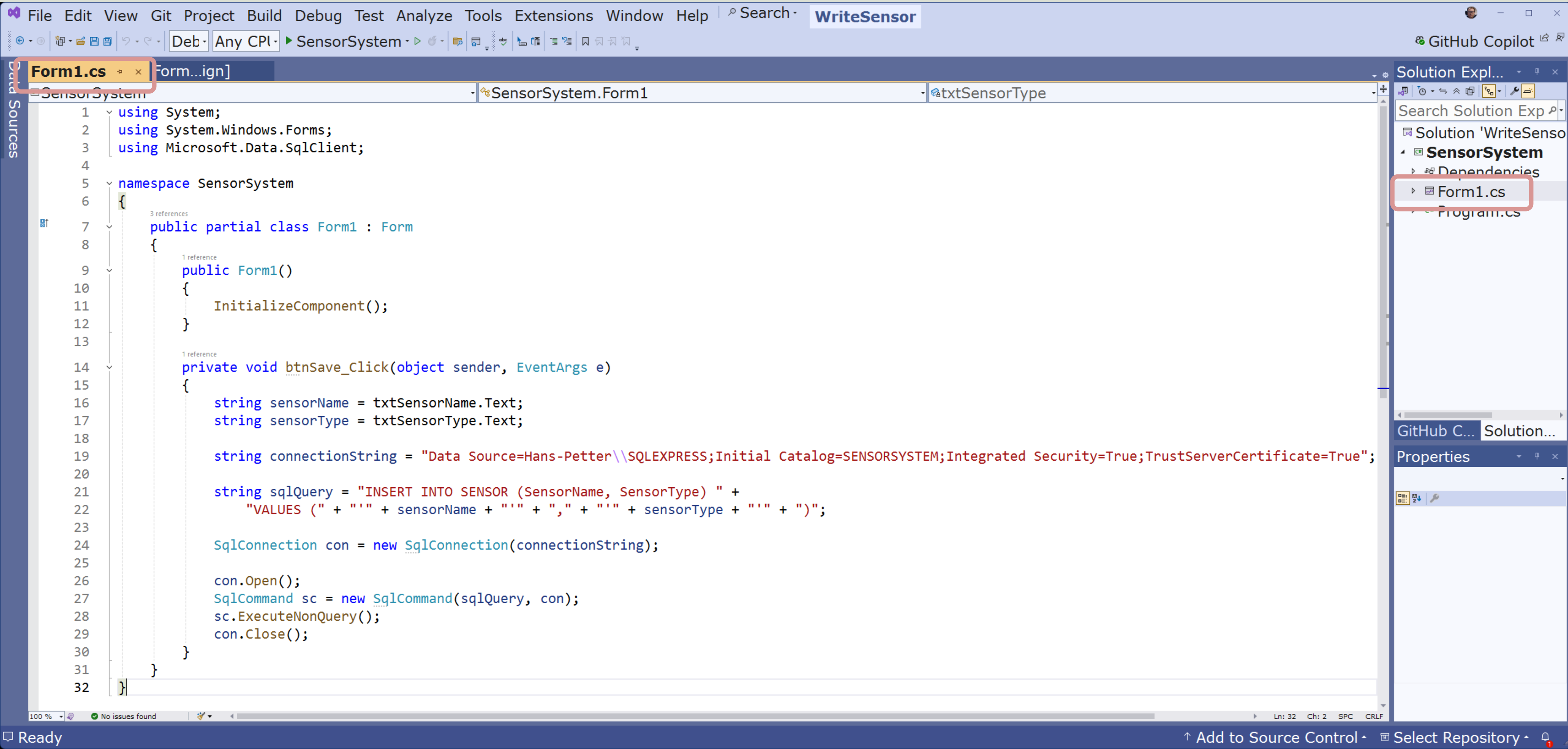
```
string connectionString = "Server=Hans-Petter\\SQLEXPRESS;
Database=SENSORSYSTEM;
Integrated Security=True;
TrustServerCertificate=True";
```

SQL Server Authentication:

```
string connectionString = "Server=Hans-Petter\\SQLEXPRESS;
Database=SENSORSYSTEM;
Uid=sa;
Pwd=YourPassword;
TrustServerCertificate=True";
```

Here the “sa” user is used. If you select “Mixed mode” during installation of SQL Server Express, you need to specify the Password for the “sa” user. In general, better to create additional SQL users in the SQL Server Management Studio and use that instead of “sa”.

C#



```
using System;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;

namespace SensorSystem
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            string sensorName = txtSensorName.Text;
            string sensorType = txtSensorType.Text;

            string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated Security=True;TrustServerCertificate=True";

            string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) " +
                "VALUES (" + "'" + sensorName + "'" + "," + "'" + sensorType + "'" + ")";

            SqlConnection con = new SqlConnection(connectionString);

            con.Open();
            SqlCommand sc = new SqlCommand(sqlQuery, con);
            sc.ExecuteNonQuery();
            con.Close();
        }
    }
}
```

Form1.cs

```
using System;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;

namespace SensorSystem
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            string sensorName = txtSensorName.Text;
            string sensorType = txtSensorType.Text;

            string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated
            Security=True;TrustServerCertificate=True";

            string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) " +
                "VALUES (" + "'" + sensorName + "'" + "," + "'" + sensorType + "'" + ")";

            SqlConnection con = new SqlConnection(connectionString);

            con.Open();
            SqlCommand sc = new SqlCommand(sqlQuery, con);
            sc.ExecuteNonQuery();
            con.Close();
        }
    }
}
```

Step 2: Create **Method**

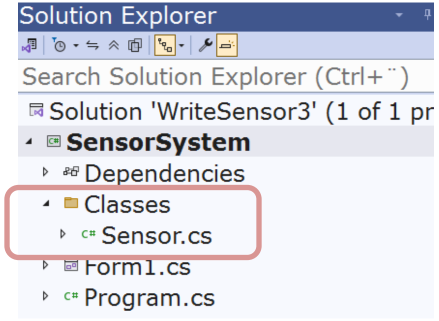
```
private void btnSave_Click(object sender, EventArgs e)
{
 SaveData();
}

private void SaveData()
{
        string sensorName = txtSensorName.Text;
        string sensorType = txtSensorType.Text;

        string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial
            Catalog=SENSORSYSTEM;Integrated Security=True;TrustServerCertificate=True";

        string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) " +
              "VALUES (" + "'" + sensorName + "'" + "," + "'" + sensorType + "'" + ")";

        SqlConnection con = new SqlConnection(connectionString);
        con.Open();
        SqlCommand sc = new SqlCommand(sqlQuery, con);
        sc.ExecuteNonQuery();
        con.Close();
}
```

Form1.cs: Here is a separate Method “**SaveData()**” is made to improve Code Quality

Step 3a: Create a **Class**

```
using Microsoft.Data.SqlClient;

namespace SensorSystem.Classes
{
  class Sensor
  {
    public void SaveSensorData(string sensorName, string sensorType)
    {
      string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated
         Security=True;TrustServerCertificate=True";

      string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) " +
        "VALUES (" + "'" + sensorName + "'" + "," + "'" + sensorType + "'" + ")";

      SqlConnection con = new SqlConnection(connectionString);
      con.Open();
      SqlCommand sc = new SqlCommand(sqlQuery, con);
      sc.ExecuteNonQuery();
      con.Close();
    }
  }
}
```

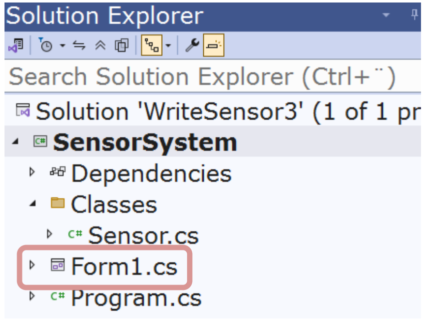
Here is a separate **Class "Sensor"** and a Method "**SaveSensorData()**" made to improve Code Quality



Step 3b: Use the Class

```
private void btnSave_Click(object sender, EventArgs e)
{
   SaveData();
}

private void SaveData()
{
   string sensorName = txtSensorName.Text;
   string sensorType = txtSensorType.Text;

   Sensor sensor = new Sensor();

   sensor.SaveSensorData(sensorName, sensorType);
}
```

Then we use the Class and Method in “Form1.cs”


Solution Explorer

Search Solution Explorer (Ctrl+¨)

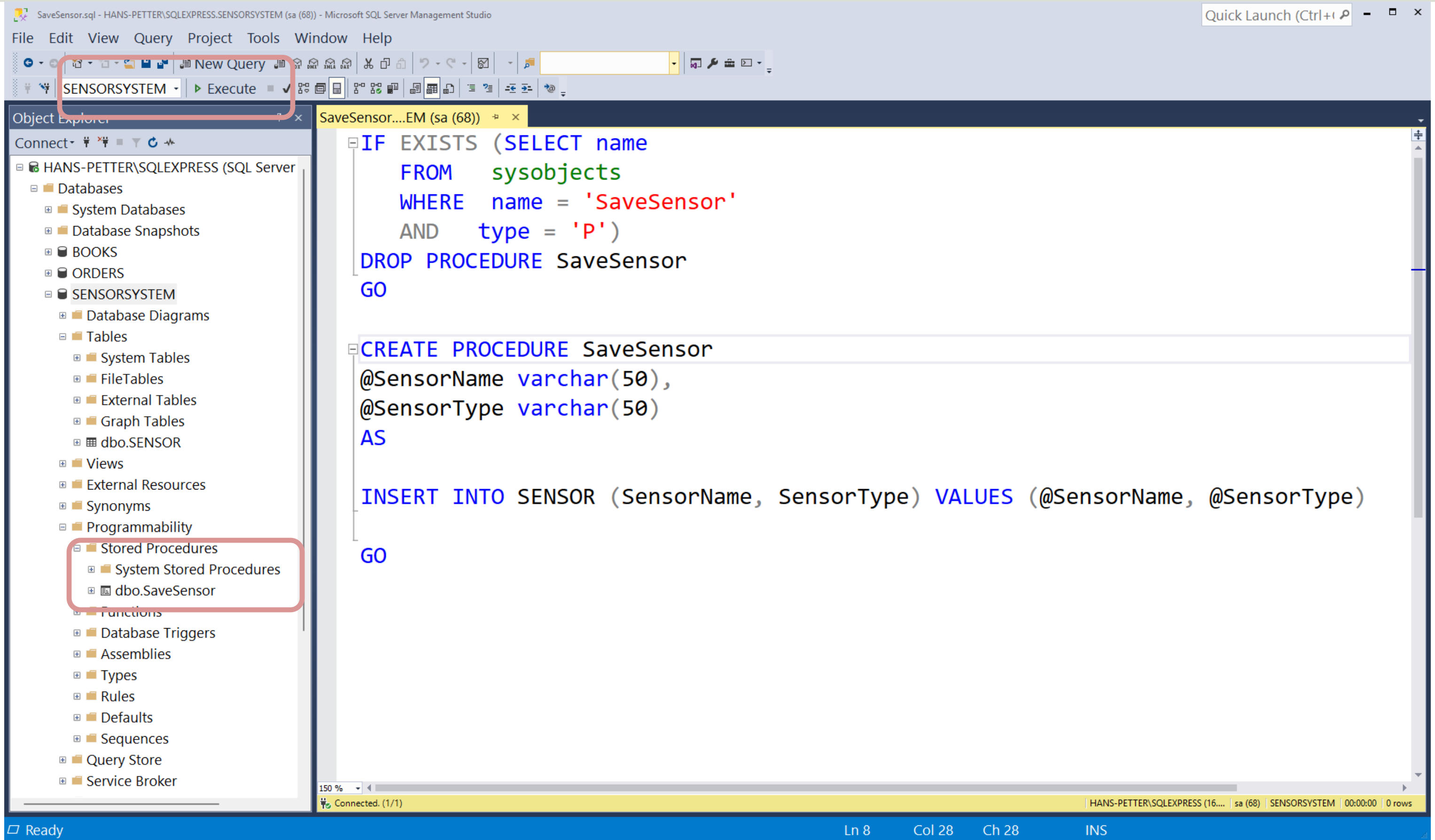
Solution 'WriteSensor3' (1 of 1 pr
- SensorSystem
  - Dependencies
  - Classes
    - Sensor.cs
  - Form1.cs
  - Program.cs

Step 4a: Create **Stored Procedure**

Create Stored Procedure “**SaveSensor**” in SQL Server Management Studio:

```sql
IF EXISTS (SELECT name
   FROM   sysobjects
   WHERE  name = 'SaveSensor'
   AND   type = 'P')
DROP PROCEDURE SaveSensor
GO

CREATE PROCEDURE SaveSensor
@SensorName varchar(50),
@SensorType varchar(50)
AS

INSERT INTO SENSOR (SensorName, SensorType) VALUES (@SensorName, @SensorType)
GO
```

Step 4a: Create **Stored Procedure**



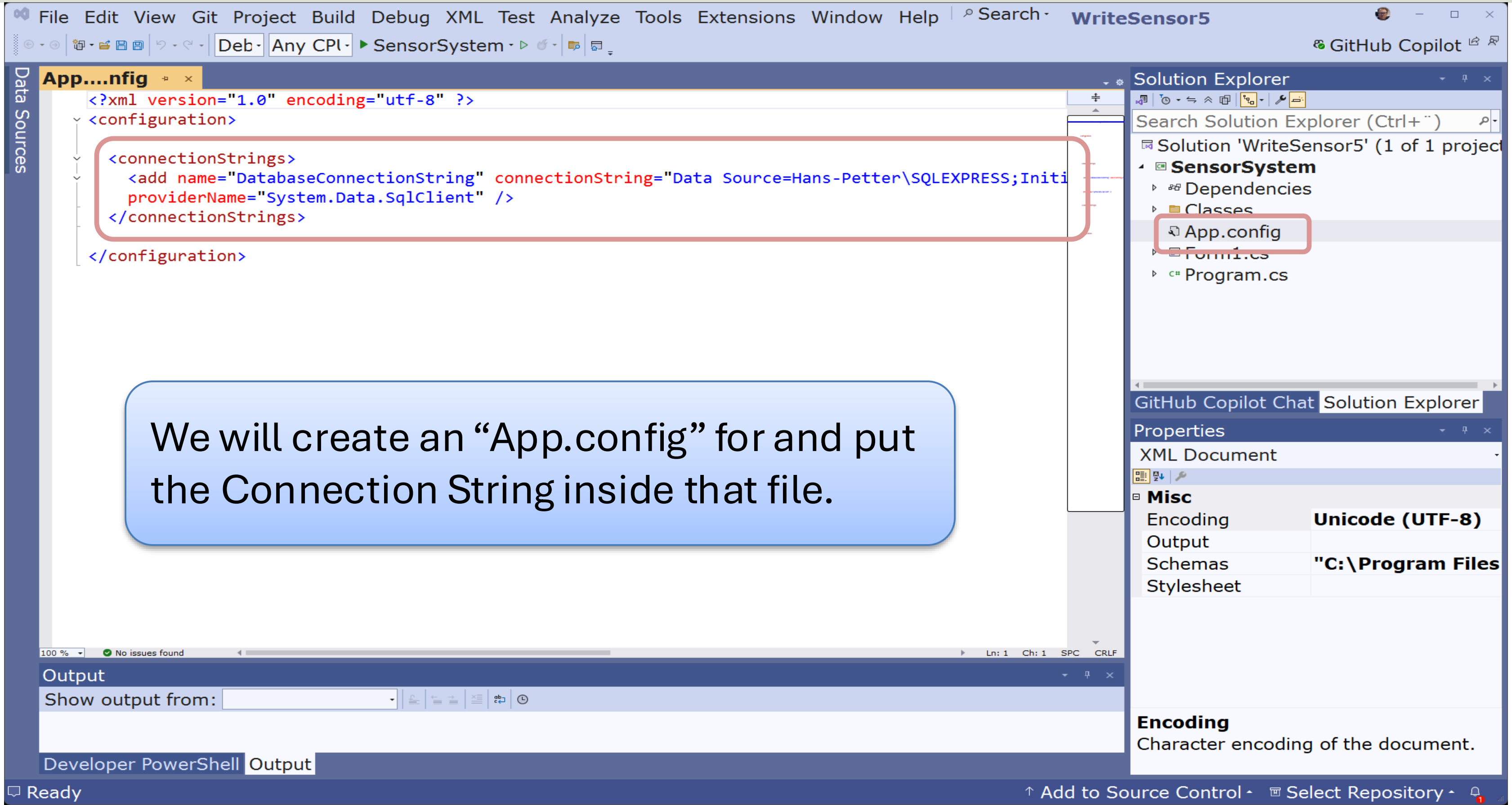
```sql
IF EXISTS (SELECT name
    FROM   sysobjects
    WHERE  name = 'SaveSensor'
    AND   type = 'P')
DROP PROCEDURE SaveSensor
GO

CREATE PROCEDURE SaveSensor
@SensorName varchar(50),
@SensorType varchar(50)
AS

INSERT INTO SENSOR (SensorName, SensorType) VALUES (@SensorName, @SensorType)

GO
```

Step 4b: Use Stored Procedure

Sensor.cs

```
using System.Data;
using Microsoft.Data.SqlClient;

namespace SensorSystem.Classes
{
    class Sensor
    {
        public void SaveSensorData(string sensorName, string sensorType)
        {
            string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial
                Catalog=SENSORSYSTEM;Integrated Security=True;TrustServerCertificate=True";

            SqlConnection con = new SqlConnection(connectionString);
            con.Open();

            SqlCommand cmd = new SqlCommand("SaveSensor", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add(new SqlParameter("@SensorName", sensorName));
            cmd.Parameters.Add(new SqlParameter("@SensorType", sensorType));

            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
}
```

Step 5a: Create App.config



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <connectionStrings>
    <add name="DatabaseConnectionString" connectionString="Data Source=Hans-Petter\SQLEXPRESS;Initi
    providerName="System.Data.SqlClient" />
  </connectionStrings>

</configuration>
```

We will create an "App.config" for and put the Connection String inside that file.

App.config

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <connectionStrings>
    <add name="DatabaseConnectionString" connectionString="Server=Hans-Petter\SQLEXPRESS;
    Database=SENSORSYSTEM;Integrated Security=True;TrustServerCertificate=True"
    providerName="System.Data.SqlClient" />
  </connectionStrings>

</configuration>
```

Step 5b: Use App.config

Sensor.cs

```
using System.Data;
using Microsoft.Data.SqlClient;
using System.Configuration;

namespace SensorSystem.Classes
{
   class Sensor
   {
      public void SaveSensorData(string sensorName, string sensorType)
      {
         string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

         SqlConnection con = new SqlConnection(connectionString);
         con.Open();

         SqlCommand cmd = new SqlCommand("SaveSensor", con);
         cmd.CommandType = CommandType.StoredProcedure;

         cmd.Parameters.Add(new SqlParameter("@SensorName", sensorName));
         cmd.Parameters.Add(new SqlParameter("@SensorType", sensorType));

         cmd.ExecuteNonQuery();
         con.Close();
      }
   }
}
```

SQL Server and Visual Studio

Get Data from Database

Hans-Petter Halvorsen

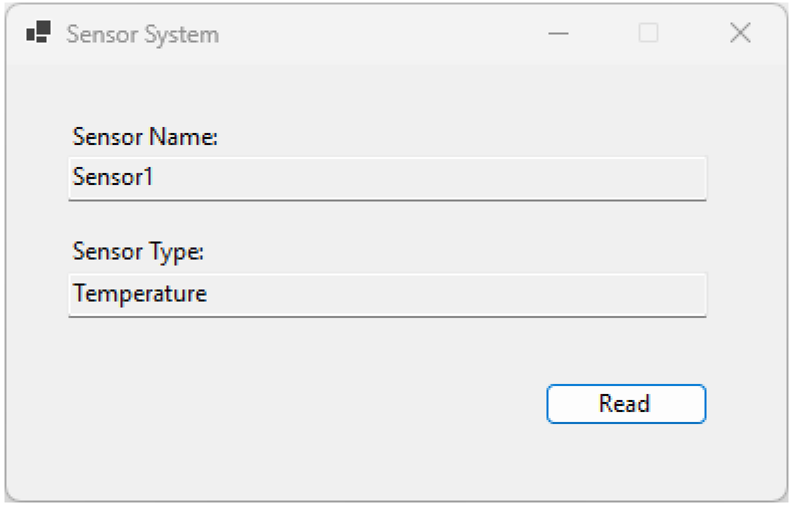
Table of Contents

Get Data from Database

We will create different Applications:

a) Get Data into TextBoxes
b) Get Data into into a ListBox and a ComboBox
c) Get Data into into a DataGridView

“Read Sensor Data” App



Sensor Name:
Sensor1

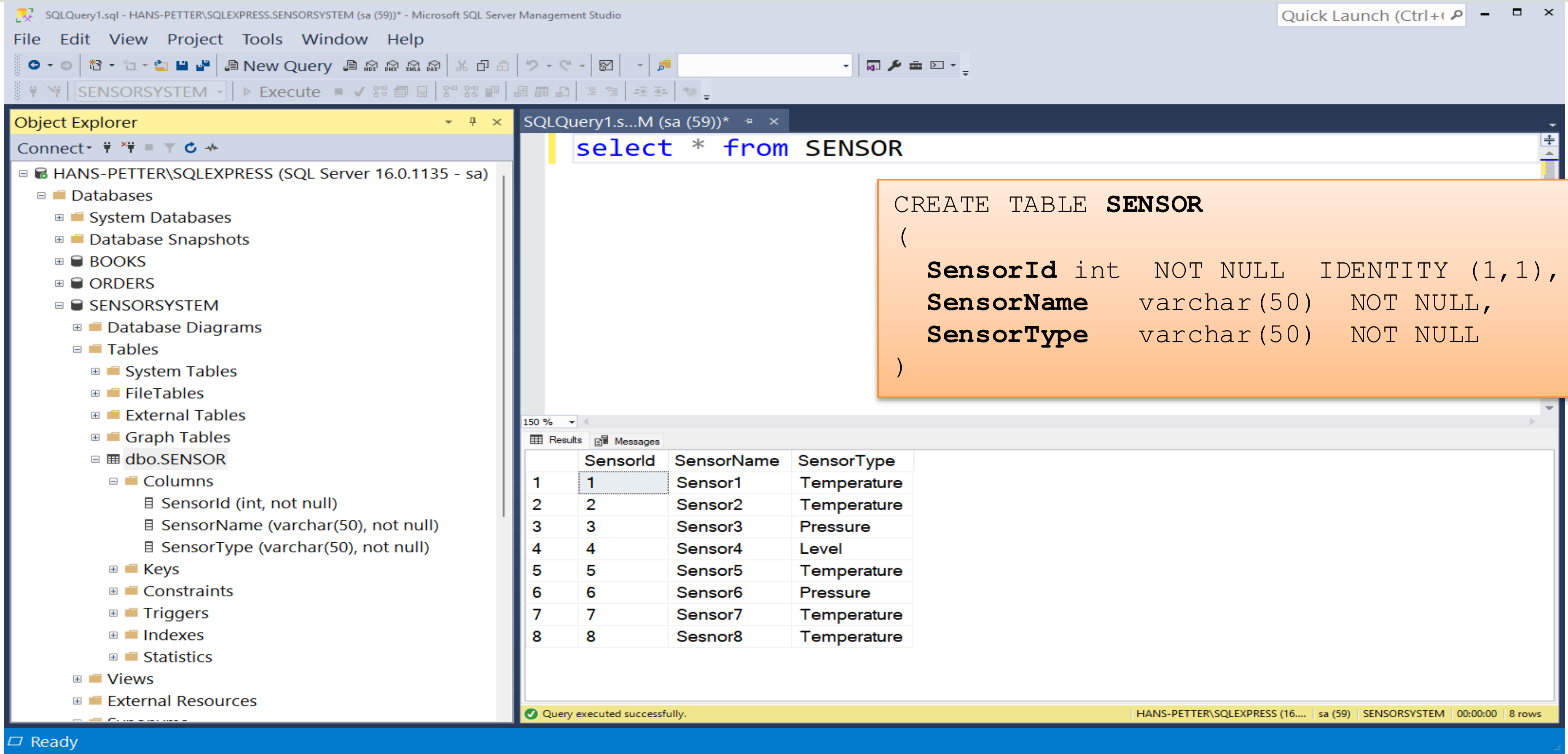
Sensor Type:
Temperature

When clicking “Read”, data from the SQL Server Database will be shown in the TextBoxes.

The following SQL query will be executed in the C# code:

```
SELECT SensorName, SensorType FROM SENSOR WHERE SensorId = 1
```

Database



SQLQuery1.sql - HANS-PETTER\SQLEXPRESS.SENSORSYSTEM (sa (59))* - Microsoft SQL Server Management Studio

Object Explorer

- HANS-PETTER\SQLEXPRESS (SQL Server 16.0.1135 - sa)
  - Databases
    - System Databases
    - Database Snapshots
    - BOOKS
    - ORDERS
    - SENSORSYSTEM
      - Database Diagrams
      - Tables
        - System Tables
        - FileTables
        - External Tables
        - Graph Tables
        - dbo.SENSOR
          - Columns
            - SensorId (int, not null)
            - SensorName (varchar(50), not null)
            - SensorType (varchar(50), not null)
          - Keys
          - Constraints
          - Triggers
          - Indexes
          - Statistics
      - Views
      - External Resources

```sql
select * from SENSOR
```

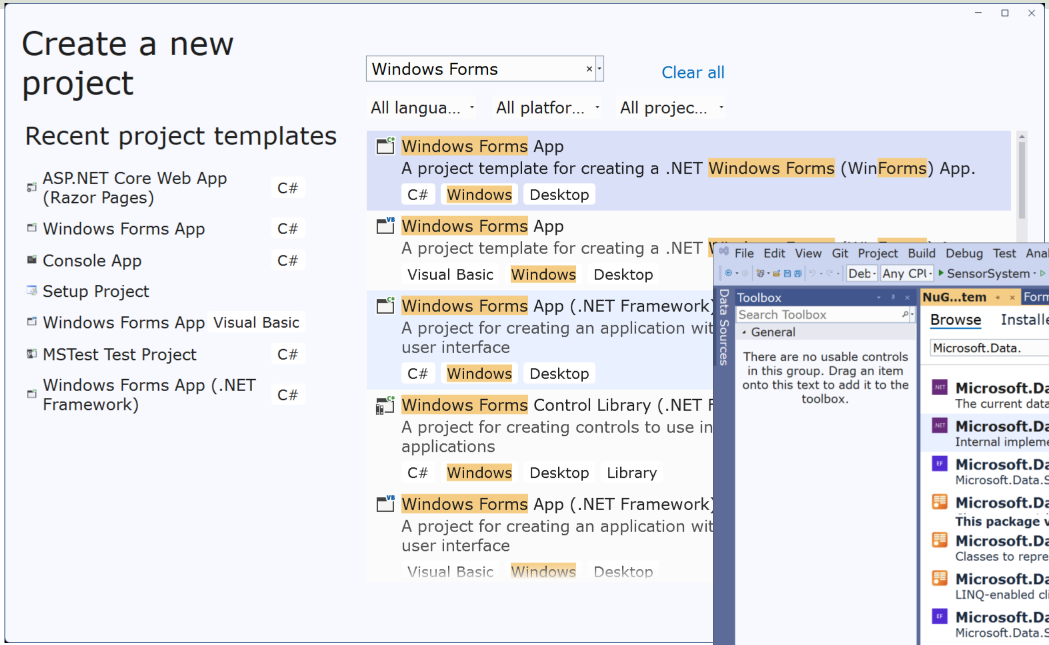
```sql
CREATE TABLE SENSOR
(
  SensorId int  NOT NULL  IDENTITY (1,1),
  SensorName   varchar(50)  NOT NULL,
  SensorType   varchar(50)  NOT NULL
)
```

	SensorId	SensorName	SensorType
1	1	Sensor1	Temperature
2	2	Sensor2	Temperature
3	3	Sensor3	Pressure
4	4	Sensor4	Level
5	5	Sensor5	Temperature
6	6	Sensor6	Pressure
7	7	Sensor7	Temperature
8	8	Sesnor8	Temperature

Query executed successfully. | HANS-PETTER\SQLEXPRESS (16.... | sa (59) | SENSORSYSTEM | 00:00:00 | 8 rows

Ready

Windows Forms Application



Create a new project

Recent project templates

- ASP.NET Core Web App (Razor Pages) C#
- Windows Forms App C#
- Console App C#
- Setup Project
- Windows Forms App Visual Basic
- MSTest Test Project C#
- Windows Forms App (.NET Framework) C#

Windows Forms

Clear all

All langua... All platfor... All projec...

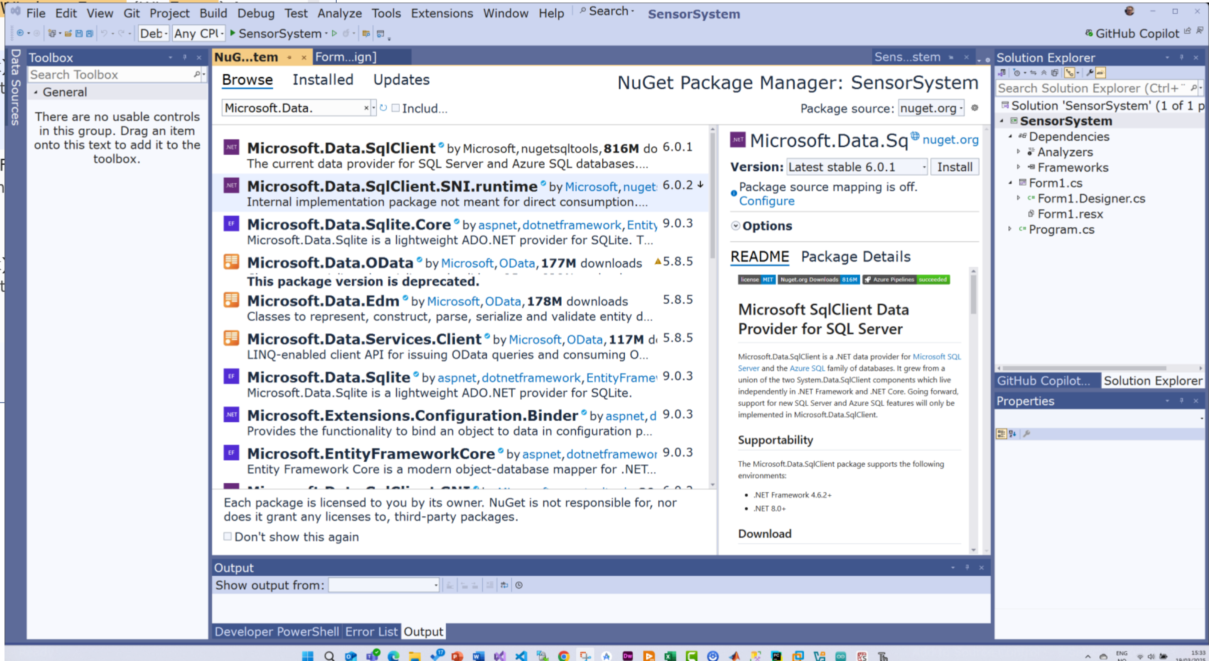
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

Windows Forms App
A project template for creating a .NET
Visual Basic Windows Desktop

Windows Forms App (.NET Framework)
A project for creating an application wit
user interface
C# Windows Desktop

Windows Forms Control Library (.NET F
A project for creating controls to use in
applications
C# Windows Desktop Library

Windows Forms App (.NET Framework)
A project for creating an application wit
user interface
Visual Basic Windows Desktop



File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search SensorSystem

Toolbox — General: There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

NuGet Package Manager: SensorSystem — Browse Installed Updates

Search: Microsoft.Data. Package source: nuget.org

- **Microsoft.Data.SqlClient** by Microsoft, nugetsqltools, 816M do 6.0.1 — The current data provider for SQL Server and Azure SQL databases....
- **Microsoft.Data.SqlClient.SNI.runtime** by Microsoft, nuget 6.0.2 — Internal implementation package not meant for direct consumption....
- **Microsoft.Data.Sqlite.Core** by aspnet, dotnetframework, Entity 9.0.3 — Microsoft.Data.Sqlite is a lightweight ADO.NET provider for SQLite. T...
- **Microsoft.Data.OData** by Microsoft, OData, 177M downloads 5.8.5 — This package version is deprecated.
- **Microsoft.Data.Edm** by Microsoft, OData, 178M downloads 5.8.5 — Classes to represent, construct, parse, and validate entity d...
- **Microsoft.Data.Services.Client** by Microsoft, OData, 117M d 5.8.5 — LINQ-enabled client API for issuing OData queries and consuming O...
- **Microsoft.Data.Sqlite** by aspnet, dotnetframework, EntityFrame 9.0.3 — Microsoft.Data.Sqlite is a lightweight ADO.NET provider for SQLite.
- **Microsoft.Extensions.Configuration.Binder** by aspnet, d 9.0.3 — Provides the functionality to bind an object to data in configuration p...
- **Microsoft.EntityFrameworkCore** by aspnet, dotnetframewor 9.0.3 — Entity Framework Core is a modern object-database mapper for .NET...

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Don't show this again

Microsoft.Data.Sq nuget.org
Version: Latest stable 6.0.1 Install
Package source mapping is off. Configure
Options
README Package Details

license MIT | Nuget.org Downloads 816M | Azure Pipelines succeeded

Microsoft SqlClient Data Provider for SQL Server

Microsoft.Data.SqlClient is a .NET data provider for Microsoft SQL Server and the Azure SQL family of databases. It grew from a union of the two System.Data.SqlClient components which live independently in .NET Framework and .NET Core. Going forward, support for new SQL Server and Azure SQL features will only be implemented in Microsoft.Data.SqlClient.

Supportability

The Microsoft.Data.SqlClient package supports the following environments:

- .NET Framework 4.6.2+
- .NET 8.0+

Download

Solution Explorer
Solution 'SensorSystem' (1 of 1 p
- SensorSystem
  - Dependencies
    - Analyzers
    - Frameworks
  - Form1.cs
    - Form1.Designer.cs
    - Form1.resx
  - Program.cs

Output — Show output from:

Developer PowerShell Error List Output

Connection String

The Connection String to connect to the Database can be written in many ways. Here is some examples (There are many other ways also):

Windows Authentication (the current Windows account credentials are used for authentication):

```
string connectionString = "Server=Hans-Petter\\SQLEXPRESS;
Database=SENSORSYSTEM;
Integrated Security=True;
TrustServerCertificate=True";
```

SQL Server Authentication:

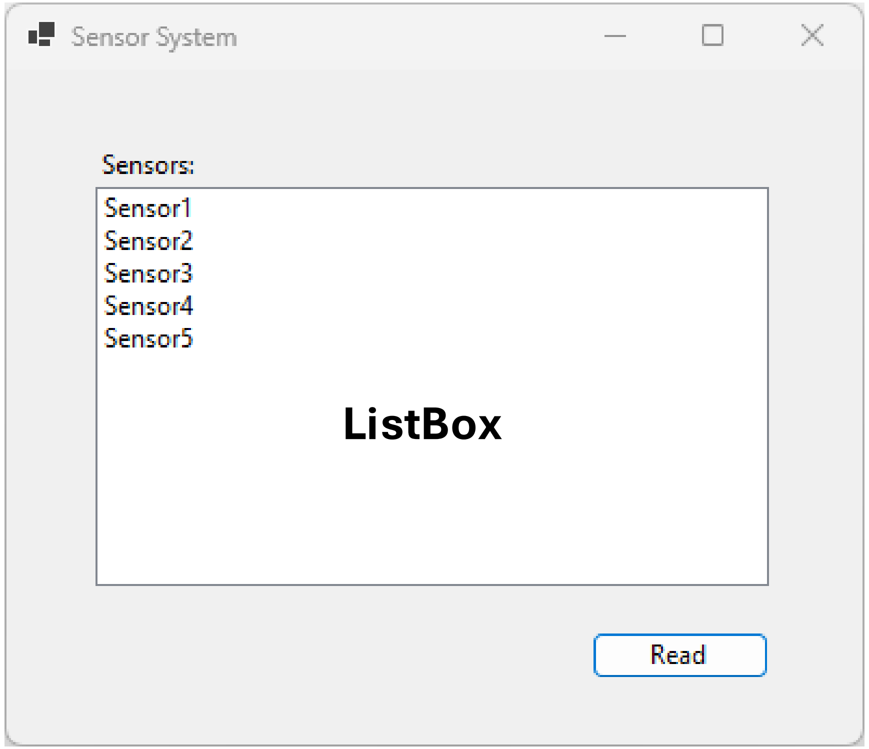
```
string connectionString = "Server=Hans-Petter\\SQLEXPRESS;
Database=SENSORSYSTEM;
Uid=sa;
Pwd=YourPassword;
TrustServerCertificate=True";
```

Here the "sa" user is used. If you select "Mixed mode" during installation of SQL Server Express, you need to specify the Password for the "sa" user. In general, better to create additional SQL users in the SQL Server Management Studio and use that instead of "sa".

Code

```
using System;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;

namespace SensorSystem
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void btnRead_Click(object sender, EventArgs e)
      {
         string connectionString = ”Server=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated
             Security=True;TrustServerCertificate=True";

         string sqlQuery = "select SensorName, SensorType from SENSOR WHERE SensorId=1";

         SqlConnection con = new SqlConnection(connectionString);
         con.Open();

         SqlCommand cmd = new SqlCommand(sqlQuery, con);
         SqlDataReader dr = cmd.ExecuteReader();

         if (dr.Read()) {
            txtSensorName.Text = dr["SensorName"].ToString();
            txtSensorType.Text = dr["SensorType"].ToString();
         }
         con.Close();
      }
   }
}
```

“Read Sensors” App



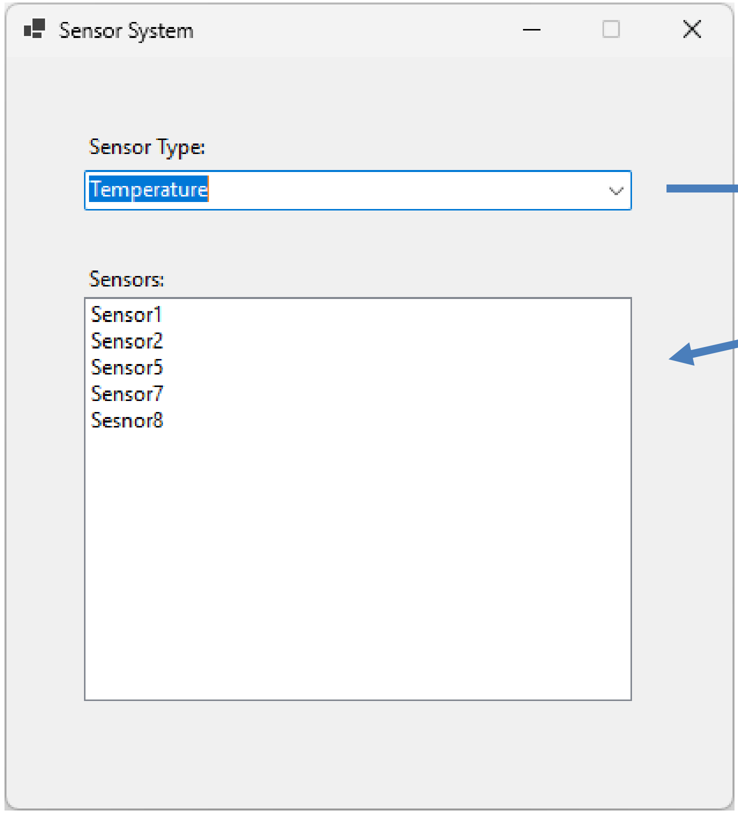
Here we use a **ListBox**

When clicking “Read”, data from the SQL Server Database will be shown in the **ListBox**.

Code

```
using System;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;

namespace SensorSystem
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnRead_Click(object sender, EventArgs e)
        {
            string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated Security=True;TrustServerCertificate=True";

            string sqlQuery = "select SensorName from SENSOR";
            SqlConnection con = new SqlConnection(connectionString);
            con.Open();

            SqlCommand cmd = new SqlCommand(sqlQuery, con);
            SqlDataReader dr = cmd.ExecuteReader();

            lstSensors.Items.Clear();
            if (dr != null)
            {
                while (dr.Read())
                {
                    string sensor = dr["SensorName"].ToString();
                    lstSensors.Items.Add(sensor);
                }
            }
            con.Close();
        }
    }
}
```

"Read Sensors" App 2


Sensor System

Sensor Type:

Temperature

Sensors:

Sensor1
Sensor2
Sensor5
Sensor7
Sesnor8

Here we use a **ComboBox** and a **ListBox**

- The "SensorType" ComboBox are filled with Data from the SQL Server Database.
- When Selecting a specific "SensorType", we get available "Sensors" from the SQL Server Database

Form1.cs

```
using System;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;

namespace SensorSystem
{
   public partial class Form1 : Form
   {
      string connectionString = “server=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated
         Security=True;TrustServerCertificate=True";

      public Form1()
      {
         InitializeComponent();
         GetSensorTypes();
      }

      private void cboSensorTypes_SelectedIndexChanged(object sender, EventArgs e)
      {
         string sensorTypeSelected = cboSensorTypes.SelectedItem.ToString();
         GetSensors(sensorTypeSelected);
      }

      … (see next page)
```

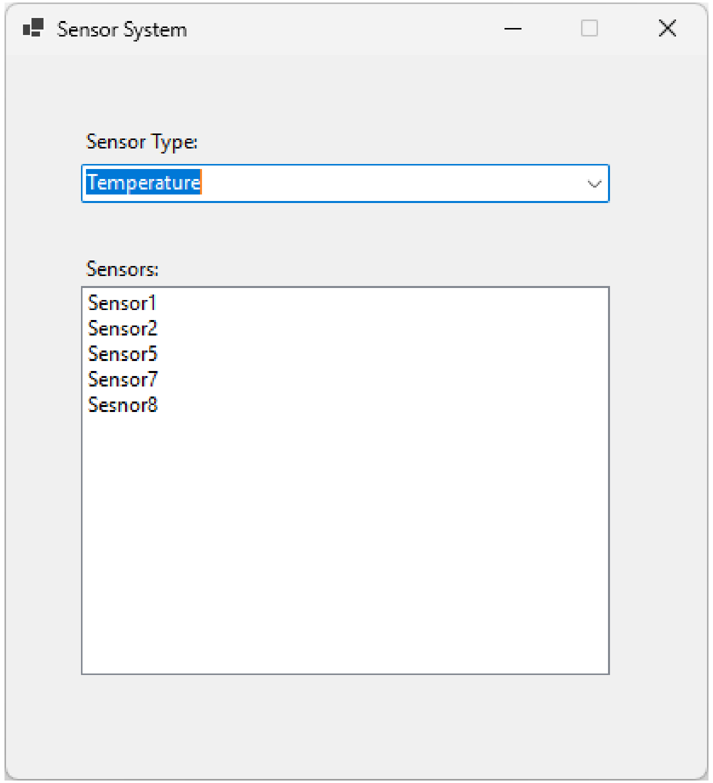
GetSensorTypes()

```
private void GetSensorTypes()
{
    string sqlQuery = "select distinct SensorType from SENSOR order by SensorType";

    SqlConnection con = new SqlConnection(connectionString);
    con.Open();

    SqlCommand cmd = new SqlCommand(sqlQuery, con);
    SqlDataReader dr = cmd.ExecuteReader();

    cboSensorTypes.Items.Clear();
    if (dr != null)
    {
        while (dr.Read())
        {
            string sensorType = dr["SensorType"].ToString();
            cboSensorTypes.Items.Add(sensorType);
        }
    }
    con.Close();
}
```

GetSensors()

```
private void GetSensors(string sensorTypeSelected)
{
    string sqlQuery = "select SensorName from SENSOR where SensorType = '" + sensorTypeSelected +
     "' order by SensorName";

    SqlConnection con = new SqlConnection(connectionString);
    con.Open();

    SqlCommand cmd = new SqlCommand(sqlQuery, con);
    SqlDataReader dr = cmd.ExecuteReader();

    lstSensors.Items.Clear();
    if (dr != null)
    {
      while (dr.Read())
      {
          string sensor = dr["SensorName"].ToString();
          lstSensors.Items.Add(sensor);
      }
    }
   con.Close();
}
```

“Read Sensors” App 2b - Class



Sensor Type:

Temperature

Sensors:

Sensor1
Sensor2
Sensor5
Sensor7
Sesnor8

We will improve code structure by creating a separate **Class** called “Sensor” and move most of the code into that class.

Sensor.cs

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

namespace SensorSystem.Classes
{
   public class Sensor
   {
      string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated Security=True;TrustServerCertificate=True";

      public List<string> GetSensorTypes()
      {
         string sqlQuery = "select distinct SensorType from SENSOR order by SensorType";

         SqlConnection con = new SqlConnection(connectionString);
         con.Open();

         SqlCommand cmd = new SqlCommand(sqlQuery, con);
         SqlDataReader dr = cmd.ExecuteReader();

         List<string> itemsSensorTypes = new List<string>();

         if (dr != null)
         {
            while (dr.Read())
            {
               string sensorType = dr["SensorType"].ToString();
               itemsSensorTypes.Add(sensorType);
            }
         }
         con.Close();
         return itemsSensorTypes;
      }

      public List<string> GetSensors(string sensorTypeSelected)
      {
         string sqlQuery = "select SensorName from SENSOR where SensorType = '" + sensorTypeSelected + "' order by SensorName";

         SqlConnection con = new SqlConnection(connectionString);
         con.Open();

         SqlCommand cmd = new SqlCommand(sqlQuery, con);
         SqlDataReader dr = cmd.ExecuteReader();

         List<string> itemsSensors = new List<string>();

         if (dr != null)
         {
            while (dr.Read())
            {
               string sensor = dr["SensorName"].ToString();
               itemsSensors.Add(sensor);
            }
         }
         con.Close();
         return itemsSensors;
      }
   }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using SensorSystem.Classes;

namespace SensorSystem
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            GetSensorTypes();
        }

        private void cboSensorTypes_SelectedIndexChanged(object sender, EventArgs e)
        {
            string sensorTypeSelected = cboSensorTypes.SelectedItem.ToString();
            GetSensors(sensorTypeSelected);
        }

        private void GetSensorTypes()
        {
            //See Next Pages
        }

        private void GetSensors(string sensorTypeSelected)
        {
            //See Next Pages
        }
    }
}
```

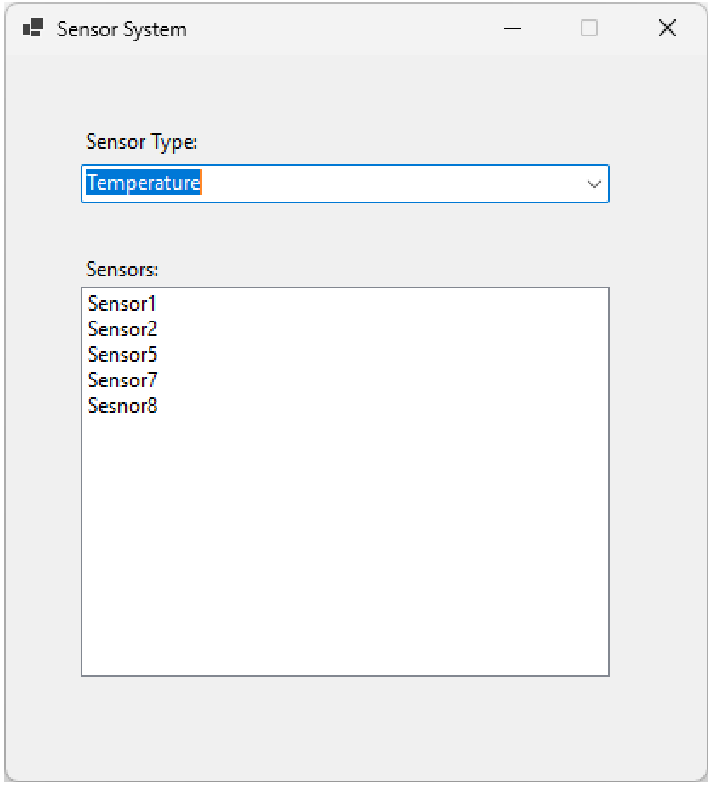
GetSensorTypes()

```
private void GetSensorTypes()
{
        cboSensorTypes.Items.Clear();

        Sensor sensor = new Sensor();

        List<string> itemsSensorTypes = new List<string>();
        itemsSensorTypes = sensor.GetSensorTypes();

        foreach (string sensorType in itemsSensorTypes)
        {
          cboSensorTypes.Items.Add(sensorType);
        }
}
```

GetSensors()

```
private void GetSensors(string sensorTypeSelected)
{
        lstSensors.Items.Clear();

        Sensor sensor = new Sensor();

        List<string> itemsSensors = new List<string>();
        itemsSensors = sensor.GetSensors(sensorTypeSelected);

        foreach (string sensorName in itemsSensors)
        {
          lstSensors.Items.Add(sensorName);
        }
}
```

“Read Sensors” App 2c – App.config



We will improve the App by putting the Connection String into a Configuration File called “**App.config**”.

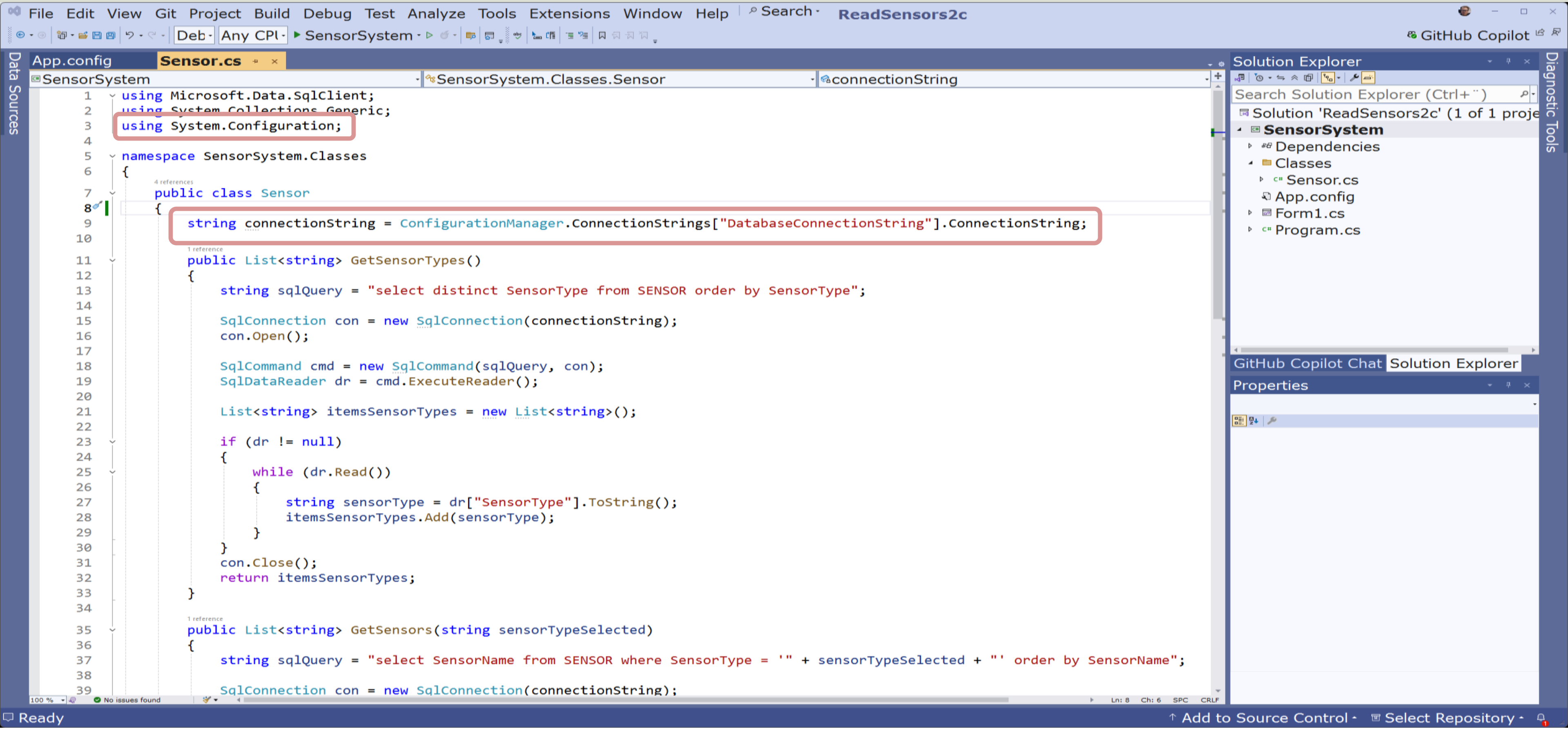
That’s because it makes it easier to change the Connection string without changing the C# code.

Then we can make an Executable Application and distribute the Application to others that do not have Visual Studio.

App.config

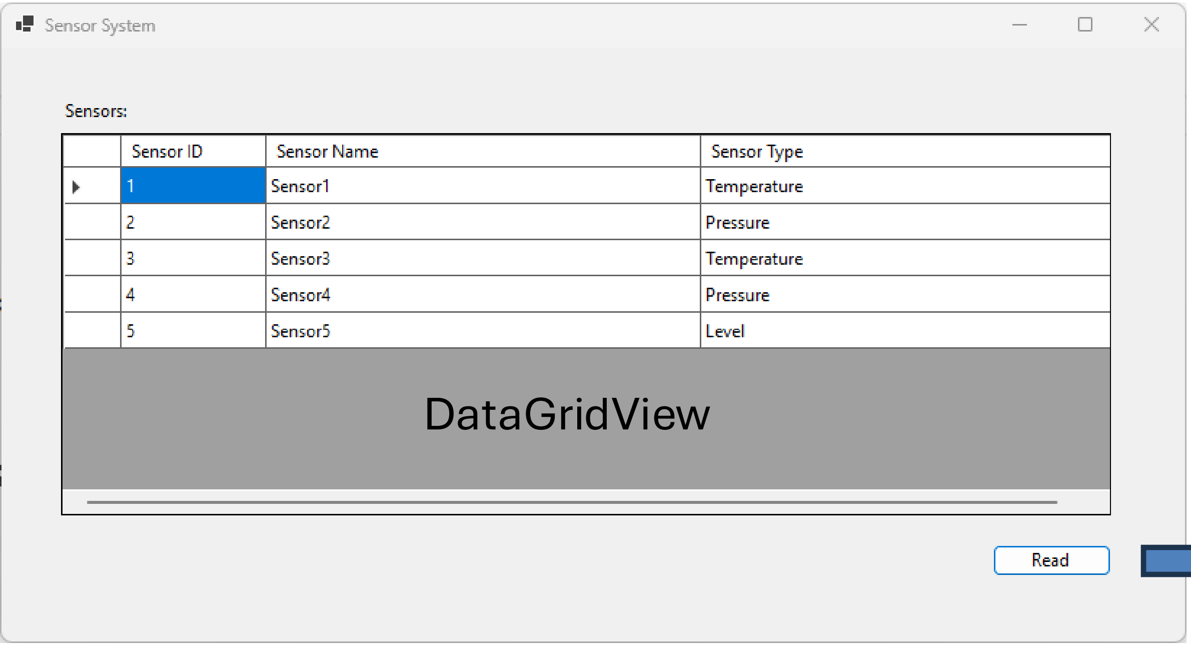
```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <connectionStrings>
    <add name="DatabaseConnectionString" connectionString="Server=Hans-Petter\SQLEXPRESS;Database=SENSORSYSTEM;Integrated
Security=True;TrustServerCertificate=True"
    providerName="System.Data.SqlClient" />
  </connectionStrings>

</configuration>
```

Updated “Sensor.cs”



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

namespace SensorSystem.Classes
{
    4 references
    public class Sensor
    {
        string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

        1 reference
        public List<string> GetSensorTypes()
        {
            string sqlQuery = "select distinct SensorType from SENSOR order by SensorType";

            SqlConnection con = new SqlConnection(connectionString);
            con.Open();

            SqlCommand cmd = new SqlCommand(sqlQuery, con);
            SqlDataReader dr = cmd.ExecuteReader();

            List<string> itemsSensorTypes = new List<string>();

            if (dr != null)
            {
                while (dr.Read())
                {
                    string sensorType = dr["SensorType"].ToString();
                    itemsSensorTypes.Add(sensorType);
                }
            }
            con.Close();
            return itemsSensorTypes;
        }

        1 reference
        public List<string> GetSensors(string sensorTypeSelected)
        {
            string sqlQuery = "select SensorName from SENSOR where SensorType = '" + sensorTypeSelected + "' order by SensorName";

            SqlConnection con = new SqlConnection(connectionString);
```

“Read Sensors” App3 - DataGridView

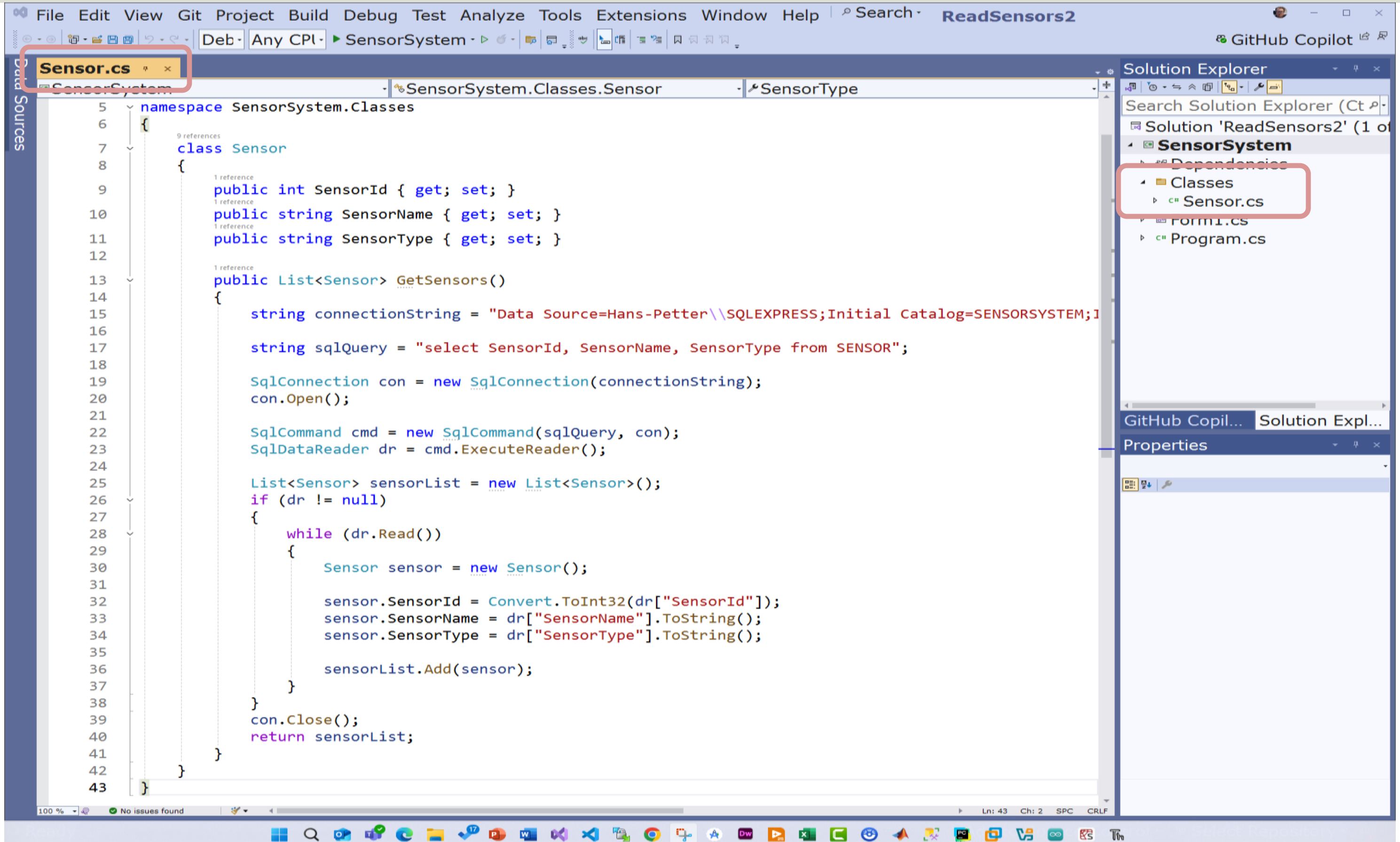


Sensor ID	Sensor Name	Sensor Type
1	Sensor1	Temperature
2	Sensor2	Pressure
3	Sensor3	Temperature
4	Sensor4	Pressure
5	Sensor5	Level

Here we will use a **DataGridView**

When clicking “Read”, data from the SQL Server Database will be shown in the **DataGridView**.

Create Class and Method



```
namespace SensorSystem.Classes
{
    class Sensor
    {
        public int SensorId { get; set; }
        public string SensorName { get; set; }
        public string SensorType { get; set; }

        public List<Sensor> GetSensors()
        {
            string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;I

            string sqlQuery = "select SensorId, SensorName, SensorType from SENSOR";

            SqlConnection con = new SqlConnection(connectionString);
            con.Open();

            SqlCommand cmd = new SqlCommand(sqlQuery, con);
            SqlDataReader dr = cmd.ExecuteReader();

            List<Sensor> sensorList = new List<Sensor>();
            if (dr != null)
            {
                while (dr.Read())
                {
                    Sensor sensor = new Sensor();

                    sensor.SensorId = Convert.ToInt32(dr["SensorId"]);
                    sensor.SensorName = dr["SensorName"].ToString();
                    sensor.SensorType = dr["SensorType"].ToString();

                    sensorList.Add(sensor);
                }
            }
            con.Close();
            return sensorList;
        }
    }
}
```

Class “Sensor.cs”

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

namespace SensorSystem.Classes
{
   class Sensor
   {
      public int SensorId { get; set; }
      public string SensorName { get; set; }
      public string SensorType { get; set; }

      public List<Sensor> GetSensors()
      {
         string connectionString = "Data Source=Hans-Petter\\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Integrated
            Security=True;TrustServerCertificate=True";

         string sqlQuery = "select SensorId, SensorName, SensorType from SENSOR";

         SqlConnection con = new SqlConnection(connectionString);
         con.Open();

         SqlCommand cmd = new SqlCommand(sqlQuery, con);
         SqlDataReader dr = cmd.ExecuteReader();

         List<Sensor> sensorList = new List<Sensor>();
         if (dr != null)
         {
            while (dr.Read())
            {
               Sensor sensor = new Sensor();

               sensor.SensorId = Convert.ToInt32(dr["SensorId"]);
               sensor.SensorName = dr["SensorName"].ToString();
               sensor.SensorType = dr["SensorType"].ToString();

               sensorList.Add(sensor);
            }
         }
         con.Close();
         return sensorList;
      }
   }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using SensorSystem.Classes;

namespace SensorSystem
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void btnRead_Click(object sender, EventArgs e)
    {
      GetData();
    }

    private void GetData()
    {
      Sensor sensor = new Sensor();
      List<Sensor> sensorList = new List<Sensor>();

      sensorList = sensor.GetSensors();

      dgwSensors.DataSource = sensorList;
      FormatDataGridView();
    }

    private void FormatDataGridView()
    {
      dgwSensors.Columns[0].HeaderText = "Sensor ID";
      dgwSensors.Columns[1].HeaderText = "Sensor Name";
      dgwSensors.Columns[2].HeaderText = "Sensor Type";

      dgwSensors.Columns[0].Width = 100;
      dgwSensors.Columns[1].Width = 300;
      dgwSensors.Columns[2].Width = 300;
    }
  }
}
```

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog