

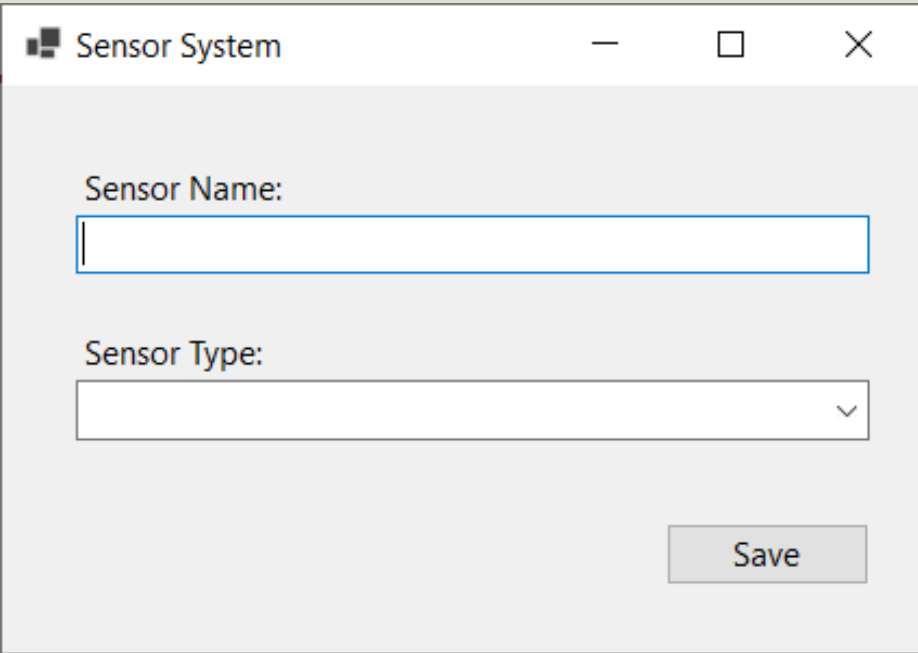
<https://www.halvorsen.blog>



# SQL Server with C# Windows Forms App

Hans-Petter Halvorsen

# Windows Forms App



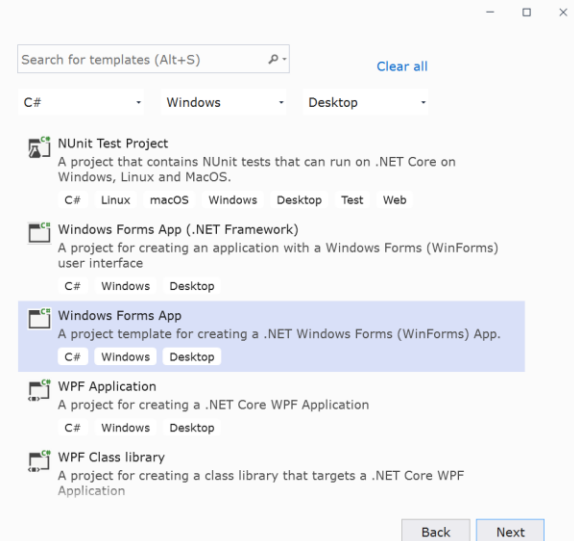
The screenshot shows a window titled "Sensor System" with a standard Windows title bar (minimize, maximize, close buttons). The window contains two input fields: "Sensor Name:" with a text box below it, and "Sensor Type:" with a dropdown menu below it. A "Save" button is located at the bottom right of the window.

We will create a basic Windows Forms App that saves data to an SQL Server Database. The App will also retrieve Data from the SQL Server Database.

## Create a new project

### Recent project templates

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



The screenshot shows the "Create a new project" dialog in Visual Studio. The "Search for templates (Alt+S)" field is empty. The "Clear all" button is visible. The "C#" language is selected, and the "Windows" and "Desktop" categories are chosen. The "Windows Forms App (.NET Framework)" template is selected and highlighted in blue. The description for this template is: "A project for creating an application with a Windows Forms (WinForms) user interface". The "C#" language and "Windows" and "Desktop" categories are also selected for this template. The "Back" and "Next" buttons are visible at the bottom right.

# Contents

- SQL Server
- ADO.NET
- C# WinForms Examples
- Structured Query Language (SQL)
- Saving Data to SQL Server
- Retrieving Data from SQL Server

# Audience

- This Tutorial is made for rookies making their first basic C# Windows Forms Database Application
- You don't need any experience in either Visual Studio or C#
- No skills in Automation or Control System is necessary

# C# Examples

## Note!

- The examples provided can be considered as a “proof of concept”
- The sample code is very simplified for clarity and doesn't necessarily represent best practices.

<https://www.halvorsen.blog>



# SQL Server

Hans-Petter Halvorsen

# What is a Database?

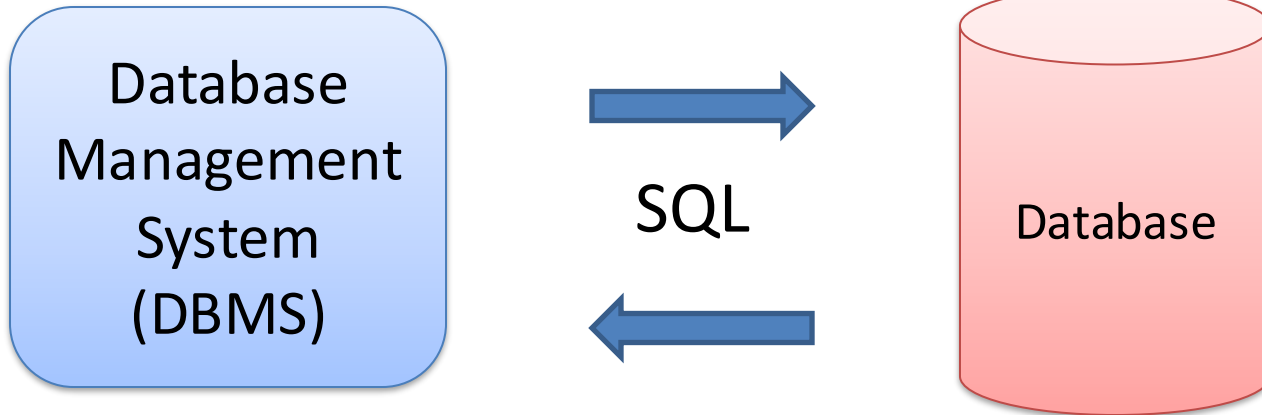
- A Database is a structured way to store lots of information.
- The information inside the database is stored in different tables.
- - “Everything” today is stored in databases!

## Examples:

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

# Database Systems

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



SQL – Structured Query Language



# Database Systems

- Oracle
- MySQL
- MariaDB
- Sybase
- Microsoft Access
- Microsoft SQL Server
- ... (we have hundreds different DBMS)

# SQL Server

- SQL Server Express
  - Free version of SQL Server that has all we need for the exercises in this Tutorial
- SQL Server Express consist of 2 parts (separate installation packages):
  - SQL Server Express
  - SQL Server Management Studio (SSMS) – This software can be used to create Databases, create Tables, Insert/Retrieve or Modify Data, etc.
- SQL Server Express Installation:  
<https://youtu.be/hhhggAlUYo8>

# SQL Server Management Studio

**1** Your SQL Server

**2** Your Database

**3** New Query

**4** Write your Query here

**5** The result from your Query

Query executed successfully. | PC88235\DEVELOPMENT (10.50 ... | sa (52) | SCHOOL | 00:00:00 | 4 rows

SchoolId	SchoolName	Description	Address	Phone	PostCode	PostAddress
1	TUC	The best school	Telemark	NULL	NULL	NULL
2	MIT	OK School	USA	NULL	NULL	NULL
3	NTNU	The second best school	Trondheim	NULL	NULL	NULL
4	University of Oslo	The third best school	Oslo	NULL	NULL	NULL

Properties window: Current connection parameters, Aggregate Status, Connection, Connection Details.

# Structured Query Language

- 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 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

<https://www.halvorsen.blog>



# ADO.NET

Hans-Petter Halvorsen

# 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

# Installation in Visual Studio

- 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



Search Toolbox

General

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

Browse Installed Updates

Microsoft.Data  Include prerelease

Package source: nuget.org

- Microsoft.Data.Edm** by Microsoft Corporation, **87.6M** downloads 5.8.4  
Classes to represent, construct, parse, serialize and validate entity data...
- Microsoft.Data.OData** by Microsoft Corporation, **87.6M** downloads 5.8.4  
Classes to serialize, deserialize and validate OData JSON payloads.
- Microsoft.Data.Services.Client** by Microsoft Corporation, **65.2M** 5.8.4  
LINQ-enabled client API for issuing OData queries and consuming OData...
- Microsoft.Data.SqlClient** by Microsoft, **65.6M** downloads 3.0.0  
Provides the data provider for SQL Server. These classes provide access t...
- Microsoft.Data.Sqlite.Core** by Microsoft, **43.8** 6.0.0-preview.4.21253.1  
Prerelease Microsoft.Data.Sqlite is a lightweight ADO.NET provid...
- Microsoft.Extensions.Configuration.Binder** 6.0.0-preview.4.21253.7  
Prerelease Functionality to bind an object to data in configuratio...
- Microsoft.Data.Sqlite** by Microsoft, **22.3M** dow 6.0.0-preview.4.21253.1  
Prerelease Microsoft.Data.Sqlite is a lightweight ADO.NET provid...
- Microsoft.AspNet.WebApi.Client** by Microsoft, **197M** downloads 5.2.7  
This package adds support for formatting and content negotiation to Syst...
- Microsoft.EntityFrameworkCore** by Microso 6.0.0-preview.4.21253.1  
Prerelease Entity Framework Core is a modern object-database...
- Microsoft.AspNetCore.Mvc.DataAnnotations** by Microsoft, **1** 2.2.0  
ASP.NET Core MVC metadata and validation system using System.Compo...

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

Do not show this again

**Microsoft.Data.SqlClient** nuget.org

Version: Latest stable 3.0.0

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:

- Microsoft.Data.SqlClient.SqlConnection
- Microsoft.Data.SqlClient.SqlException
- Microsoft.Data.SqlClient.SqlParameter
- Microsoft.Data.SqlClient.SqlDataReader
- Microsoft.Data.SqlClient.SqlCommand
- Microsoft.Data.SqlClient.SqlTransaction
- Microsoft.Data.SqlClient.SqlParameterCollecti on
- Microsoft.Data.SqlClient.SqlClientFactory

When using NuGet 3.x this package requires at least version 3.4.

Version: 3.0.0

Author(s): Microsoft

License: MIT

Date published: Wednesday, June 9, 2021 (6/9/2021)

Project URL: <https://aka.ms/sqlclientproject>

Report Abuse: <https://www.nuget.org/packages/Microsoft.Data.SqlClient/3.0.0/ReportAbuse>

Search Solution Explorer (Ctrl+)

Solution 'SensorSystem' (1 of 1 project)

- SensorSystem**
  - Dependencies
  - Analyzers
  - Frameworks
  - Form1.cs
  - Form1.Designer.cs
  - Form1.resx
  - Program.cs

Solution Explorer Team Explorer

Properties

<https://www.halvorsen.blog>



# Windows Forms App

Hans-Petter Halvorsen

# Windows Forms App

## Create a new project

### Recent project templates

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

Search for templates (Alt+S)  [Clear all](#)

C# Windows Desktop

- NUnit Test Project**  
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.  
C# Linux macOS Windows Desktop Test Web
- Windows Forms App (.NET Framework)**  
A project for creating an application with a Windows Forms (WinForms) user interface  
C# Windows Desktop
- Windows Forms App**  
A project template for creating a .NET Windows Forms (WinForms) App.  
C# Windows Desktop
- WPF Application**

## Configure your new project

Windows Forms App C# Windows Desktop

### Project name

### Location

 ...

### Solution name [?](#)

Place solution and project in the same directory

Back

Next

## Additional information

Windows Forms App C# Windows Desktop

### Target Framework [?](#)

Back

Create

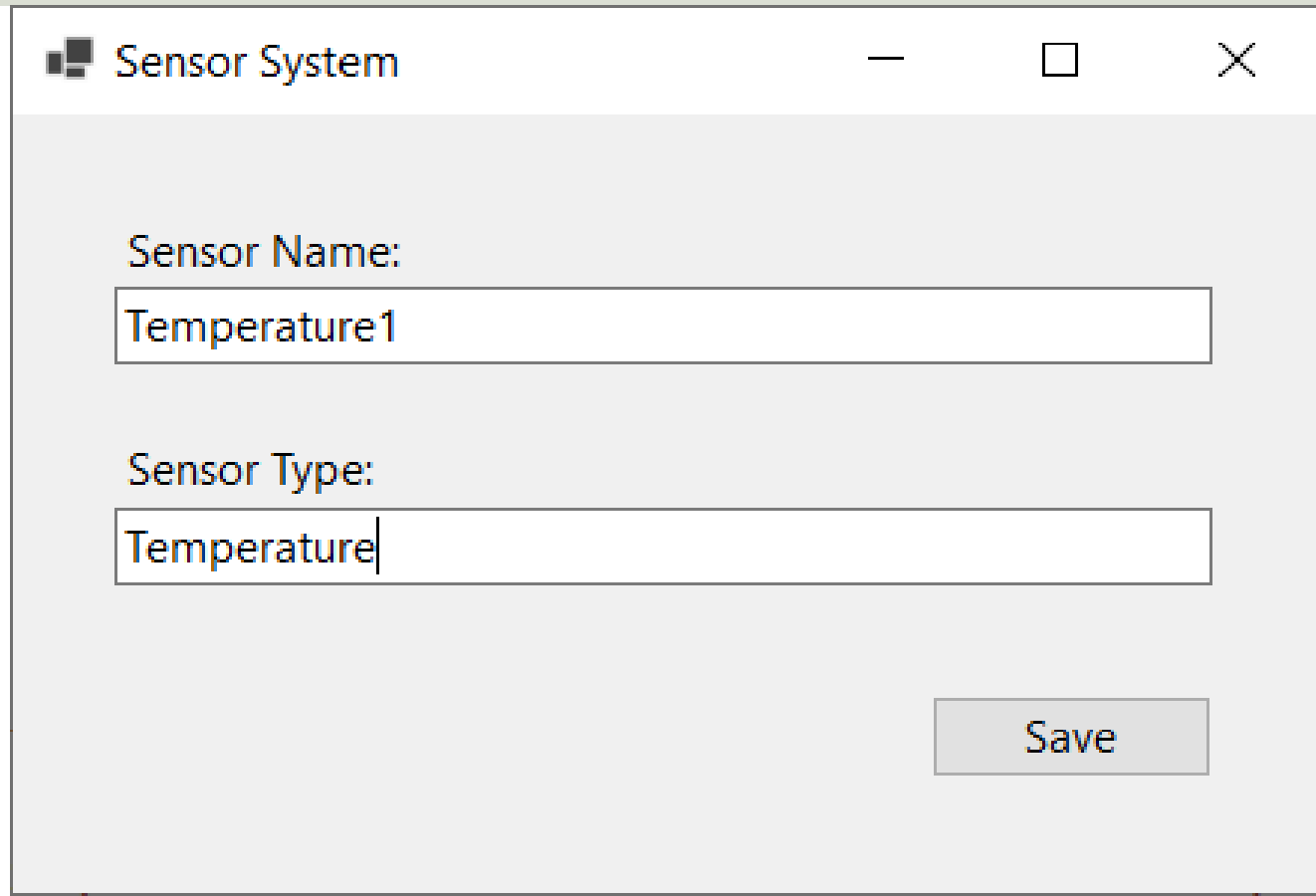
<https://www.halvorsen.blog>



# Basic Example

Hans-Petter Halvorsen

# Basic Example



Sensor System

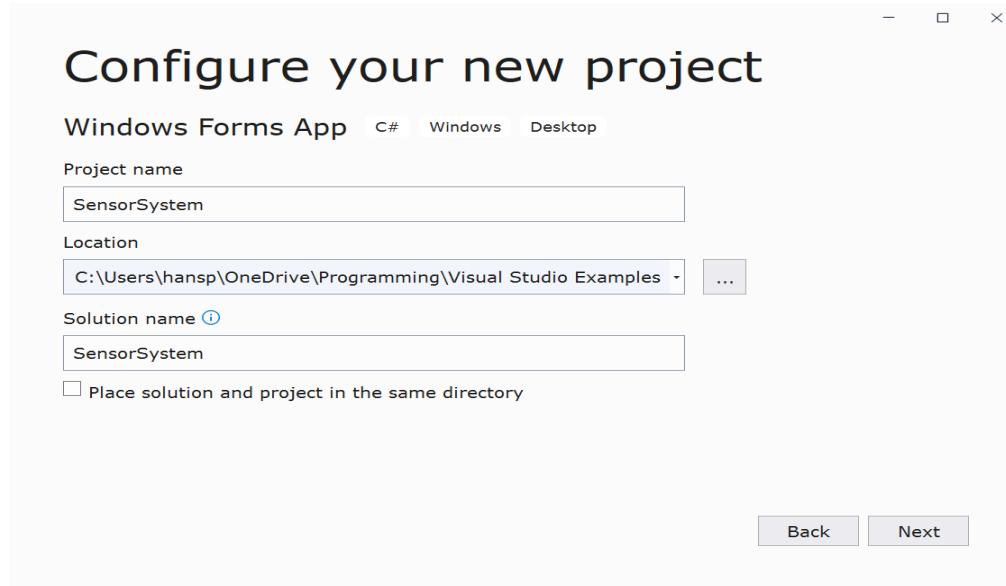
Sensor Name:  
Temperature1

Sensor Type:  
Temperature

Save

# Basic Example

- Sensor Type
  - Temperature, Pressure, ..
- Sensor Name



Configure your new project

Windows Forms App C# Windows Desktop

Project name

SensorSystem

Location

C:\Users\hansp\OneDrive\Programming\Visual Studio Examples ...

Solution name ⓘ

SensorSystem

Place solution and project in the same directory

Back Next

# Database

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

# Visual Studio



## Configure your new project

Windows Forms App C# Windows Desktop

Project name

Location

Solution name [i](#)

Place solution and project in the same directory



```
1 using System;
2 using Microsoft.Data.SqlClient;
3 using System.Windows.Forms;
4
5 namespace SensorSystem
6 {
7     3 references
8     public partial class Form1 : Form
9     {
10         1 reference
11         public Form1()
12         {
13             InitializeComponent();
14         }
15
16         1 reference
17         private void btnSave_Click(object sender, EventArgs e)
18         {
19             string connectionString = "Data Source= ;Initial Catalog=SENSORSYSTEM;Integrated Security=True";
20
21             string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) VALUES (" + "\"" + txtSensorName.Text + "\"" + "," + "\"" + txtSensorType.Text + "\"" + ")";
22
23             SqlConnection con = new SqlConnection(connectionString);
24
25             con.Open();
26             SqlCommand sc = new SqlCommand(sqlQuery, con);
27             sc.ExecuteNonQuery();
28             con.Close();
29         }
30     }
31 }
```

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

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

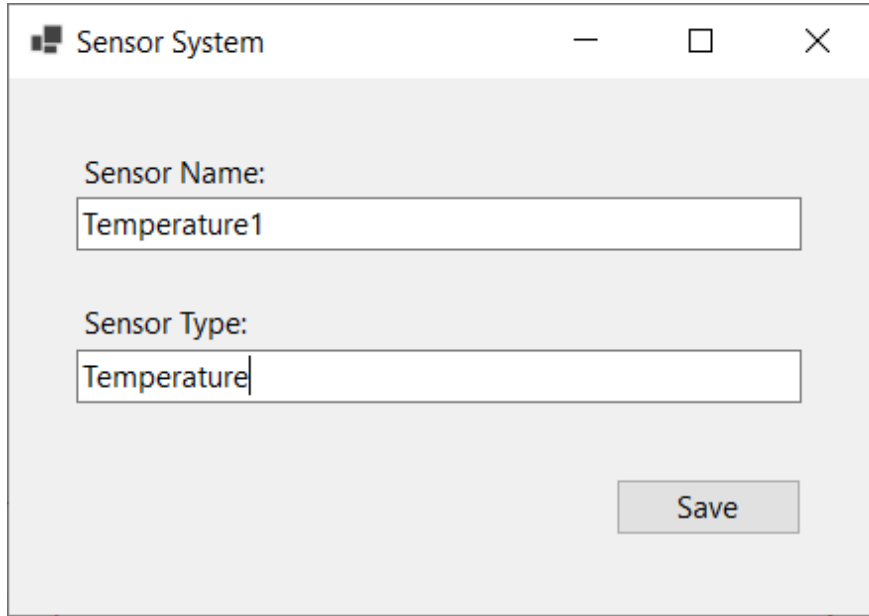
        private void btnSave_Click(object sender, EventArgs e)
        {
            string connectionString = "Data Source=xxx;Initial Catalog=xxx;Integrated Security=True";

            string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType)
                VALUES (" + "'" + txtSensorName.Text + "'" + "," + "'" + txtSensorType.Text + "'" + ")";

            SqlConnection con = new SqlConnection(connectionString);

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

# Running the Application



The image shows a window titled "Sensor System" with standard window controls (minimize, maximize, close). Inside the window, there are two text input fields. The first is labeled "Sensor Name:" and contains the text "Temperature1". The second is labeled "Sensor Type:" and contains the text "Temperature". Below these fields is a "Save" button.

```
INSERT INTO SENSOR (SensorName, SensorType)
VALUES ('Temperature1', 'Temperature')
```

# Certificate Issue?

Do you get this one:

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

## Exception Unhandled

Microsoft.Data.SqlClient.SqlException: 'A connection was successfully established with the server, but then an error occurred during the login process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority that is not trusted.)'

### Inner Exception

Win32Exception: The certificate chain was issued by an authority that is not

[Ask Copilot](#) | [Show Call Stack](#) | [View Details](#) | [Copy Details](#) | [Start Live Share session..](#)

▶ [Exception Settings](#)

Add `TrustServerCertificate=True` in the Connection String:

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

File Edit View Project Tools Window Help



Object Explorer

Connect

- BOOKS
- BOOKSTORE
- CHART
- LIBRARY
- LOGGINGSYSTEM
- MEASUREMENTS
- MEASUREMENTSYSTEM
- PRESSURESYSTEM
- SENSORSYSTEM**
- Database Diagrams
- Tables
  - System Tables
  - FileTables
  - External Tables
  - Graph Tables
  - dbo.SENSOR
- Views
- External Resources
- Synonyms
- Programmability

SQLQuery1.sql - N...ORSYSTEM (sa (57))\*

```
select * from SENSOR
```

Select \* from SENSOR

100 %

Results Messages

	SensorTypeId	SensorName	SensorType
1	1	Temperature1	Temperature

We see that the data has been stored in the Database

Query executed successfully.

NUCHPH\SQLEXPRESS (15.0 RTM) | sa (57) | SENSORSYSTEM | 00:00:00 | 1 rows

# Improvements

- Use App.config
- Use SQL Parameters
- Use Stored Procedure
- Use Try ... Catch
- Create separate Classes and Methods
- Improve Database structure
- ...

<https://www.halvorsen.blog>



# App.config

Hans-Petter Halvorsen

# Use App.config

App.config

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3
4
5 <connectionStrings>
6   <add name="DatabaseConnectionString" connectionString="Data Source=NUCHPH\SQLEXPRESS;Initial Catalog=SENSORSYSTEM;Trusted_Connection=True"
7     providerName="System.Data.SqlClient" />
8 </connectionStrings>
9
10
11 </configuration>
```

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

<connectionStrings>
  <add name="DatabaseConnectionString" connectionString="Data Source=x;Initial Catalog=x;Trusted_Connection=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>

</configuration>
```



```
1 using System;
2 using Microsoft.Data.SqlClient;
3 using System.Configuration;
4 using System.Windows.Forms;
5
6 namespace SensorSystem
7 {
8     3 references
9     public partial class Form1 : Form
10    {
11        1 reference
12        public Form1()
13        {
14            InitializeComponent();
15        }
16
17        1 reference
18        private void btnSave_Click(object sender, EventArgs e)
19        {
20            string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;
21            string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) VALUES (" + "'" + txtSensorName.Text + "'" + "," + "'" + txtSensorType.Te
22            SqlConnection con = new SqlConnection(connectionString);
23            con.Open();
24            SqlCommand sc = new SqlCommand(sqlQuery, con);
25            sc.ExecuteNonQuery();
26            con.Close();
27        }
28    }
29 }
```

# Code

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

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

        private void btnSave_Click(object sender, EventArgs e)
        {
            string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

            string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType)
                VALUES ('" + txtSensorName.Text + "'" + "," + "'" + txtSensorType.Text + "'" + ")";

            SqlConnection con = new SqlConnection(connectionString);

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

<https://www.halvorsen.blog>



# SQL Parameters

Hans-Petter Halvorsen

# Use SQL Parameters

- Using SQL Parameters are safer than putting the values into the string because the parameters are passed to the database separately, protecting against SQL injection attacks.
- It is also be more efficient if you execute the same SQL repeatedly with different parameters.
- The Example is showing Windows Forms using C#
- Other Languages like PHP, Python, etc. offer the same functionality

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

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

        private void btnSave_Click(object sender, EventArgs e)
        {
            string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;
            string sqlQuery = "INSERT INTO SENSOR (SensorName, SensorType) VALUES (@sensorname, @sensortype)";

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

            SqlCommand cmd = new SqlCommand(sqlQuery, con);

            var sensorNameParameter = new SqlParameter("sensorname", System.Data.SqlDbType.VarChar);
            sensorNameParameter.Value = txtSensorName.Text;
            cmd.Parameters.Add(sensorNameParameter);

            var sensorTypeParameter = new SqlParameter("sensortype", System.Data.SqlDbType.VarChar);
            sensorTypeParameter.Value = txtSensorType.Text;
            cmd.Parameters.Add(sensorTypeParameter);

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

<https://www.halvorsen.blog>



# Stored Procedure

Hans-Petter Halvorsen

# Use Stored Procedure

- A Stored Procedure is a premade SQL Script which you can use inside your C# Code
- Here you also use SQL Parameters
- Using Stored Procedure and SQL Parameters prevent SQL Injection

# Stored Procedure

```
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
```



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

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

        private void btnSave_Click(object sender, EventArgs e)
        {
            string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

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

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

            string sensorName = txtSensorName.Text;
            string sensorType = txtSensorType.Text;

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

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

<https://www.halvorsen.blog>



# Try .. Catch ..

Hans-Petter Halvorsen

# Use Try ... Catch

- When executing C# code, different errors may occur
- When an error occurs, C# will normally stop and generate an error message.
- Typically, we just want to show an Error Message to the user without stopping the application
- Then we can use Try ... Catch

# Try ... Catch

```
try
{
    // Put your ordinary Code here
}
catch (Exception ex)
{
    // Code for Handling Errors
}
```

# Code

```
private void btnSave_Click(object sender, EventArgs e)
{
    string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

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

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

        string sensorName = txtSensorName.Text;
        string sensorType = txtSensorType.Text;

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

        cmd.ExecuteNonQuery();
        con.Close();
    }
    catch
    {
        MessageBox.Show("Error Writing Data to Database");
    }
}
```

<https://www.halvorsen.blog>



# Classes and Methods

Hans-Petter Halvorsen

# Create Classes and Methods

- So far, we have used the Button Click Event Method

`btnSave_Click()` and then we created all code inside that Method

- Better to create separate Classes and Methods

# Create a Separate Method

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

private void SaveData ()
{
    string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;

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

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

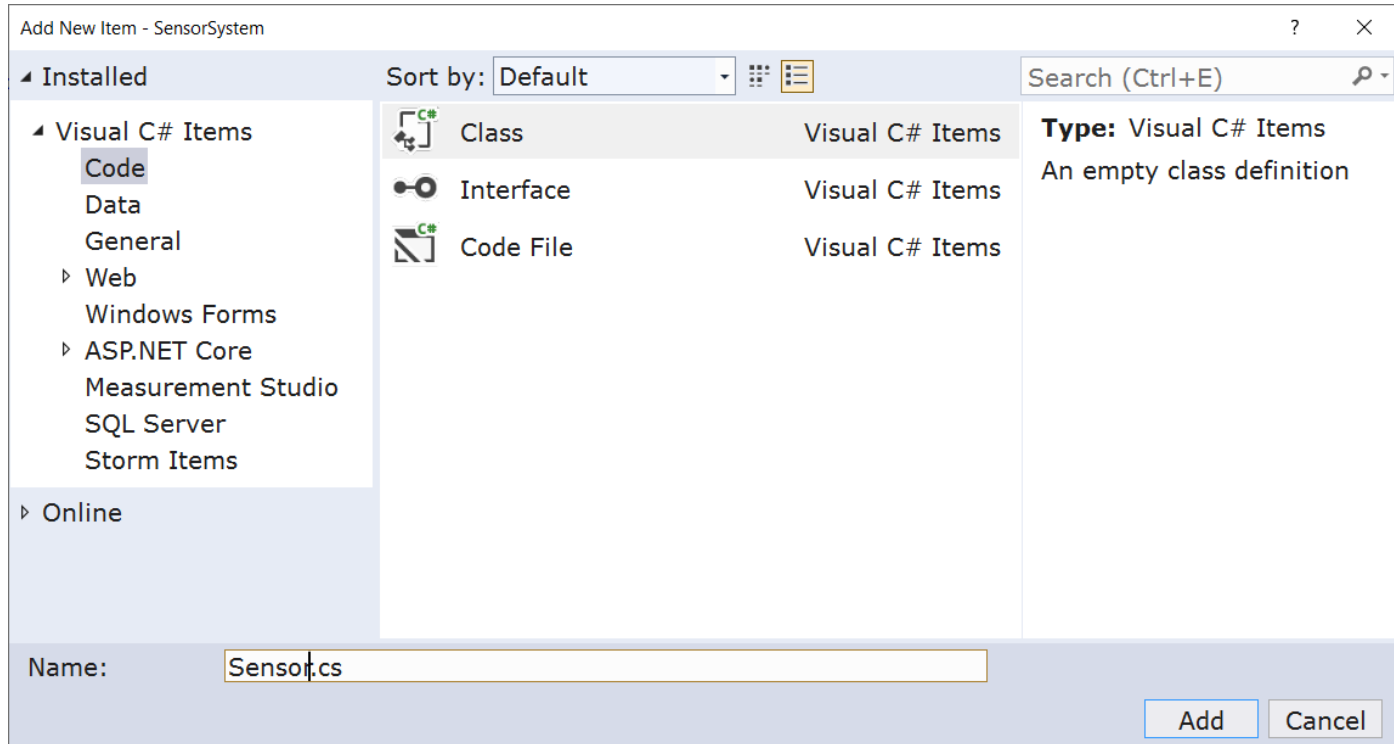
        string sensorName = txtSensorName.Text;
        string sensorType = txtSensorType.Text;

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

        cmd.ExecuteNonQuery();
        con.Close();
    }
    catch
    {
        MessageBox.Show("Error Writing Data to Database");
    }
}
```



# Create a Class and Method



# Create a Class and Method

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

            try
            {
                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();
            }
            catch
            {
                MessageBox.Show("Error Writing Data to Database");
            }
        }
    }
}
```

# Using the Class and Method

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

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

        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);
        }
    }
}
```

<https://www.halvorsen.blog>



# Improve Database

Hans-Petter Halvorsen

# Updated Database

```
CREATE TABLE SENSOR_TYPE
(
SensorTypeId int PRIMARY KEY IDENTITY (1,1),
SensorType varchar(50) NOT NULL UNIQUE
)
GO
```

```
CREATE TABLE SENSOR
(
SensorId int PRIMARY KEY IDENTITY (1,1),
SensorName varchar(50) UNIQUE NOT NULL,
SensorTypeId int NOT NULL FOREIGN KEY REFERENCES SENSOR_TYPE(SensorTypeId)
)
GO
```

# Test Data

```
insert into SENSOR_TYPE (SensorType) values ('Temperature')
insert into SENSOR_TYPE (SensorType) values ('Pressure')
insert into SENSOR_TYPE (SensorType) values ('Level')
insert into SENSOR_TYPE (SensorType) values ('Proximity')
```

# Update Stored Procedure

```
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

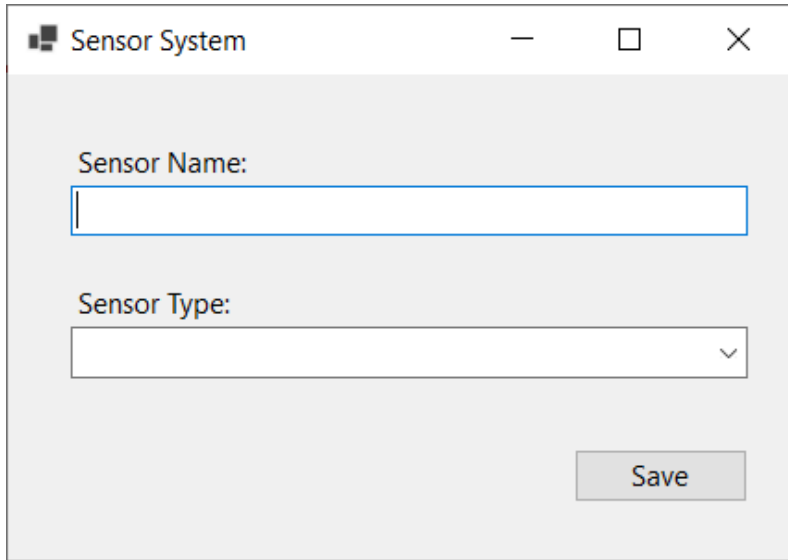
DECLARE
@SensorTypeId int

SELECT @SensorTypeId=SensorTypeId FROM SENSOR_TYPE WHERE SensorType=@SensorType

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

GO
```

# Updated GUI



Sensor System

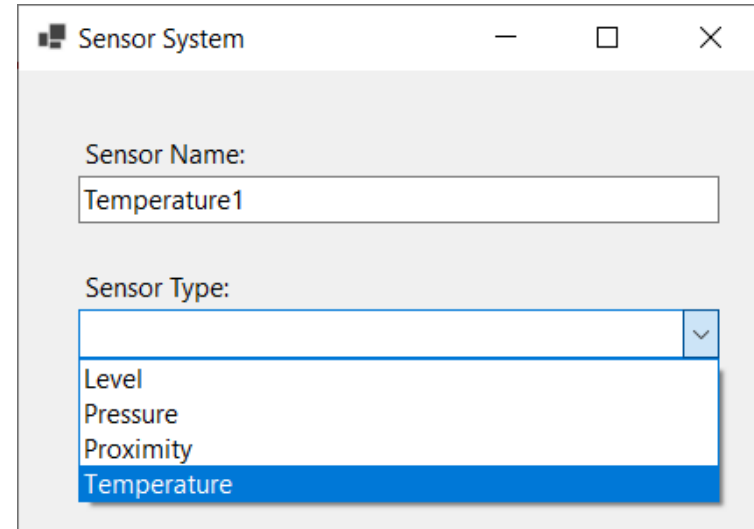
Sensor Name:

Sensor Type:

Save

Sensor Types are now a Drop-down List. This prevent you from spelling mistakes, and getting Sensor Types like “Temperature”, “Tmperature”, ..

The different Sensor Types will now be retrieved from the SQL Server Database



Sensor System

Sensor Name:

Sensor Type:

- Level
- Pressure
- Proximity
- Temperature



## SensorType.cs

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

namespace SensorSystem.Classes
{
    class SensorType
    {
        string connectionString = ConfigurationManager.ConnectionStrings["DatabaseConnectionString"].ConnectionString;
        public int SensorTypeId { get; set; }
        public string SensorTypeName { get; set; }

        public List<SensorType> GetSensorTypes()
        {
            List<SensorType> sensorTypeList = new List<SensorType>();

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

            string sqlQuery = "select SensorTypeId, SensorType from SENSOR_TYPE order by SensorType";
            SqlCommand cmd = new SqlCommand(sqlQuery, con);

            SqlDataReader dr = cmd.ExecuteReader();

            if (dr != null)
            {
                while (dr.Read())
                {
                    SensorType sensorType = new SensorType();

                    sensorType.SensorTypeId = Convert.ToInt32(dr["SensorTypeId"]);
                    sensorType.SensorTypeName = dr["SensorType"].ToString();

                    sensorTypeList.Add(sensorType);
                }
            }
            con.Close();
            return sensorTypeList;
        }
    }
}
```

## Sensor.cs

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

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

        public void SaveSensorData(string sensorName, string sensorType)
        {
            try
            {
                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();
            }
            catch
            {
                MessageBox.Show("Error Writing Data to Database");
            }
        }
    }
}
```

## 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();
            FillSensorTypeComboBox();
        }

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

        private void FillSensorTypeComboBox()
        {
            SensorType sensorType = new SensorType();

            List<SensorType> sensorTypeList = new List<SensorType>();

            sensorTypeList = sensorType.GetSensorTypes();

            foreach (SensorType sensorTypeItem in sensorTypeList)
            {
                comboSensorType.Items.Add(sensorTypeItem.SensorTypeName);
            }
        }

        private void SaveData()
        {
            string sensorName = txtSensorName.Text;
            string sensorType = comboSensorType.SelectedItem.ToString();

            Sensor sensor = new Sensor();
            sensor.SaveSensorData(sensorName, sensorType);
        }
    }
}
```

# Discussions

- We have made a simple Windows Forms App for saving Data to a SQL Server Database
- First, I made it work, then I improved the code step by step
- Still, lots of improvements to make, but I leave that for you

# 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>

