https://www.halvorsen.blog

ASP.NET Core Basics

# ASP.NET Core

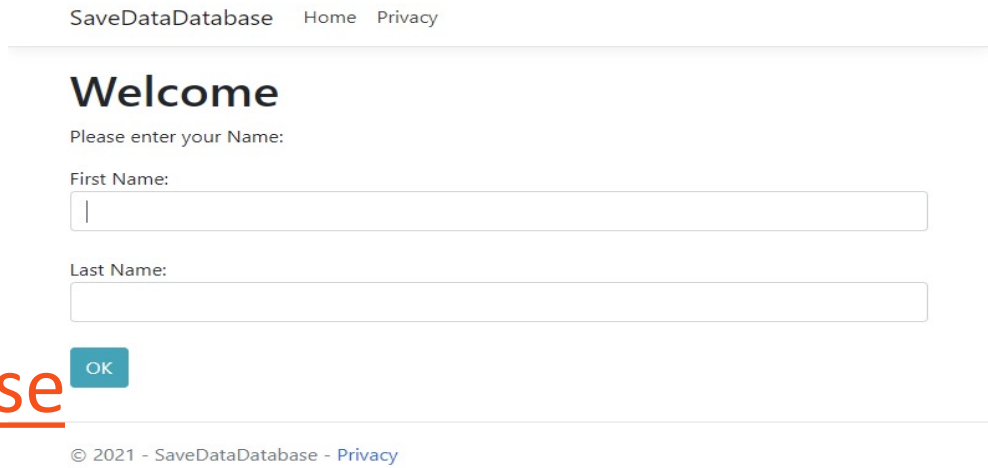
## Save Data to Database

Hans-Petter Halvorsen

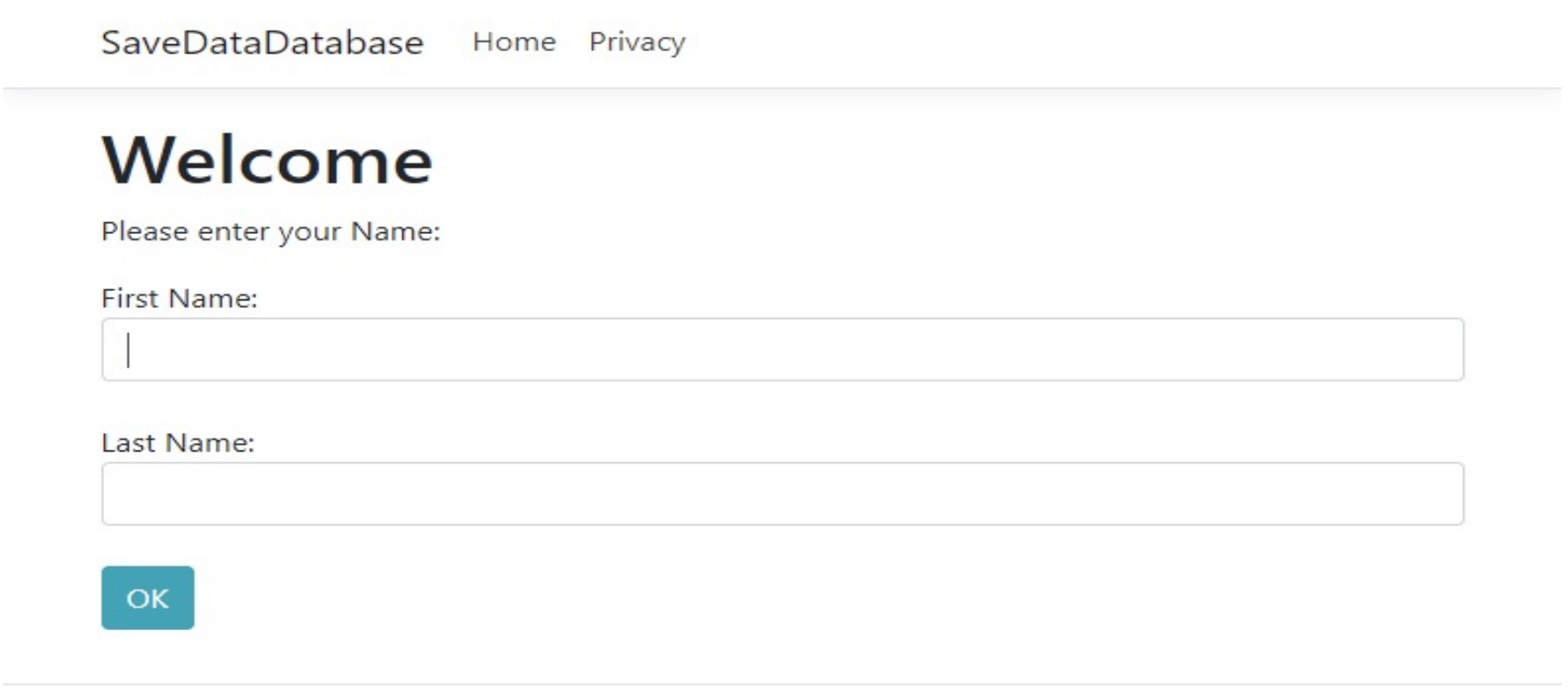
ASP.NET Core Basics

# Contents

- ASP.NET Core
- SQL Server
- Practical Examples
  - Save Data to Database
  - Create Method
  - Create Class
  - Create Stored Procedure
  - Try .. Catch ..



# Example


SaveDataDatabase
Home
Privacy
Welcome
Please enter your Name:
First Name:
Last Name:
OK
© 2021 - SaveDataDatabase - Privacy

# ASP.NET Core

Hans-Petter Halvorsen

Table of Contents

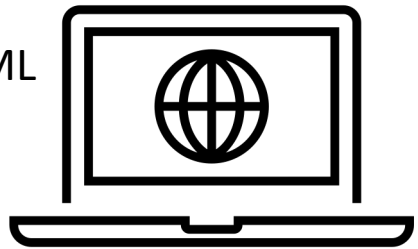
# ASP.NET Core

- ASP.NET Core is an open-source web framework, created by Microsoft, for building web applications and web services
- You create ASP.NET Core Web Applications using Visual Studio and the C# Programming language
- ASP.NET Resources:

https://www.halvorsen.blog/documents/programming/web/aspnet

# Web Fundamentals

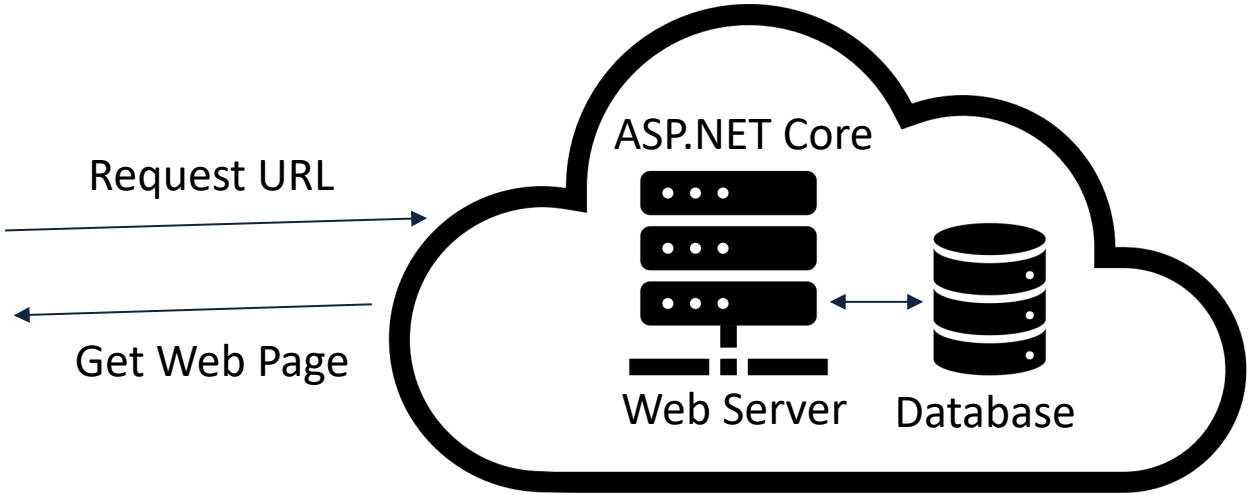
## Client-side




Local Computer
with Web Browser

A pure HTML File is sent to
your local Web Browser

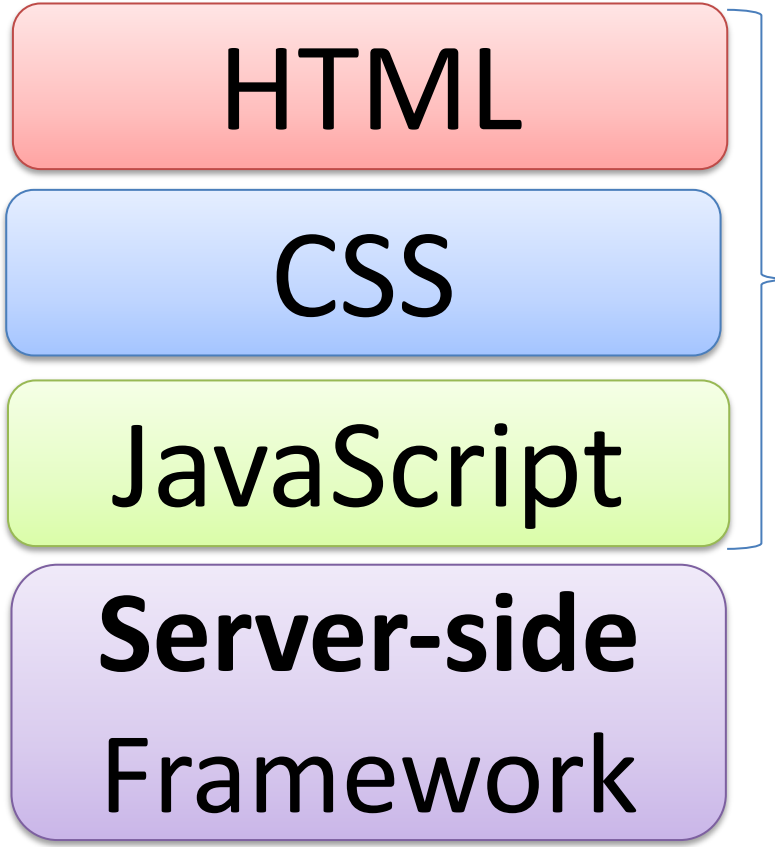
## Server-side




The Cloud/Internet

Server-side Code (C#) is executed on the Server

# Web Fundamentals




**HTML** is the standard markup language for creating Web pages

**CSS** is the language we use to style an HTML document. CSS describes how HTML elements should be displayed.

**JavaScript** is the programming language of the Web. JavaScript is probably the most used programming language in the world.

**ASP.NET Core** (C#), PHP or similar

https://www.halvorsen.blog

# SQL Server

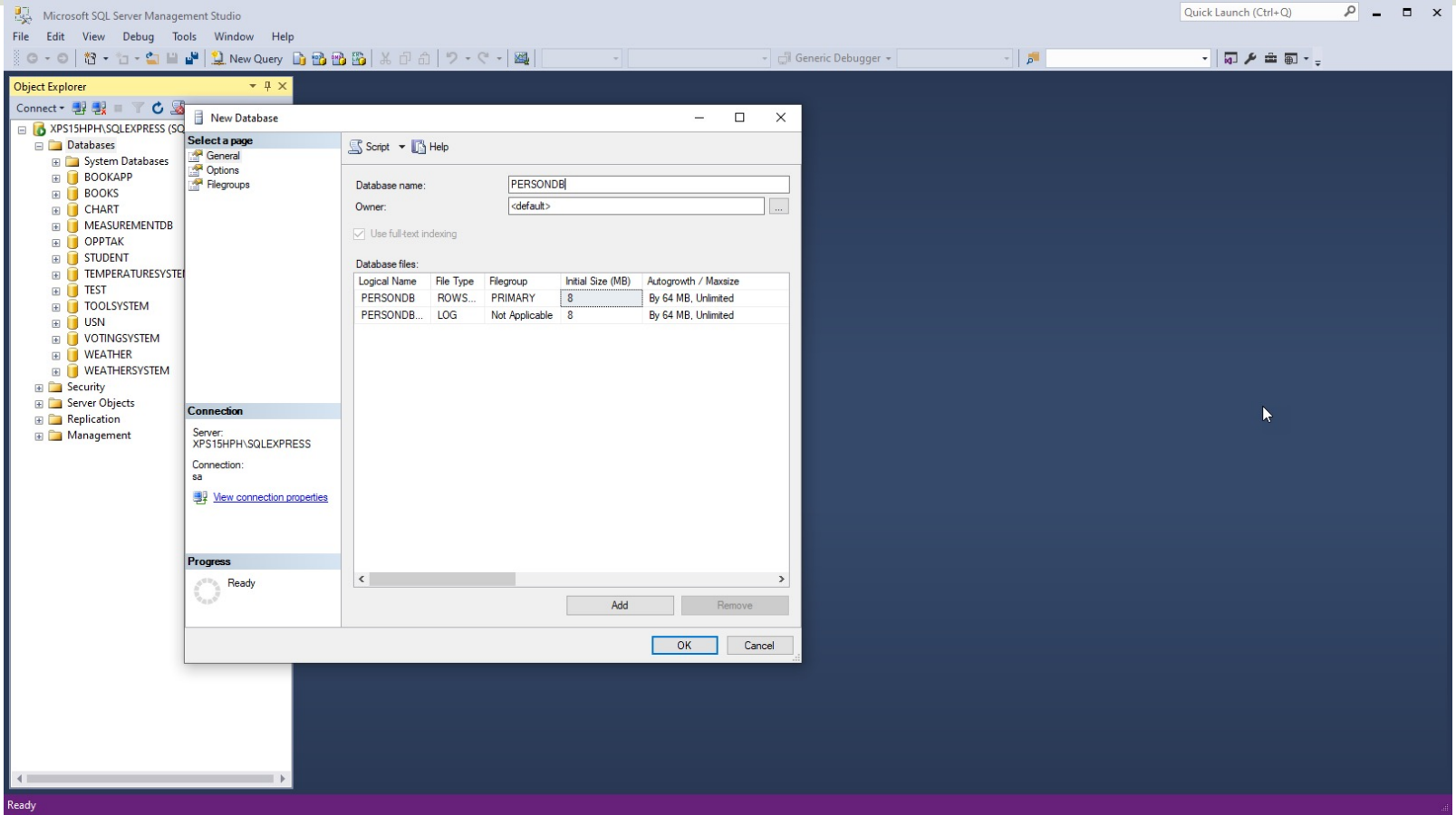
Hans-Petter Halvorsen

Table of Contents

# 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

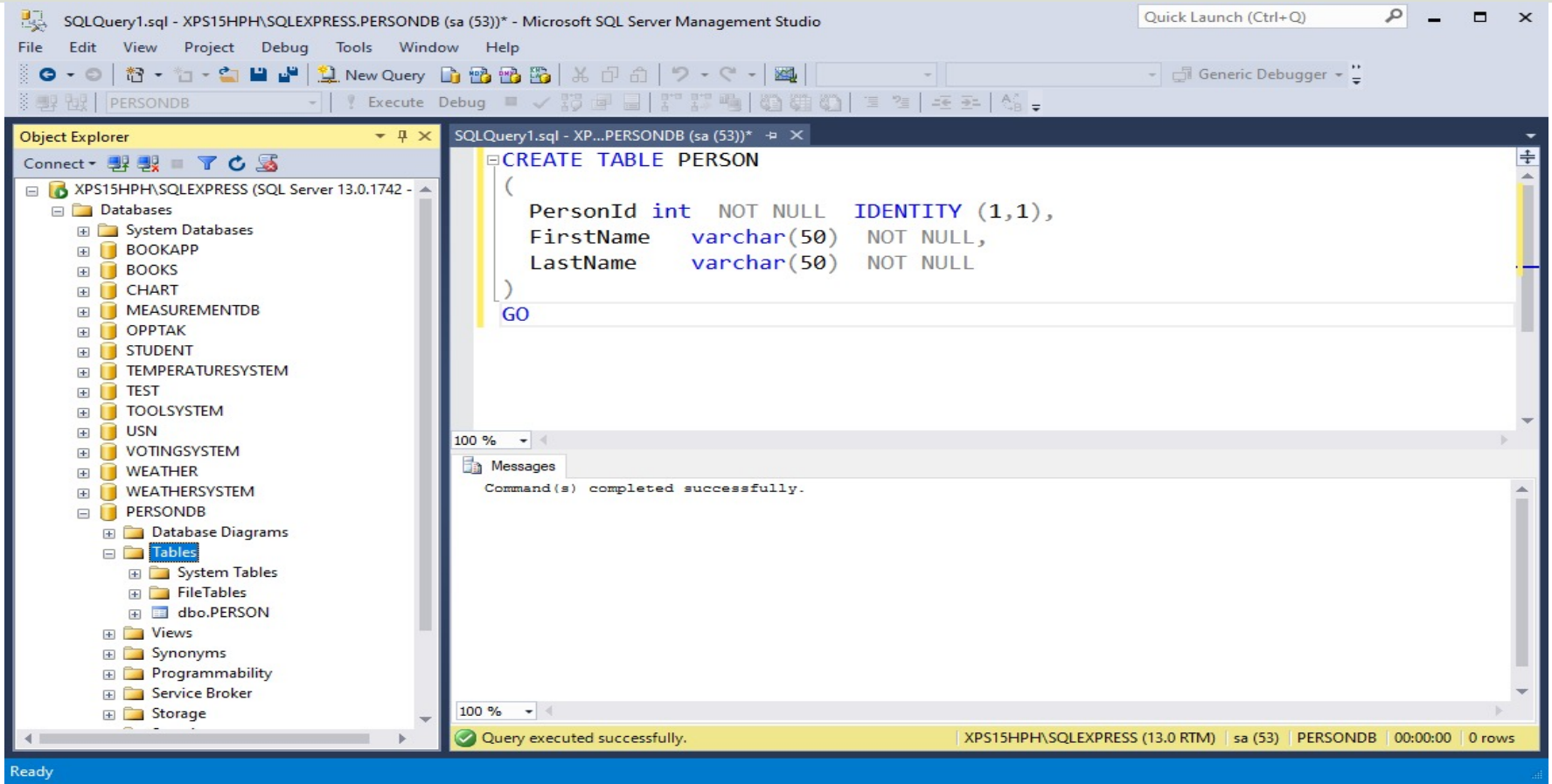
# Create Database



# Database Table

```
CREATE TABLE PERSON
(
  PersonId int  NOT NULL  IDENTITY (1,1),
  FirstName    varchar(50)  NOT NULL,
  LastName     varchar(50)  NOT NULL
)
GO
```

# Create Table



# Code Example

Hans-Petter Halvorsen

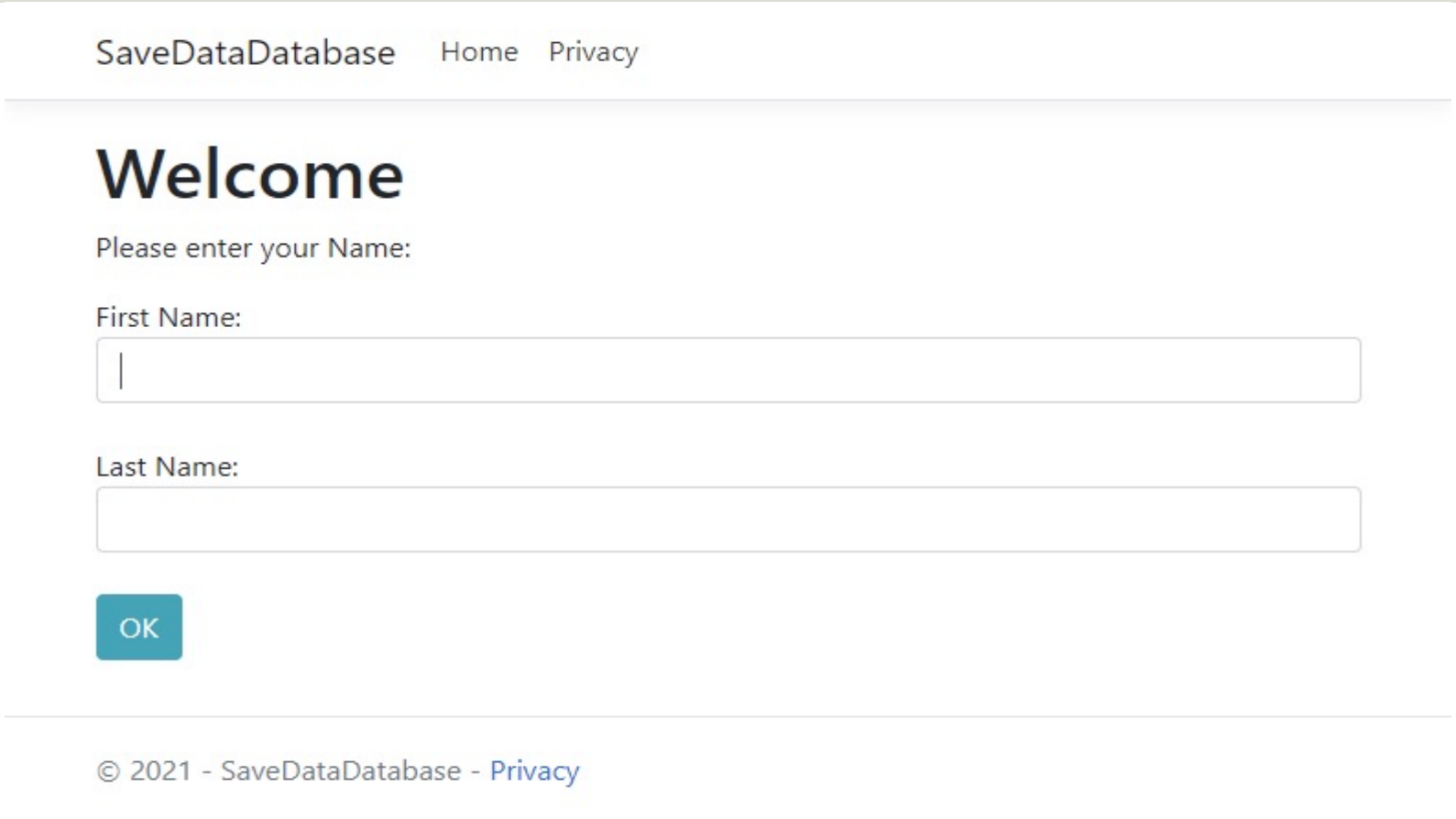
Table of Contents

# Code Examples

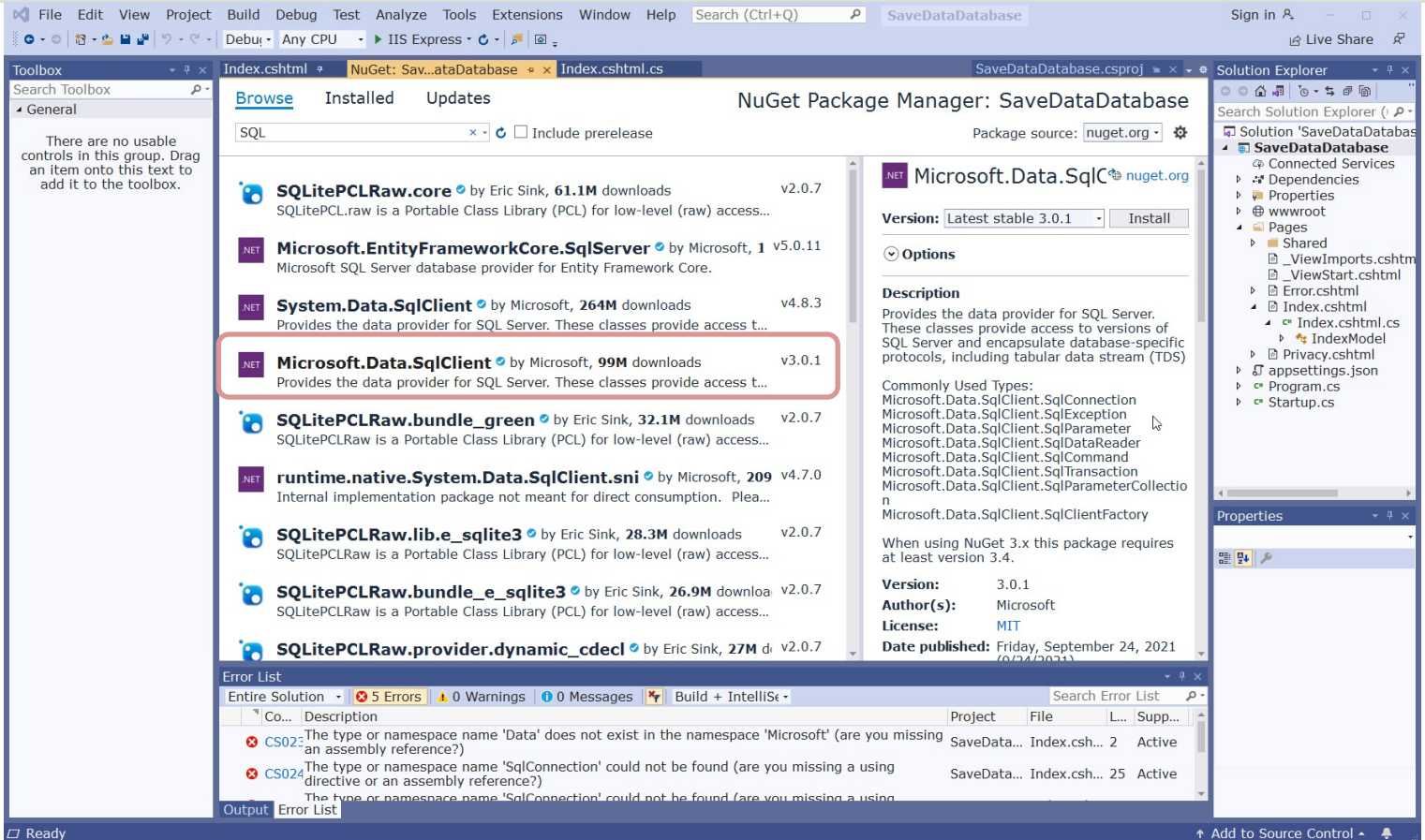
Note!

- 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.

# ASP.NET Core Example



# NuGet


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
SaveDataDatabase
Sign in
Live Share
Debug
Any CPU
IIS Express
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Index.cshtml
NuGet: Sav...ataDatabase
Index.cshtml.cs
SaveDataDatabase.csproj
Browse
Installed
Updates
NuGet Package Manager: SaveDataDatabase
SQL
Include prerelease
Package source: nuget.org
SQLitePCLRaw.core by Eric Sink, 61.1M downloads v2.0.7
SQLitePCL.raw is a Portable Class Library (PCL) for low-level (raw) access...
Microsoft.EntityFrameworkCore.SqlServer by Microsoft, 1 v5.0.11
Microsoft SQL Server database provider for Entity Framework Core.
System.Data.SqlClient by Microsoft, 264M downloads v4.8.3
Provides the data provider for SQL Server. These classes provide access t...
Microsoft.Data.SqlClient by Microsoft, 99M downloads v3.0.1
Provides the data provider for SQL Server. These classes provide access t...
SQLitePCLRaw.bundle_green by Eric Sink, 32.1M downloads v2.0.7
SQLitePCLRaw is a Portable Class Library (PCL) for low-level (raw) access...
runtime.native.System.Data.SqlClient.sni by Microsoft, 209 v4.7.0
Internal implementation package not meant for direct consumption. Plea...
SQLitePCLRaw.lib.e_sqlite3 by Eric Sink, 28.3M downloads v2.0.7
SQLitePCLRaw is a Portable Class Library (PCL) for low-level (raw) access...
SQLitePCLRaw.bundle_e_sqlite3 by Eric Sink, 26.9M downloa v2.0.7
SQLitePCLRaw is a Portable Class Library (PCL) for low-level (raw) access...
SQLitePCLRaw.provider.dynamic_cdecl by Eric Sink, 27M d v2.0.7
Microsoft.Data.SqlC nuget.org
Version: Latest stable 3.0.1
Install
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.SqlParameterCollection
Microsoft.Data.SqlClient.SqlClientFactory
When using NuGet 3.x this package requires at least version 3.4.
Version: 3.0.1
Author(s): Microsoft
License: MIT
Date published: Friday, September 24, 2021
Solution Explorer
Search Solution Explorer
Solution 'SaveDataDatabas
SaveDataDatabase
Connected Services
Dependencies
Properties
wwwroot
Pages
Shared
_ViewImports.cshtm
_ViewStart.cshtml
Error.cshtml
Index.cshtml
Index.cshtml.cs
IndexModel
Privacy.cshtml
appsettings.json
Program.cs
Startup.cs
Properties
Error List
Entire Solution
5 Errors
0 Warnings
0 Messages
Build + IntelliSe
Search Error List
Co... Description Project File L... Supp...
CS025 The type or namespace name 'Data' does not exist in the namespace 'Microsoft' (are you missing an assembly reference?) SaveData... Index.csh... 2 Active
CS024 The type or namespace name 'SqlConnection' could not be found (are you missing a using directive or an assembly reference?) SaveData... Index.csh... 25 Active
Output
Error List
Ready
Add to Source Control

# Index.cshtml

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div>

    <h1>Welcome</h1>
    <p>Please enter your Name:</p>

    <form name="NameForm" id="NameForm" method="post">

        First Name:
        <br />
        <input name="FirstName" type="text" class="form-control input-lg" autofocus required />
        <br />

        Last Name:
        <br />
        <input name="LastName" type="text" class="form-control input-lg" required />
        <br />

        <input id="SaveButton" type="submit" value="Save" class="btn btn-info" />

    </form>

</div>
```

# Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Data.SqlClient;

namespace SaveDataDatabase.Pages
{
    public class IndexModel : PageModel
    {
        public void OnGet()
        {

        }

        public void OnPost()
        {
            string firstName = Request.Form["FirstName"];
            string lastName = Request.Form["LastName"];

            string connectionString = "Data Source=XXX;Initial Catalog=PERSONDB;Integrated Security=True";

            string sqlQuery = "INSERT INTO PERSON (FirstName, LastName)
                               VALUES (" + "'" + firstName + "'" + "," + "'" + lastName + "'" + ")";

            SqlConnection con = new SqlConnection(connectionString);

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

# Create Method

Hans-Petter Halvorsen

Table of Contents

# ASP.NET Core Example


The Web Page is the same as in previous Example

# Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Data.SqlClient;

namespace SaveDataDatabase.Pages
{
    public class IndexModel : PageModel
    {
        public void OnGet()
        {
        }
        public void OnPost()
        {
            string firstName = Request.Form["FirstName"];
            string lastName = Request.Form["LastName"];

            SaveData(firstName, lastName);
        }

        void SaveData(string firstName, string lastName)
        {
            string connectionString = "Data Source=XXX;Initial Catalog=PERSONDB;Integrated Security=True";

            string sqlQuery = "INSERT INTO PERSON (FirstName, LastName)
                              VALUES (" + "'" + firstName + "'" + "," + "'" + lastName + "'" + ")";

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

# Create Class

Hans-Petter Halvorsen

Table of Contents

# ASP.NET Core Example


The Web Page is the same as in previous Example

# Person.cs

```
using Microsoft.Data.SqlClient;

namespace SaveDataDatabase.Models
{
    public class Person
    {
        public void SaveData(string firstName, string lastName)
        {
            string connectionString = "Data Source=XXX;Initial Catalog=PERSONDB;
                                       Integrated Security=True";

            string sqlQuery = "INSERT INTO PERSON (FirstName, LastName)
                              VALUES (" + "'" + firstName + "'" + "," + "'" + lastName + "'" + ")";

            SqlConnection con = new SqlConnection(connectionString);

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

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using SaveDataDatabase.Models;

namespace SaveDataDatabase.Pages
{
    public class IndexModel : PageModel
    {
        public void OnGet()
        {
        }
        public void OnPost()
        {
            string firstName = Request.Form["FirstName"];
            string lastName = Request.Form["LastName"];

            Person person = new Person();

            person.SaveData(firstName, lastName);
        }
    }
}
```

# Create Stored Procedure

Hans-Petter Halvorsen

Table of Contents

# ASP.NET Core Example


The Web Page is the same as in previous Example

# Stored Procedure

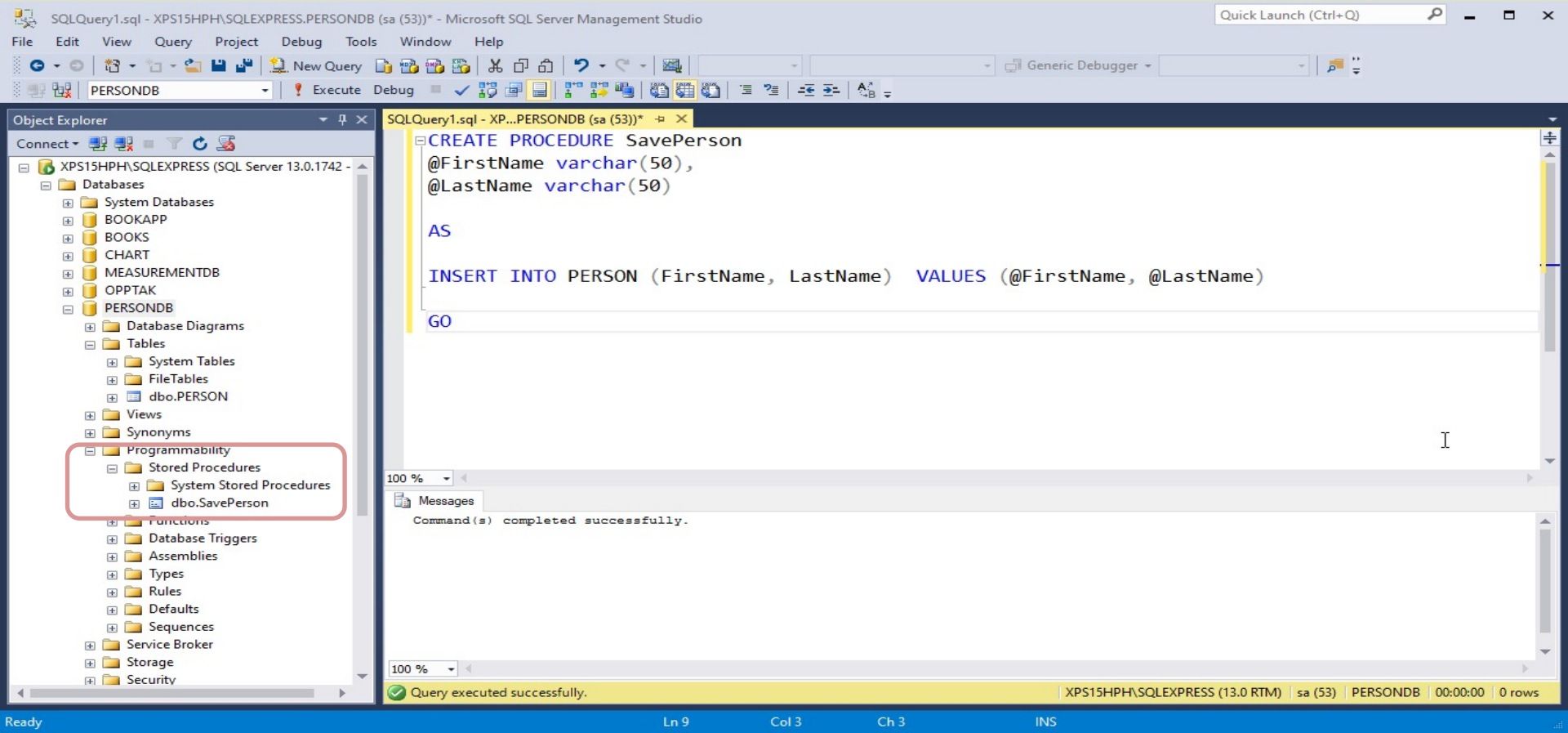
```
CREATE TABLE PERSON
(
  PersonId int  NOT NULL  IDENTITY (1,1),
  FirstName    varchar(50)  NOT NULL,
  LastName     varchar(50)  NOT NULL
)
GO
```

```
CREATE PROCEDURE SavePerson
@FirstName varchar(50),
@LastName varchar(50)

AS

INSERT INTO PERSON (FirstName, LastName)
VALUES (@FirstName, @LastName)

GO
```

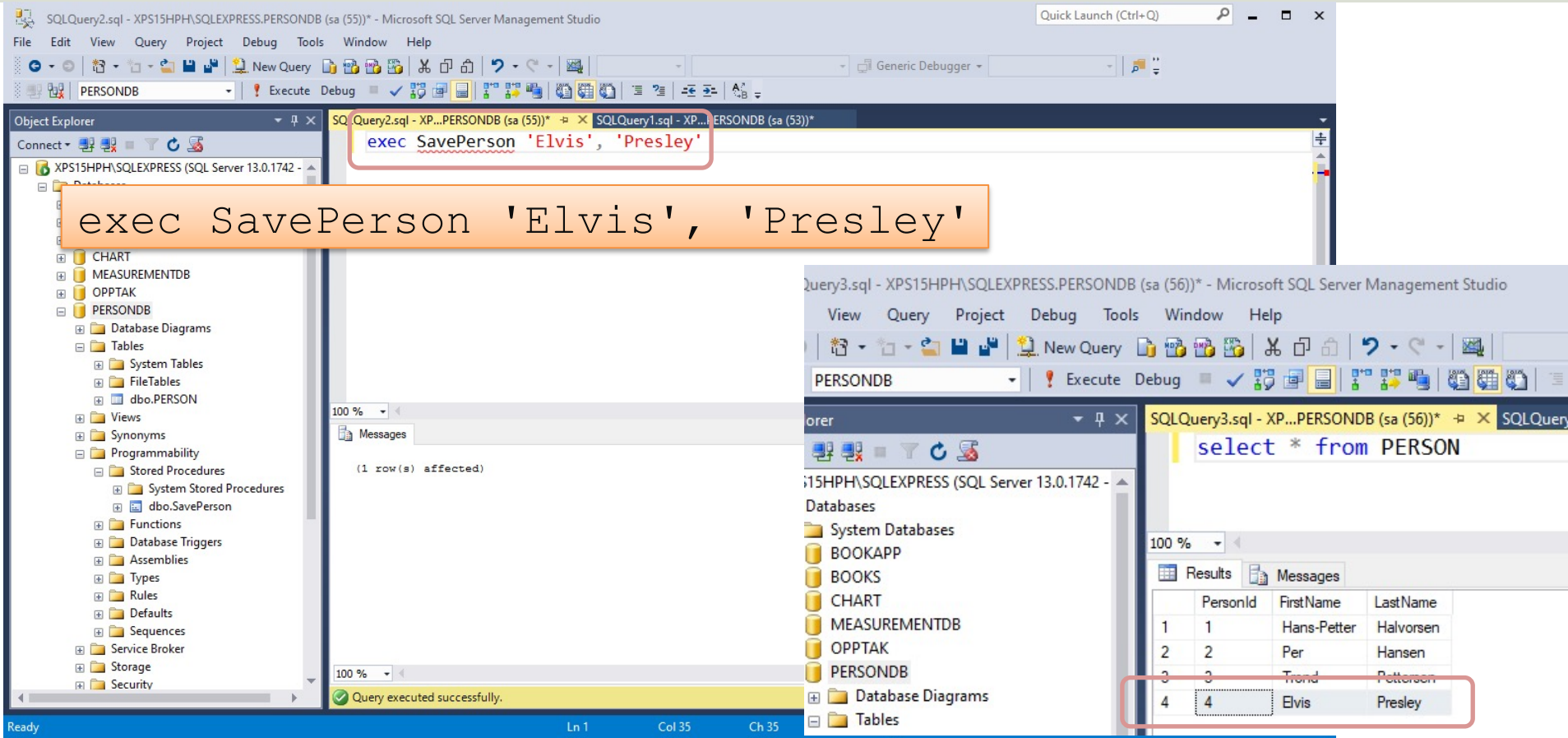
Input Parameters

SQL Query that inserts FirstName and LastName into the PERSON table

# Stored Procedure



# Test the Stored Procedure



Person.cs

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

namespace SaveDataDatabase.Models
{
    public class Person
    {
        public void SaveData(string firstName, string lastName)
        {
            string connectionString = "Data Source=XXX;Initial Catalog=PERSONDB;
                                                Integrated Security=True";

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

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

            cmd.Parameters.Add(new SqlParameter("@FirstName", firstName));
            cmd.Parameters.Add(new SqlParameter("@LastName", lastName));

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

# Try .. Catch..

Hans-Petter Halvorsen

Table of Contents

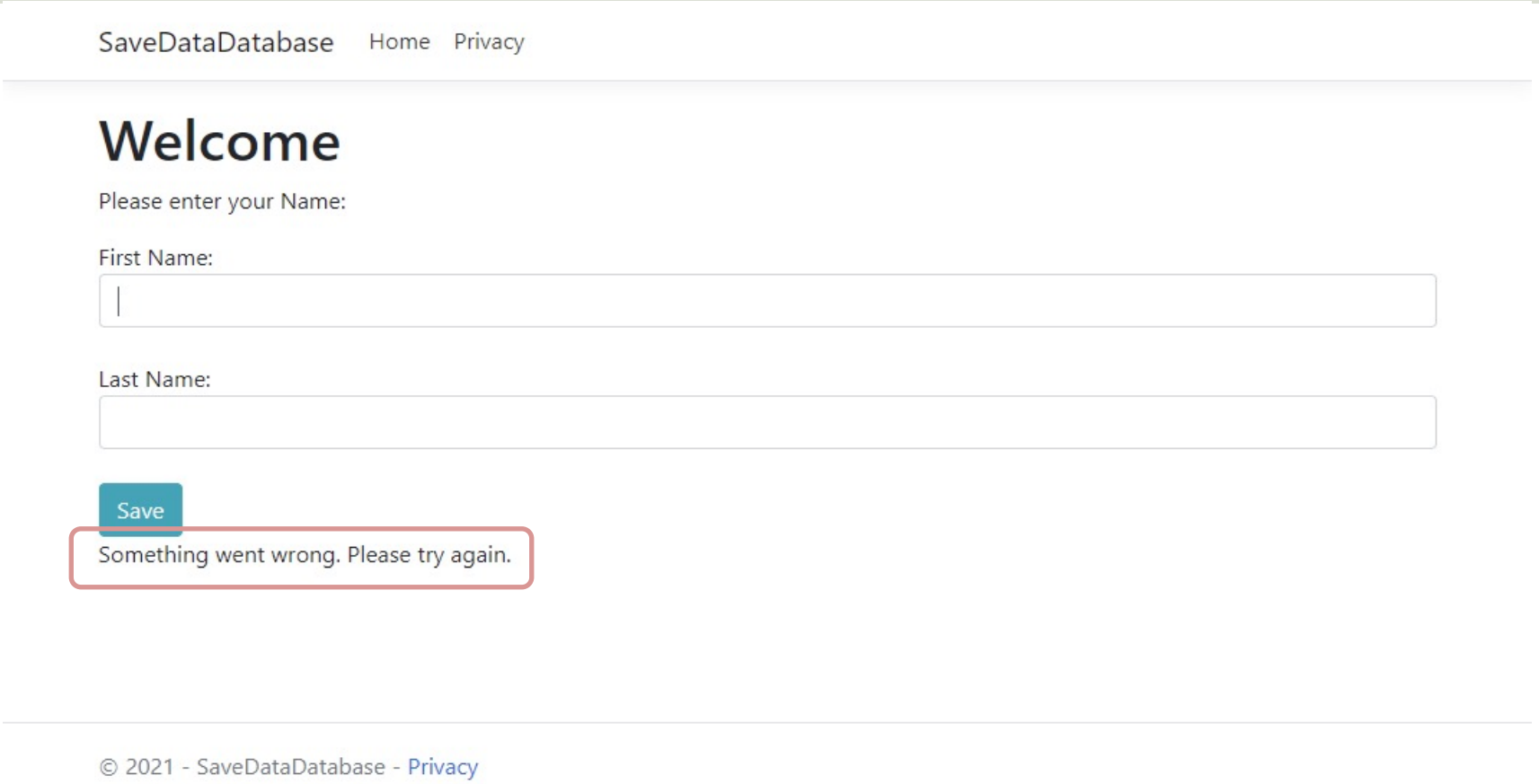
# Use Try … Catch

- 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
}
```

# Example


SaveDataDatabase
Home
Privacy
Welcome
Please enter your Name:
First Name:
Last Name:
Save
Something went wrong. Please try again.
© 2021 - SaveDataDatabase - Privacy

Person.cs

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

namespace SaveDataDatabase.Models
{
    public class Person
    {
        public string SaveData(string firstName, string lastName)
        {
            string connectionString = "Data Source=XXX;Initial Catalog=PERSONDB;Integrated Security=True";
            string errorMessage = null;

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

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

                cmd.Parameters.Add(new SqlParameter("@FirstName", firstName));
                cmd.Parameters.Add(new SqlParameter("@LastName", lastName));

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

                errorMessage = null;
                return errorMessage;
            }
            catch
            {
                return errorMessage = "Something went wrong. Please try again.";
            }
        }
    }
}
```

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using SaveDataDatabase.Models;

namespace SaveDataDatabase.Pages
{
    public class IndexModel : PageModel
    {
        public string errorMessage;

        public void OnGet()
        {

        }
        public void OnPost()
        {
            string firstName = Request.Form["FirstName"];
            string lastName = Request.Form["LastName"];

            Person person = new Person();

            errorMessage = person.SaveData(firstName, lastName);
        }
    }
}
```

Index.cshtml

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div>
    <h1>Welcome</h1>
    <p>Please enter your Name:</p>

    <form name="NameForm" id="NameForm" method="post">

        First Name:
        <br />
        <input name="FirstName" type="text" class="form-control input-lg" autofocus required />
        <br />

        Last Name:
        <br />
        <input name="LastName" type="text" class="form-control input-lg" required />
        <br />

        <input id="SaveButton" type="submit" value="Save" class="btn btn-info" />

    </form>

    <p>
        @Model.errorMessage
    </p>
</div>
```

# appSettings.json

Hans-Petter Halvorsen

Table of Contents

# appSettings.json

- appSettings.json is a Configuration File used in ASP.NET Core
- Here you can, e.g., put the Connection String for your Database
- In that way you don’t need to hardcode this inside your C# code
- I have shown this in other ASP.NET Core Videos

# Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog