

<https://www.halvorsen.blog>



# ASP.NET Core CRUD Application

Hans-Petter Halvorsen

# Contents

- SQL Server
- Visual Studio
- Models
- Web Pages
  - Books
  - New Book
  - Edit Book
  - Delete Book

# Introduction

- A basic **CRUD Application** example will be given using **ASP.NET Core**.
- If you have never used ASP.NET Core, I suggest the following Videos:
  - ASP.NET Core - Hello World  
<https://youtu.be/lcQsWYgQXK4>
  - ASP.NET Core – Introduction  
<https://youtu.be/zkOtiBcwo8s>

# CRUD Application

- **CRUD** Application means **Creating, Reading, Updating** and **Deleting** Data in a Database from your Application
- The CRUD application presented here can be a foundation for all your Applications in ASP.NET Core
- Typically, all Applications today need to communicate with a Database and has CRUD functionality
- When you have learned to create a basic CRUD Application, you have all the necessary tools you need to create any kind of Application

# ASP.NET Core Web Application

The following Application will be demonstrated here:

BookApp Home Books

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

New Book

We will retrieve these data from a SQL Server Database

# ASP.NET Core Web Application

BookApp Home Books

## Book Store

Welcome to my Book Store



Create  
Read  
Update  
Delete

# CRUD

The acronym **CRUD** refers to all the major functions that are implemented for communication with a database.

Operation	SQL	HTTP/REST API
Create	INSERT	POST
Read	SELECT	GET
Update	UPDATE	PUT
Delete	DELETE	DELETE

We will show how we can use ASP.NET Core to get (read, retrieve, select) data from the database, insert data into the database, update the data inside the database and deleting data inside the database.



# SQL Server



# SQL Server

- We will use SQL Server in this example as our database.
- You should have SQL Server locally installed on your computer
- SQL Server Express is recommended.



# Database

# SQL Server - Create Database

The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the 'BOOKS' database structure, including tables like 'AUTHOR', 'BOOK', 'CATEGORY', and 'PUBLISHER'. The main query editor contains the following SQL code:

```
/*Books Tables Script*/  
/*  
AUTHOR  
PUBLISHER  
CATEGORY  
BOOK  
*/  
  
if not exists (select * from dbo.sysobjects where id = object_id(N'[AUTHOR]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)  
CREATE TABLE [AUTHOR]  
(  
    [AuthorId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,  
    [AuthorName] [varchar](50) NOT NULL UNIQUE,  
    [Address] [varchar](50) NULL,  
    [Phone] [varchar](50) NULL,  
)
```

The Messages pane at the bottom shows the execution results, indicating that the table was created successfully and that 1 row was affected for each of the 10 messages displayed.

Query executed successfully. XPS15HPH\SQLEXPRESS (13.0.RTM) sa (56) BOOKS 00:00:00 0 rows

# Database Tables

- AUTHOR
- PUBLISHER
- CATEGORY
- BOOK

You can use SQL Server Management Studio in order to run this SQL Script

# Database Tables

```
CREATE TABLE [AUTHOR]
(
    [AuthorId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [AuthorName] [varchar](50) NOT NULL UNIQUE,
    [Address] [varchar](50) NULL,
    [Phone] [varchar](50) NULL,
    [PostCode] [varchar](50) NULL,
    [PostAddress] [varchar](50) NULL,
)
```

```
CREATE TABLE [PUBLISHER]
(
    [PublisherId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [PublisherName] [varchar](50) NOT NULL UNIQUE,
    [Description] [varchar](1000) NULL,
    [Address] [varchar](50) NULL,
    [Phone] [varchar](50) NULL,
    [PostCode] [varchar](50) NULL,
    [PostAddress] [varchar](50) NULL,
    [EMail] [varchar](50) NULL,
)
```

```
CREATE TABLE [BOOK]
(
    [BookId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [Title] [varchar](50) NOT NULL UNIQUE,
    [ISBN] [varchar](20) NOT NULL,
    [PublisherId] [int] NOT NULL FOREIGN KEY REFERENCES [PUBLISHER]([PublisherId]),
    [AuthorId] [int] NOT NULL FOREIGN KEY REFERENCES [AUTHOR]([AuthorId]),
    [CategoryId] [int] NOT NULL FOREIGN KEY REFERENCES [CATEGORY]([CategoryId]),
    [Description] [varchar](1000) NULL,
    [Year] [date] NULL,
    [Edition] [int] NULL,
    [AverageRating] [float] NULL,
)
```

```
CREATE TABLE [CATEGORY]
(
    [CategoryId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [CategoryName] [varchar](50) NOT NULL UNIQUE,
    [Description] [varchar](1000) NULL,
)
```

# Stored Procedures

The following Stored Procedures will be used:

- CreateBook
- UpdateBook
- DeleteBook

# Create Book

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

## **CREATE PROCEDURE CreateBook**

```
@Title varchar(50),
@Isbn varchar(20),
@PublisherName varchar(50),
@AuthorName varchar(50),
@CategoryName varchar(50)
AS

if not exists (select * from CATEGORY where CategoryName = @CategoryName)
INSERT INTO CATEGORY (CategoryName) VALUES (@CategoryName)

if not exists (select * from AUTHOR where AuthorName = @AuthorName)
INSERT INTO AUTHOR (AuthorName) VALUES (@AuthorName)

if not exists (select * from PUBLISHER where PublisherName = @PublisherName)
INSERT INTO PUBLISHER (PublisherName) VALUES (@PublisherName)

if not exists (select * from BOOK where Title = @Title)
INSERT INTO BOOK (Title, ISBN, PublisherId, AuthorId, CategoryId)
VALUES
(
@Title,
@ISBN,
(select PublisherId from PUBLISHER where PublisherName=@PublisherName),
(select AuthorId from AUTHOR where AuthorName=@AuthorName),
(select CategoryId from CATEGORY where CategoryName=@CategoryName)
)
GO
```

# UpdateBook

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

## **CREATE PROCEDURE UpdateBook**

```
@BookId int,
@Title varchar(50),
@ISBN varchar(20),
@PublisherName varchar(50),
@AuthorName varchar(50),
@CategoryName varchar(50)
AS
```

```
if not exists (select * from CATEGORY where CategoryName = @CategoryName)
INSERT INTO CATEGORY (CategoryName) VALUES (@CategoryName)
```

```
if not exists (select * from AUTHOR where AuthorName = @AuthorName)
INSERT INTO AUTHOR (AuthorName) VALUES (@AuthorName)
```

```
if not exists (select * from PUBLISHER where PublisherName = @PublisherName)
INSERT INTO PUBLISHER (PublisherName) VALUES (@PublisherName)
```

```
UPDATE BOOK SET
Title = @Title,
ISBN = @ISBN,
PublisherId = (select PublisherId from PUBLISHER where PublisherName=@PublisherName),
AuthorId = (select AuthorId from AUTHOR where AuthorName=@AuthorName),
CategoryId = (select CategoryId from CATEGORY where CategoryName=@CategoryName)
WHERE BookId = @BookId
```

```
GO
```



# DeleteBook

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

```
CREATE PROCEDURE DeleteBook
```

```
@BookId int
AS
```

```
delete from BOOK where BookId=@BookId
```

```
GO
```

# Views

```
IF EXISTS (SELECT name
FROM sysobjects
WHERE name = 'GetBookData'
AND type = 'V')
DROP VIEW GetBookData
GO
```

```
CREATE VIEW GetBookData
AS
```

```
SELECT
BOOK.BookId,
BOOK.Title,
BOOK.ISBN,
PUBLISHER.PublisherName,
AUTHOR.AuthorName,
CATEGORY.CategoryName

FROM BOOK
INNER JOIN AUTHOR ON BOOK.AuthorId = AUTHOR.AuthorId
INNER JOIN PUBLISHER ON BOOK.PublisherId = PUBLISHER.PublisherId
INNER JOIN CATEGORY ON BOOK.CategoryId = CATEGORY.CategoryId
```

```
GO
```

# Test Data

```
--CATEGORY -----
INSERT INTO CATEGORY (CategoryName) VALUES ('Science')
GO
INSERT INTO CATEGORY (CategoryName) VALUES ('Programming')
GO
INSERT INTO CATEGORY (CategoryName) VALUES ('Novel')
GO

--AUTHOR -----
INSERT INTO AUTHOR (AuthorName) VALUES ('Knut Hamsun')
GO
INSERT INTO AUTHOR (AuthorName) VALUES ('Gilbert Strang')
GO
INSERT INTO AUTHOR (AuthorName) VALUES ('J.R.R Tolkien')
GO
INSERT INTO AUTHOR (AuthorName) VALUES ('Dorf Bishop')
GO

--PUBLISHER -----
INSERT INTO PUBLISHER (PublisherName) VALUES ('Prentice Hall')
GO
INSERT INTO PUBLISHER (PublisherName) VALUES ('Wiley')
GO
INSERT INTO PUBLISHER (PublisherName) VALUES ('McGraw-Hill')
GO
```

```
--BOOK -----
INSERT INTO BOOK (Title, ISBN, PublisherId, AuthorId, CategoryId)
VALUES
(
  'Introduction to Linear Algebra',
  '0-07-066781-0',
  (select PublisherId from PUBLISHER where PublisherName='Prentice Hall'),
  (select AuthorId from AUTHOR where AuthorName='Gilbert Strang'),
  (select CategoryId from CATEGORY where CategoryName='Science')
)
GO

INSERT INTO BOOK (Title, ISBN, PublisherId, AuthorId, CategoryId)
VALUES
(
  'Modern Control System',
  '1-08-890781-0',
  (select PublisherId from PUBLISHER where PublisherName='Wiley'),
  (select AuthorId from AUTHOR where AuthorName='Dorf Bishop'),
  (select CategoryId from CATEGORY where CategoryName='Programming')
)
GO

INSERT INTO BOOK (Title, ISBN, PublisherId, AuthorId, CategoryId)
VALUES
(
  'The Lord of the Rings',
  '2-09-066556-2',
  (select PublisherId from PUBLISHER where PublisherName='McGraw-Hill'),
  (select AuthorId from AUTHOR where AuthorName='J.R.R Tolkien'),
  (select CategoryId from CATEGORY where CategoryName='Novel')
)
GO
```



# Visual Studio

# Create New Project

Create a new project

Recent project templates

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

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

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

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

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

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

Back Next

# Select ASP.NET Core Web Application

**Create a new ASP.NET Core web application**

.NET Core | ASP.NET Core 3.1

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

[Get additional project templates](#)

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

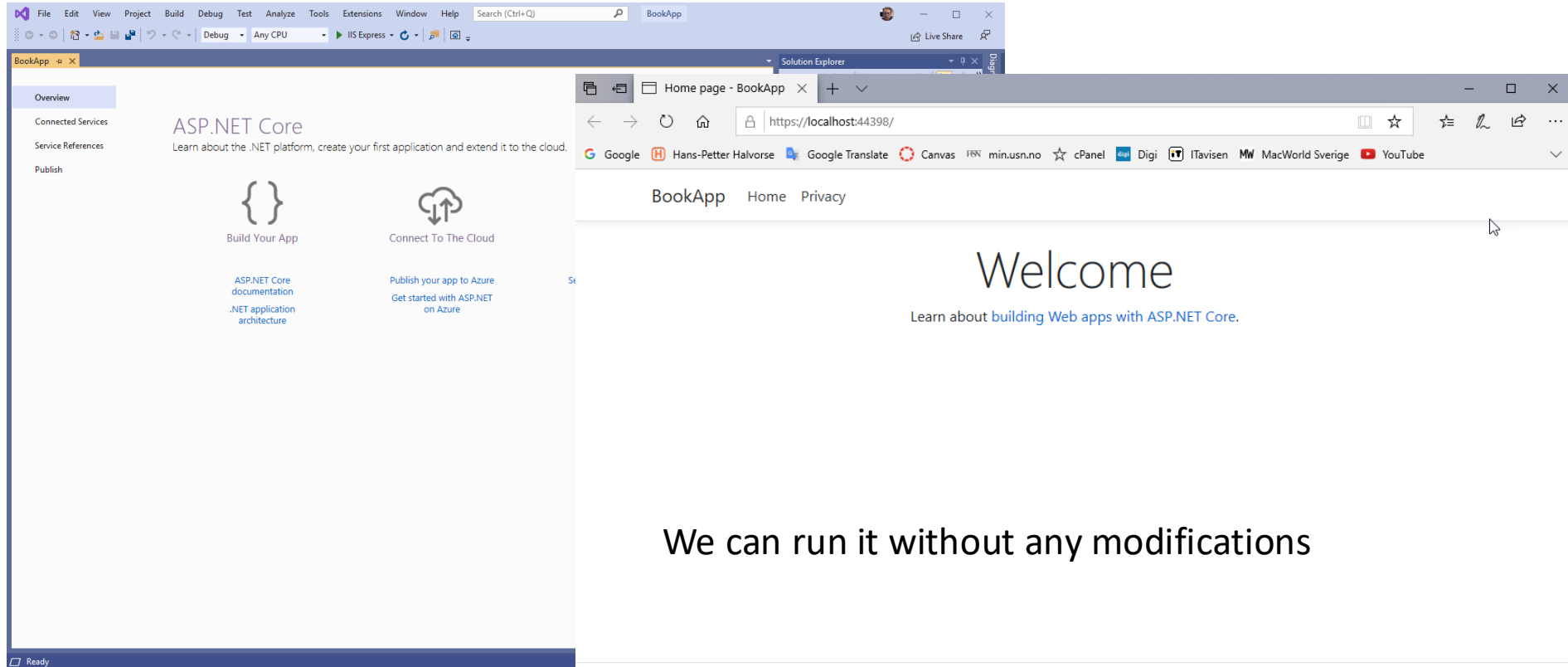
**Advanced**  
 Configure for HTTPS  
 Enable Docker Support  
(Requires [Docker Desktop](#))  
Linux

Author: Microsoft  
Source: .NET Core 3.1.0

[Back](#) [Create](#)

# Visual Studio Project

The web application project becomes as shown here



The image shows a screenshot of Visual Studio and a web browser. The Visual Studio window displays the 'ASP.NET Core' overview page, which includes links for 'Build Your App' and 'Connect To The Cloud'. The web browser window shows the 'Home page - BookApp' at the URL 'https://localhost:44398/'. The browser content includes a navigation menu with 'BookApp', 'Home', and 'Privacy' links, and a main heading 'Welcome' with the subtext 'Learn about building Web apps with ASP.NET Core.'

We can run it without any modifications



# ADO.NET



# ADO.NET

- ADO.NET is the core data access technology for .NET languages.
- **System.Data.SqlClient** (or the newer **Microsoft.Data.SqlClient**) is the provider or namespace you typically use to connect to an SQL Server.
- Typically we need to add the necessary **NuGet** package for that.

# NuGet

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

The screenshot shows the NuGet Package Manager interface for a project named 'MeasurementApp'. The search bar contains 'sql' and the package source is set to 'nuget.org'. The search results list several packages, with 'System.Data.SqlClient' highlighted in a red box. The details panel on the right shows the selected package 'System.Data.SqlClient' with version 'Latest stable 4.8.0' and an 'Install' button. The description and commonly used types are also visible.

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

**System.Data.SqlClient** by Microsoft, 64.1M downloads v4.8.0  
Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, including tabular data stream (TDS)

**Microsoft.EntityFrameworkCore.SqlServer** by Microsoft, 43.3M downloads v3.1.0  
Microsoft SQL Server database provider for Entity Framework Core.

**runtime.native.System.Data.SqlClient.sni** by Microsoft, 34.6M downloads v4.7.0  
Internal implementation package not meant for direct consumption. Please do not reference directly.

**Microsoft.Extensions.Caching.SqlServer** by Microsoft, 19.4M downloads v3.1.0  
Distributed cache implementation of Microsoft.Extensions.Caching.Distributed.IDistributedCache using Microsoft SQL Server.

**MySql.Data** by Oracle, 10.3M downloads v8.0.18  
MySql.Data.MySqlClient .Net Core Class Library

**System.Data.SqlClient** by Microsoft, 64.1M downloads v4.8.0  
Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, including tabular data stream (TDS)

**Options**

**Description**  
Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, including tabular data stream (TDS)

**Commonly Used Types:**  
System.Data.SqlClient.SqlConnection  
System.Data.SqlClient.SqlException  
System.Data.SqlClient.SqlParameter  
System.Data.SqlDbType  
System.Data.SqlClient.SqlDataReader  
System.Data.SqlClient.SqlCommand



# Models

# Create Models

```
Books.cs [X]
BookApp
    BookApp.Models.Book
1  using System;
2  using System.Collections.Generic;
3  using System.Data;
4  using System.Data.SqlClient;
5
6  namespace BookApp.Models
7  {
8      27 references
9      public class Book
10     {
11         8 references
12         public int BookId { get; set; }
13         8 references
14         public string Title { get; set; }
15         8 references
16         public string Isbn { get; set; }
17         8 references
18         public string PublisherName { get; set; }
19         8 references
20         public string AuthorName { get; set; }
21         8 references
22         public string CategoryName { get; set; }
23
24         1 reference
25         public List<Book> GetBooks(string connectionString) {...}
26
27         1 reference
28         public Book GetBookData(string connectionString, int bookId) {...}
29
30         1 reference
31         public void CreateBook(string connectionString, Book book) {...}
32
33         1 reference
34         public void EditBook(string connectionString, Book book) {...}
35
36         1 reference
37         public void DeleteBook(string connectionString, int bookId) {...}
38     }
39 }
```

- We start by creating a **Models** folder in our project using the Solutions Explorer
- Then we create a new Class (“**Book.cs**”)
- Then we create C# Code for inserting, retrieving, updating and deleting data from the Database

“Book” Class

# Book.cs

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

namespace BookApp.Models
{
    public class Book
    {
        public int BookId { get; set; }
        public string Title { get; set; }
        public string Isbn { get; set; }
        public string PublisherName { get; set; }
        public string AuthorName { get; set; }
        public string CategoryName { get; set; }

        ...
        public List<Book> GetBooks(string connectionString){}
        public Book GetBookData(string connectionString, int bookId){}
        public void CreateBook(string connectionString, Book book){}
        public void EditBook(string connectionString, Book book){}
        public void DeleteBook(string connectionString, int bookId){}
    }
}
```

# “GetBooks” Method

```
public List<Book> GetBooks (string connectionString)
{
    List<Book> bookList = new List<Book>();
    SqlConnection con = new SqlConnection(connectionString);

    string selectSQL = "select BookId, Title, Isbn, PublisherName, AuthorName, CategoryName from GetBookData";

    con.Open();

    SqlCommand cmd = new SqlCommand(selectSQL, con);

    SqlDataReader dr = cmd.ExecuteReader();

    if (dr != null)
    {
        while (dr.Read())
        {
            Book book = new Book();

            book.BookId = Convert.ToInt32(dr["BookId"]);
            book.Title = dr["Title"].ToString();
            book.Isbn = dr["ISBN"].ToString();
            book.PublisherName = dr["PublisherName"].ToString();
            book.AuthorName = dr["AuthorName"].ToString();
            book.CategoryName = dr["CategoryName"].ToString();

            bookList.Add(book);
        }
    }
    return bookList;
}
```



View

## “CreateBook” Method

```
public void CreateBook(string connectionString, Book book)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("CreateBook", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add(new SqlParameter("@Title", book.Title));
            cmd.Parameters.Add(new SqlParameter("@Isbn", book.Isbn));
            cmd.Parameters.Add(new SqlParameter("@PublisherName", book.PublisherName));
            cmd.Parameters.Add(new SqlParameter("@AuthorName", book.AuthorName));
            cmd.Parameters.Add(new SqlParameter("@CategoryName", book.CategoryName));

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```



Stored Procedure

## “GetBookData” Method

```
public Book GetBookData(string connectionString, int bookId)
{
    SqlConnection con = new SqlConnection(connectionString);

    string selectSQL = "select BookId, Title, Isbn, PublisherName, AuthorName, CategoryName
                        from GetBookData where BookId = " + bookId;

    con.Open();

    SqlCommand cmd = new SqlCommand(selectSQL, con);

    SqlDataReader dr = cmd.ExecuteReader();

    Book book = new Book();

    if (dr != null)
    {
        while (dr.Read())
        {
            book.BookId = Convert.ToInt32(dr["BookId"]);
            book.Title = dr["Title"].ToString();
            book.Isbn = dr["ISBN"].ToString();
            book.PublisherName = dr["PublisherName"].ToString();
            book.AuthorName = dr["AuthorName"].ToString();
            book.CategoryName = dr["CategoryName"].ToString();
        }
    }

    return book;
}
```



# “EditBook” Method

```
public void EditBook(string connectionString, Book book)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("UpdateBook", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add(new SqlParameter("@BookId", book.BookId));
            cmd.Parameters.Add(new SqlParameter("@Title", book.Title));
            cmd.Parameters.Add(new SqlParameter("@Isbn", book.Isbn));
            cmd.Parameters.Add(new SqlParameter("@PublisherName", book.PublisherName));
            cmd.Parameters.Add(new SqlParameter("@AuthorName", book.AuthorName));
            cmd.Parameters.Add(new SqlParameter("@CategoryName", book.CategoryName));

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

← Stored Procedure

## “DeleteBook” Method

```
public void DeleteBook(string connectionString, int bookId)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("DeleteBook", con);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.Add(new SqlParameter("@BookId", bookId));
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Stored Procedure



# Web Pages

# Web Pages

BookApp Home Books

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

New Book

BookApp Home Books

## New Book

Title:

ISBN:

Publisher:

Author:

Category:

Save

© 2019 - BookApp - Priv

© 2019 - BookApp - Privacy

BookApp Home Books

## Edit Book

Title:

ISBN:

Publisher:

Author:

Category:

Save

© 2019 - BookApp - Privacy

# Web Pages

Web Page	CRUD Operation
Books	Read
NewBook	Create
EditBook	(Read +) Update
DeleteBook	Delete

CRUD:  
Create  
Read  
Update  
Delete

# Web Pages

- Books
  - Books.cshtml (Razor Page) + Books.cshtml.cs (Page Model)
- NewBook
  - NewBook.cshtml + NewBook.cshtml.cs
- EditBook
  - EditBook.cshtml + EditBook.cshtml.cs
- DeleteBook
  - DeleteBook.cshtml + DeleteBook.cshtml.cs



# Books

# Books

BookApp [Home](#) [Books](#)

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

[New Book](#)



# appSettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",

  "ConnectionStrings": {
    "ConnectionString": "DATA SOURCE=xxx;UID=xxx;PWD=xxx;DATABASE=xxx"
  }
}
```

# Startup.cs

Then we need to add something to the “**Startup.cs**” file:

```
public void ConfigureServices (IServiceCollection services)
{
    services.AddRazorPages ();

    services.AddSingleton<IConfiguration>(Configuration);
}
```

We have added:

```
services.AddSingleton<IConfiguration>(Configuration);
```

Books.cshtml.cs

Page Model File

```
...
using BookApp.Models;

namespace BookApp.Pages
{
    public class BooksModel : PageModel
    {
        readonly IConfiguration _configuration;

        public List<Book> books = new List<Book>();

        string connectionString;

        public BooksModel(IConfiguration configuration)

        {
            _configuration = configuration;
        }

        public void OnGet()

        {
            books = GetBookList();
        }

        private List<Book> GetBookList()

        {
            connectionString = _configuration.GetConnectionString("ConnectionString");
            List<Book> bookList = new List<Book>();
            Book book = new Book();
            bookList = book.GetBooks(connectionString);
            return bookList;
        }
    }
}
```

# Books.cshtml

## Web Page with Razor

```
@page
@model BookApp.Pages.BooksModel
@{
    ViewData["Title"] = "Books";
}

<div>
    <h1>Books</h1>
    Below you see all the Books in the Book Store:<br />

    <table class="table">
        <thead>
            <tr>
                <th>BookId</th>
                <th>Title</th>
                <th>ISBN</th>
                <th>Publisher</th>
                <th>Author</th>
                <th>Category</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var book in Model.books)
            {
                <tr>
                    <td> @book.BookId </td>
                    <td><a href="./EditBook?bookid=@book.BookId">@book.Title</a> </td>
                    <td> @book.Isbn </td>
                    <td> @book.PublisherName </td>
                    <td> @book.AuthorName </td>
                    <td> @book.CategoryName </td>
                    <td><a href="./DeleteBook?bookid=@book.BookId" class="btn btn-danger" role="button">Delete Book</a> </td>
                </tr>
            }
        </tbody>
    </table>
    <a href="./NewBook" class="btn btn-info" role="button">New Book</a>
</div>
```

# Run the Application

<https://www.halvorsen.blog>



# New Book

Hans-Petter Halvorsen

[Table of Contents](#)

# New Book

BookApp Home Books

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

New Book

# New Book

## New Book

Title:

ISBN:

Publisher:

Author:

Category:



# Web Forms

```
<form name="bookForm" id="bookForm" method="post">  
Title:  
<input name="bookTitle" type="text"/>  
  
<input id="saveButton" type="submit" value="Save"/>  
</form>
```

Get Access to the Data inside the Form Field from C# Code:

```
book.Title = Request.Form["bookTitle"];
```

```
public void CreateBook(string connectionString, Book book)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("CreateBook", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add(new SqlParameter("@Title", book.Title));
            cmd.Parameters.Add(new SqlParameter("@Isbn", book.Isbn));
            cmd.Parameters.Add(new SqlParameter("@PublisherName", book.PublisherName));
            cmd.Parameters.Add(new SqlParameter("@AuthorName", book.AuthorName));
            cmd.Parameters.Add(new SqlParameter("@CategoryName", book.CategoryName));

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```



Stored Procedure

**“CreateBook” Method in “Book.cs” Class**

## NewBook.cshtml.cs

## Page Model File

```
using BookApp.Models;
namespace BookApp.Pages
{
    public class NewBookModel : PageModel
    {
        readonly IConfiguration _configuration;
        public string connectionString;
        public NewBookModel(IConfiguration configuration)
        {
            _configuration = configuration;
        }
        public void OnGet()
        {
        }
        public void OnPost()
        {
            Book book = new Book();
            book.Title = Request.Form["bookTitle"];
            book.Isbn = Request.Form["bookIsbn"];
            book.PublisherName = Request.Form["bookPublisher"];
            book.AuthorName = Request.Form["bookAuthor"];
            book.CategoryName = Request.Form["bookCategory"];
            connectionString = _configuration.GetConnectionString("ConnectionString");
            book.CreateBook(connectionString, book);
            Response.Redirect("./Books");
        }
    }
}
```

# NewBook.cshtml

## Web Page with Razor

```
@page
@model BookApp.Pages.NewBookModel

@{
    ViewData["Title"] = "New Book";
}

<div>
    <h1>New Book</h1>
    <form name="bookForm" id="bookForm" method="post">
        Title:
        <br />
        <input name="bookTitle" type="text" class="form-control input-lg" autofocus required />
        <br />

        ISBN:
        <br />
        <input name="bookIsbn" type="text" class="form-control input-lg" required />
        <br />

        Publisher:
        <br />
        <input name="bookPublisher" type="text" class="form-control input-lg" required />
        <br />

        Author:
        <br />
        <input name="bookAuthor" type="text" class="form-control input-lg" required />
        <br />

        Category:
        <br />
        <input name="bookCategory" type="text" class="form-control input-lg" required />
        <br />
        <input id="saveButton" type="submit" value="Save" class="btn btn-info" />
    </form>
</div>
```

# Run the Application

<https://www.halvorsen.blog>



# Edit Book

Hans-Petter Halvorsen

[Table of Contents](#)

# New Book

BookApp Home Books

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

New Book

# Edit Book

BookApp [Home](#) [Books](#)

## Edit Book

Title:



ISBN:

Publisher:

Author:

Category:



# Query String

The Query String is used to retrieve the variable values in the HTTP Query String. The HTTP query string is specified by the values following the question mark (?)

Below you see a Hyperlink defined in the **Books.cshtml** Web Page:

```
<a href=" ./EditBook?bookid=@book.BookId" >@book.Title</a>
```

Get Access to the Data from the Query String from C# Code:

```
bookId = Convert.ToInt16 (Request.Query ["bookid"] );
```

```
public Book GetBookData(string connectionString, int bookId)
{
    SqlConnection con = new SqlConnection(connectionString);

    string selectSQL = "select BookId, Title, Isbn, PublisherName, AuthorName, CategoryName
                        from GetBookData where BookId = " + bookId;

    con.Open();

    SqlCommand cmd = new SqlCommand(selectSQL, con);

    SqlDataReader dr = cmd.ExecuteReader();

    Book book = new Book();

    if (dr != null)
    {
        while (dr.Read())
        {
            book.BookId = Convert.ToInt32(dr["BookId"]);
            book.Title = dr["Title"].ToString();
            book.Isbn = dr["ISBN"].ToString();
            book.PublisherName = dr["PublisherName"].ToString();
            book.AuthorName = dr["AuthorName"].ToString();
            book.CategoryName = dr["CategoryName"].ToString();
        }
    }

    return book;
}
```

## “GetBookData” Method “Book.cs” Class

# “EditBook” Method

```
public void EditBook(string connectionString, Book book)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("UpdateBook", con);
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add(new SqlParameter("@BookId", book.BookId));
            cmd.Parameters.Add(new SqlParameter("@Title", book.Title));
            cmd.Parameters.Add(new SqlParameter("@Isbn", book.Isbn));
            cmd.Parameters.Add(new SqlParameter("@PublisherName", book.PublisherName));
            cmd.Parameters.Add(new SqlParameter("@AuthorName", book.AuthorName));
            cmd.Parameters.Add(new SqlParameter("@CategoryName", book.CategoryName));

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

← Stored Procedure

# EditBook.cshtml.cs

## Page Model File

```
using BookApp.Models;
namespace BookApp.Pages
{
    public class EditBookModel : PageModel
    {
        readonly IConfiguration _configuration;
        public Book bookdb = new Book();
        string connectionString;
        public int bookId;
        public EditBookModel(IConfiguration configuration)
        {
            _configuration = configuration;
        }

        public void OnGet()
        {
            bookId = Convert.ToInt16(Request.Query["bookid"]);
            Book book = new Book();
            connectionString = _configuration.GetConnectionString("ConnectionString");
            bookdb = book.GetBookData(connectionString, bookId);
        }

        public void OnPost()
        {
            Book book = new Book();
            book.BookId = Convert.ToInt16(Request.Form["bookId"]);
            book.Title = Request.Form["bookTitle"];
            book.Isbn = Request.Form["bookIsbn"];
            book.PublisherName = Request.Form["bookPublisher"];
            book.AuthorName = Request.Form["bookAuthor"];
            book.CategoryName = Request.Form["bookCategory"];
            connectionString = _configuration.GetConnectionString("ConnectionString");
            book.EditBook(connectionString, book);
            Response.Redirect("./Books");
        }
    }
}
```

# EditBook.cshtml

## Web Page with Razor

```
@page
@model BookApp.Pages.EditBookModel
@{
    ViewData["Title"] = "Edit Book";
}

<div>
<h1>Edit Book</h1>
<form name="bookForm" id="bookForm" method="post">
    <input name="bookId" type="text" value="@Model.bookdb.BookId" hidden/>

    Title:
    <br />
    <input name="bookTitle" type="text" value="@Model.bookdb.Title" class="form-control
    input-lg" autofocus required />
    <br />

    ISBN:<br />
    <input name="bookIsbn" type="text" value="@Model.bookdb.Isbn" class="form-control
    input-lg" required />
    <br />

    Publisher:<br />
    <input name="bookPublisher" type="text" value="@Model.bookdb.PublisherName" class="form-control input-lg" required />
    <br />

    Author:<br />
    <input name="bookAuthor" type="text" value="@Model.bookdb.AuthorName" class="form-control input-lg" required />
    <br />

    Category:
    <br />
    <input name="bookCategory" type="text" value="@Model.bookdb.CategoryName" class="form-control input-lg" required />
    <br />

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

</form>
</div>
```

# Run the Application



# Delete Book

Hans-Petter Halvorsen

[Table of Contents](#)

# Delete Book

BookApp Home Books

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

New Book



```
public void DeleteBook(string connectionString, int bookId)
{
    try
    {
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("DeleteBook", con); ← Stored Procedure

            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add(new SqlParameter("@BookId", bookId));

            con.Open();

            cmd.ExecuteNonQuery();

            con.Close();
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

“DeleteBook” Method  
in “Book.cs” Class

# DeleteBook.cshtml.cs

## Page Model File

```
using BookApp.Models;
namespace BookApp.Pages
{
    public class DeleteBookModel : PageModel
    {
        readonly IConfiguration _configuration;
        string connectionString;
        public int bookId;

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

        public void OnGet()
        {
            bookId = Convert.ToInt16(Request.Query["bookid"]);
            connectionString = _configuration.GetConnectionString("ConnectionString");
            Book book = new Book();
            book.DeleteBook(connectionString, bookId);
            Response.Redirect("./Books");
        }
    }
}
```

DeleteBook.cshtml

(No Code)

Web Page with Razor

```
@page  
  
@model BookApp.Pages.DeleteBookModel  
  
{  
  
}
```

# Run the Application

# Final Application

BookApp Home Books

## Books

Below you see all the Books in the Book Store:

BookId	Title	ISBN	Publisher	Author	Category	Action
1	<a href="#">Introduction to Linear Algebra</a>	0-07-066781-0	Prentice Hall	Gilbert Strang	Science	Delete Book
2	<a href="#">Modern Control System</a>	1-08-890781-0	Wiley	Dorf Bishop	Programming	Delete Book
3	<a href="#">The Lord of the Rings</a>	2-09-066556-2	McGraw-Hill	J.R.R Tolkien	Novel	Delete Book
4	<a href="#">Python</a>	55555	Halvorsen	Hans-Petter	Programming	Delete Book

New Book

BookApp Home Books

## New Book

Title:

ISBN:

Publisher:

Author:

Category:

Save

© 2019 - BookApp - Priv

© 2019 - BookApp - Privacy

BookApp Home Books

## Edit Book

Title:

ISBN:

Publisher:

Author:

Category:

Save

© 2019 - BookApp - Privacy

CRUD:  
Create  
Read  
Update  
Delete

# 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>

