https://www.halvorsen.blog

ASP.NET Core CRUD Application

Hans-Petter Halvorsen

Contents

Introduction

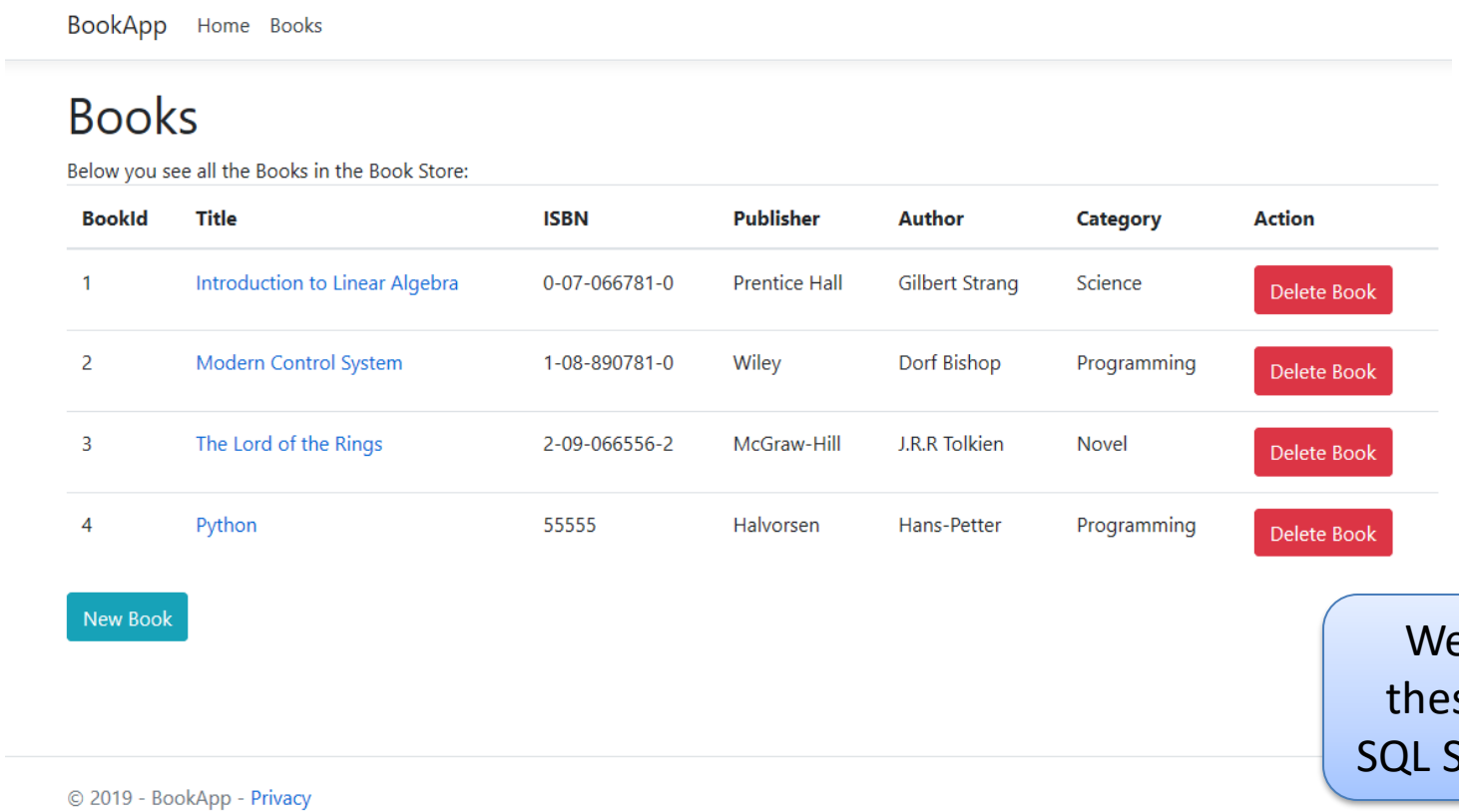
- A basic **CRUD Application** example will be given using **ASP.NET Core**.
- If you have never used ASP.NET Core, I suggest the following Videos:
  - ASP.NET Core - Hello World https://youtu.be/lcQsWYgQXK4
  - ASP.NET Core – Introduction https://youtu.be/zkOtiBcwo8s

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

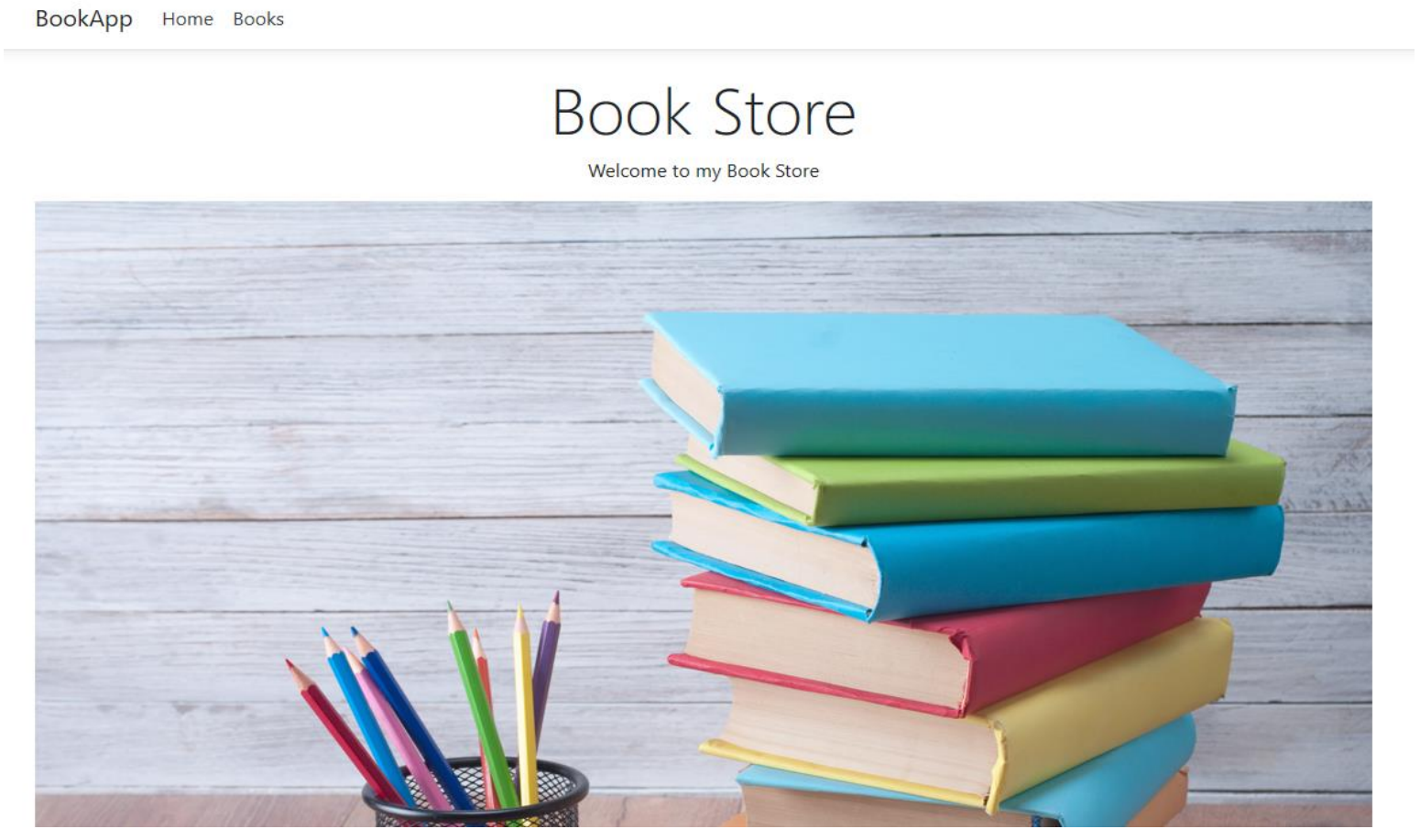
ASP.NET Core Web Application

The following Application will be demonstrated here:



We will retrieve these data from a SQL Server Database

ASP.NET Core Web Application



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

Hans-Petter Halvorsen

Table of Contents

SQL Server

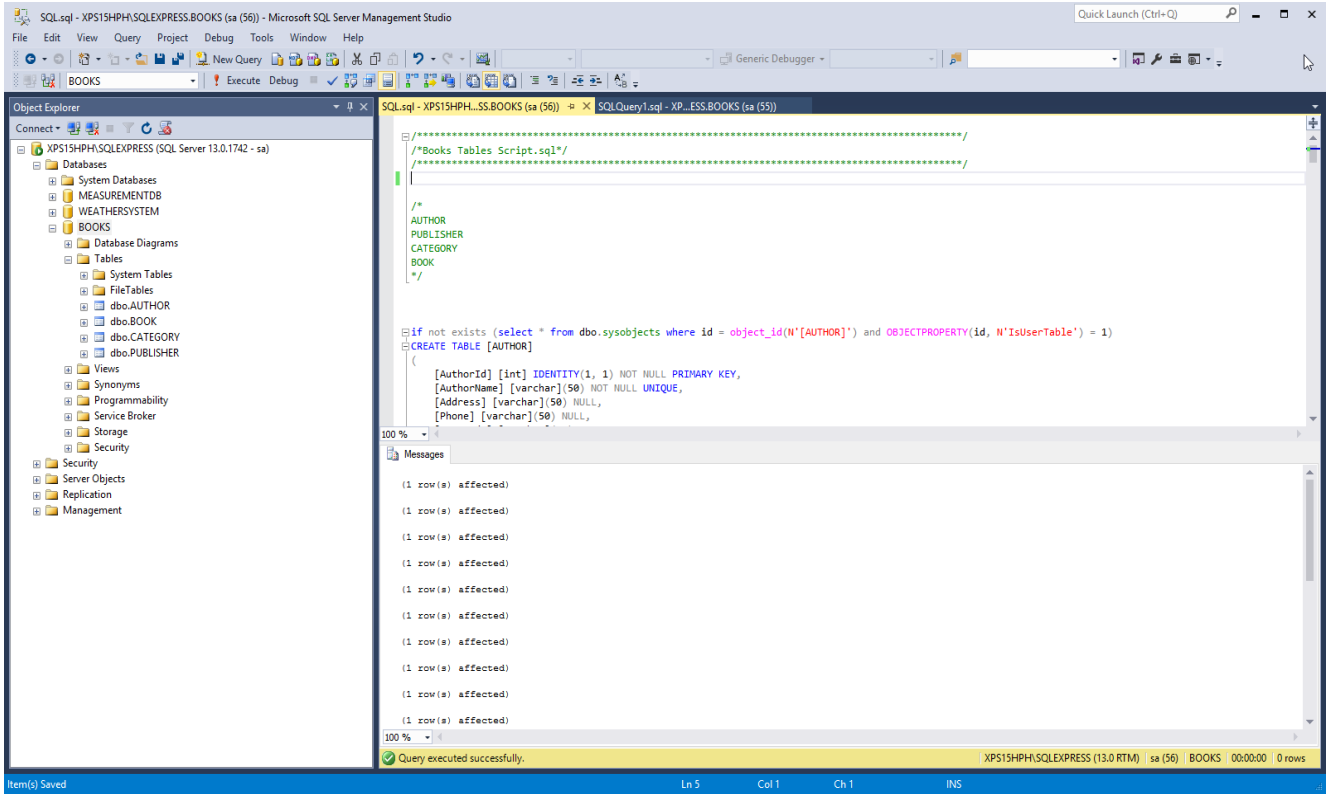
- We will use SQL Server in this example as our database.
- You should have SQL Server locally installed on your computer
- SQL Server Express is recommended.

Database

Hans-Petter Halvorsen

Table of Contents

SQL Server - Create Database



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
    [AuthorName] [varchar](50) NOT NULL
    [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 F
    [AuthorId] [int] NOT NULL FORE
    [CategoryId] [int] NOT NULL FO
    [Description] [varchar](1000)
    [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
```

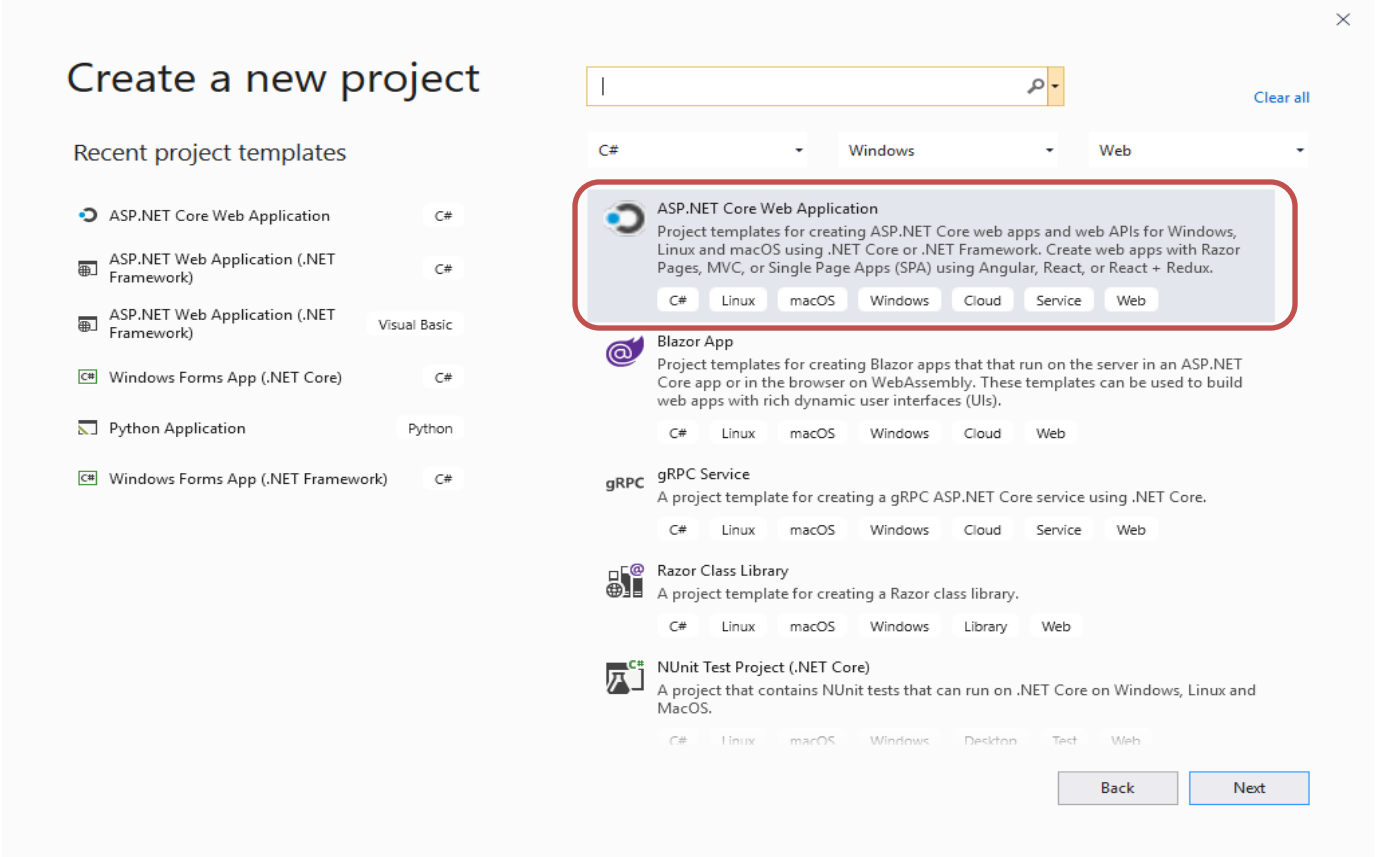
```
--BOOK -------------------------------------
INSERT INTO BOOK (Title, ISBN, PublisherId, AuthorId, CategoryId)
VALUES
(
'Introduction to Linear Algebra',
'0-07-066781-0',
(select PublisherId from PUBLISHER where PublisherName='Prentice Hall'),
(select AuthorId from AUTHOR where AuthorName='Gilbert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

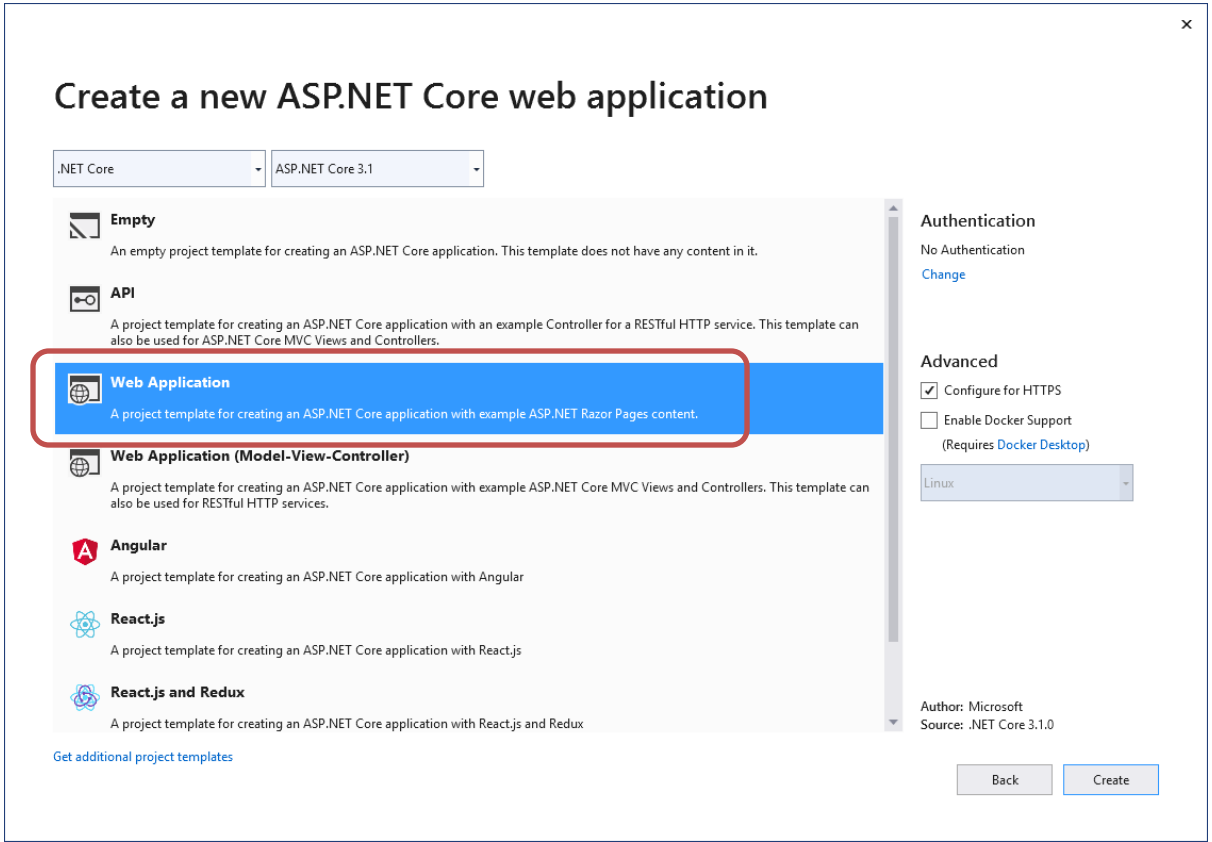
Hans-Petter Halvorsen

Table of Contents

Create New Project

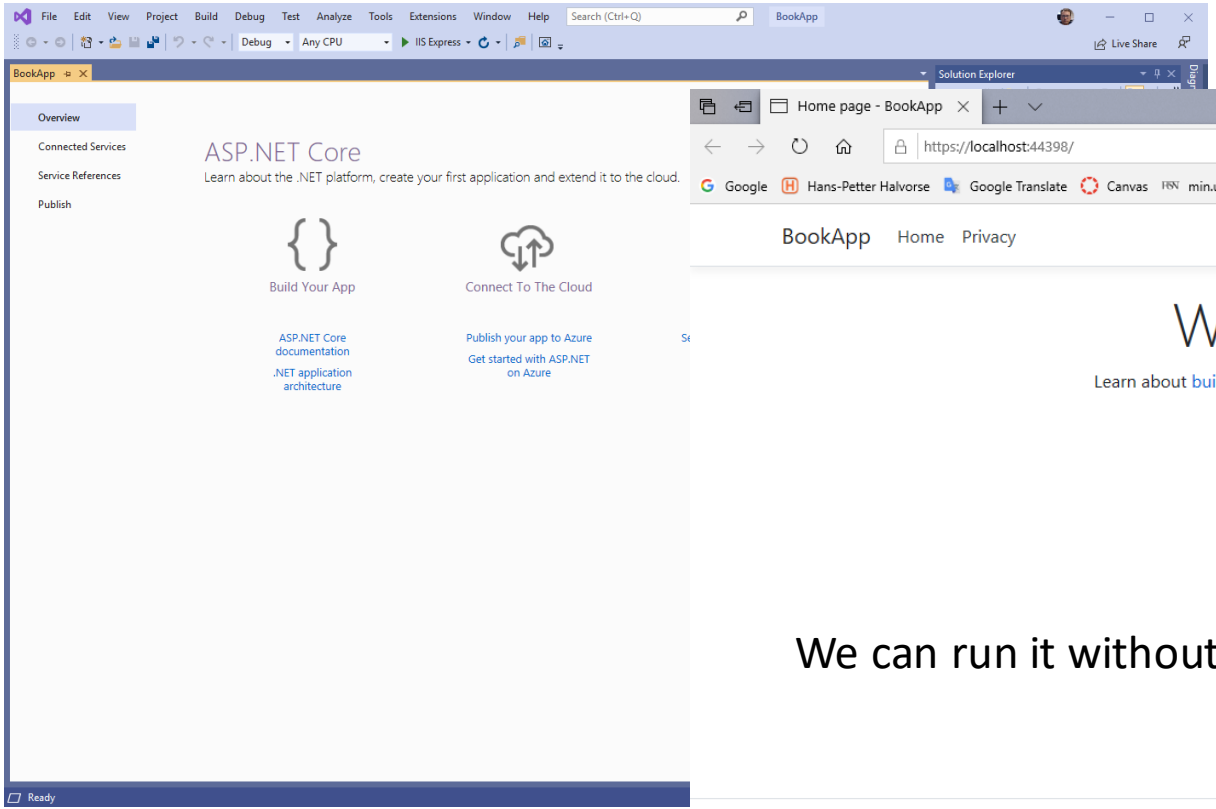


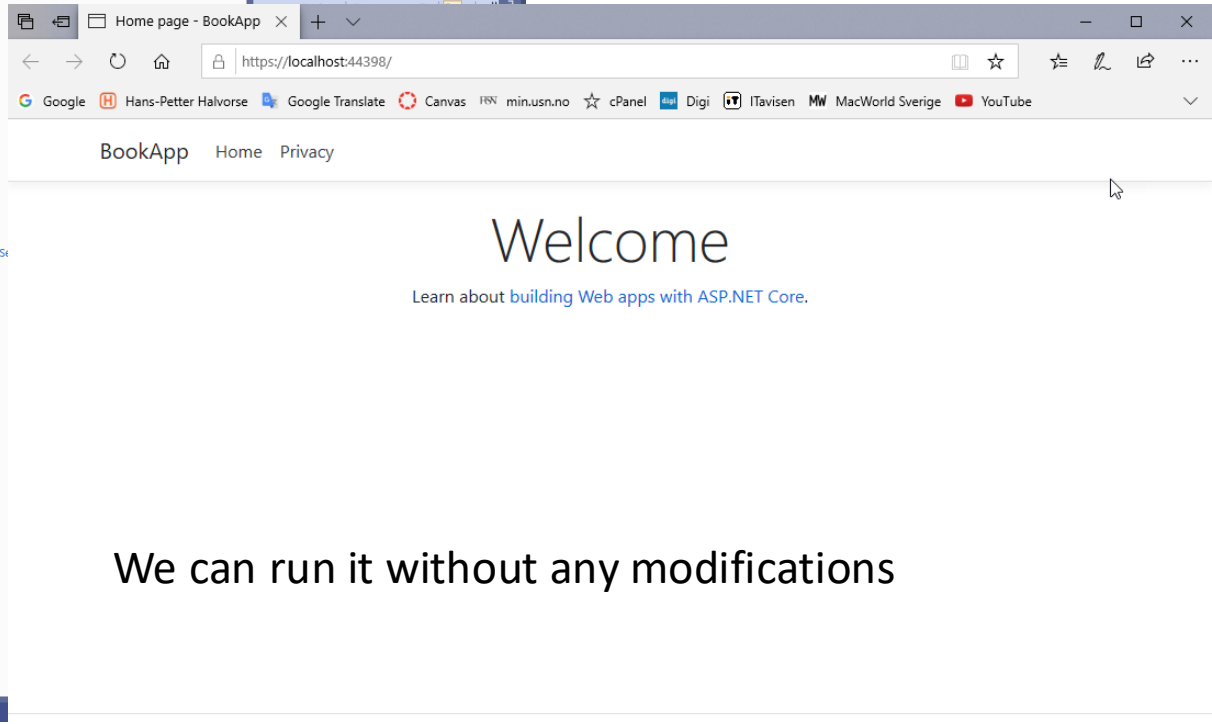
Select ASP.NET Core Web Application



Visual Studio Project

The web application project becomes as shown here





We can run it without any modifications

ADO.NET

Hans-Petter Halvorsen

Table of Contents

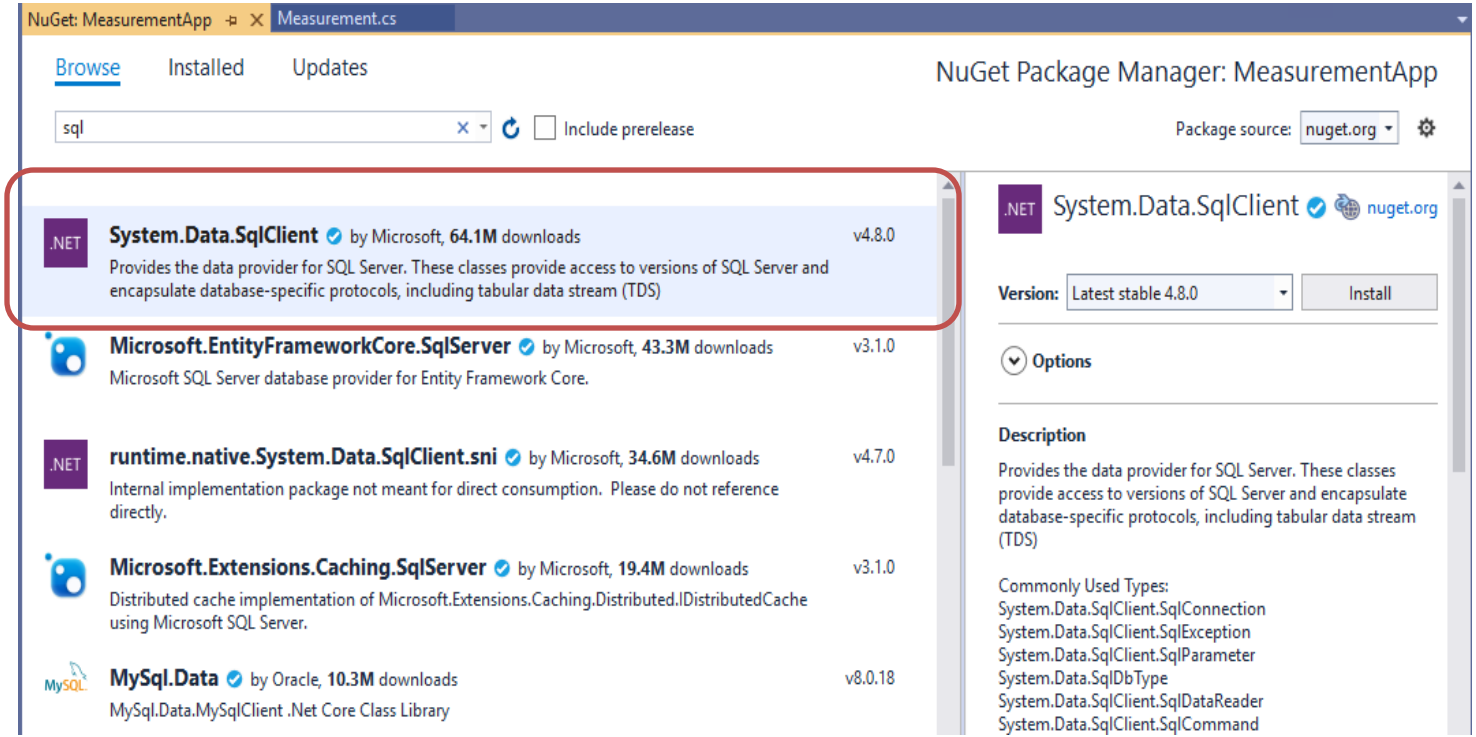
ADO.NET

- 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 is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages.

NuGet

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



Models

Hans-Petter Halvorsen

Table of Contents

Create Models

Books.cs

BookApp | BookApp.Models.Book

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

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

        1 reference
        public List<Book> GetBooks(string connectionString)...

        1 reference
        public Book GetBookData(string connectionString, int bookId)...

        1 reference
        public void CreateBook(string connectionString, Book book)...

        1 reference
        public void EditBook(string connectionString, Book book)...

        1 reference
        public void DeleteBook(string connectionString, int bookId)...

    }
}
```

- 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;
}
```

“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;
    }
}
```

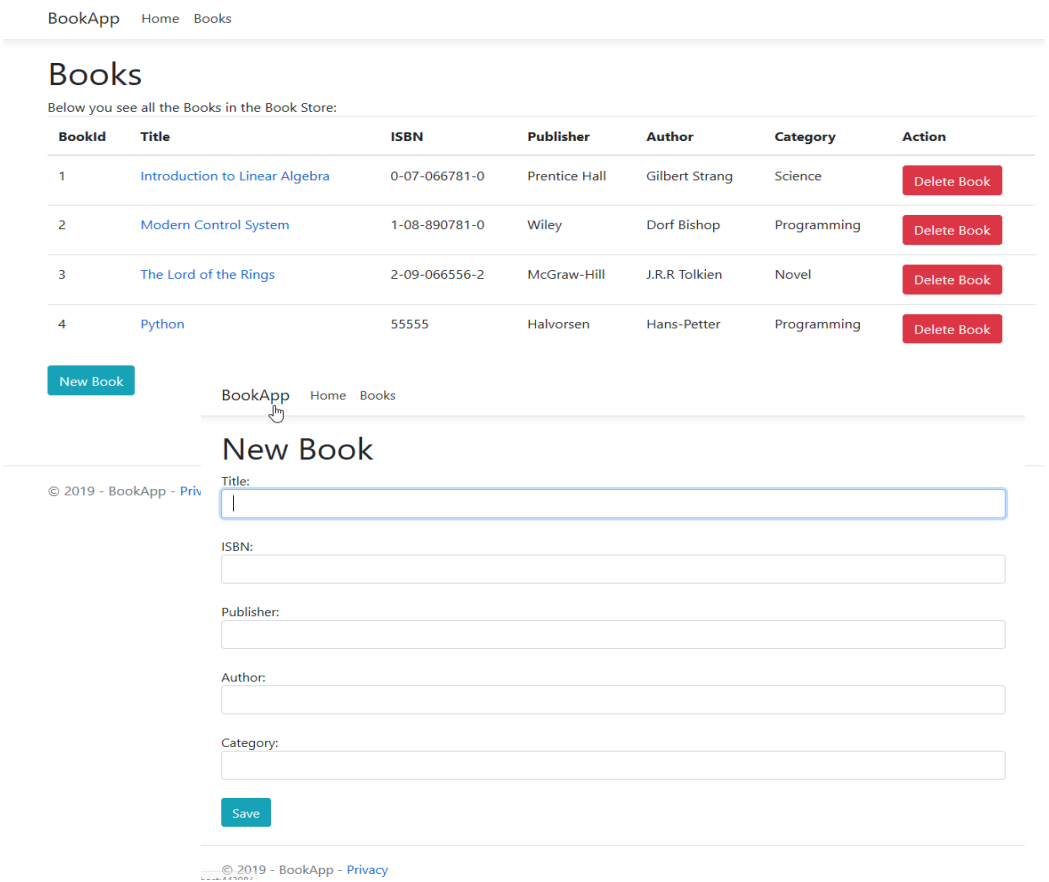
Stored Procedure

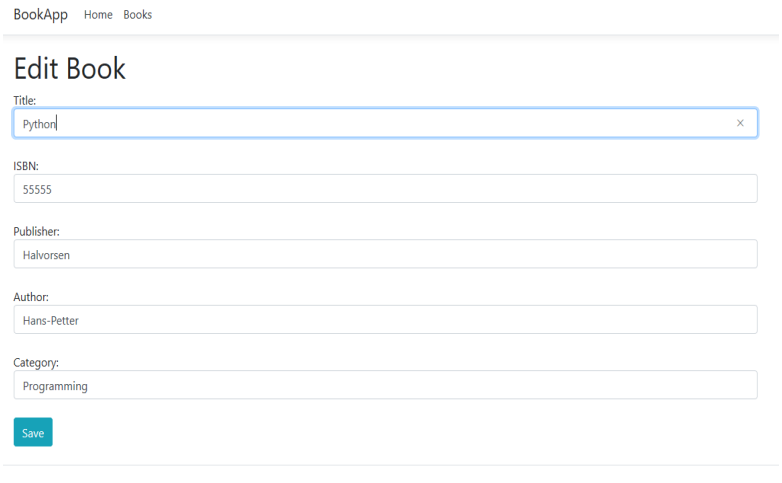
Web Pages

Hans-Petter Halvorsen

Table of Contents

Web Pages


BookApp Home Books
Books
Below you see all the Books in the Book Store:
BookId Title ISBN Publisher Author Category Action
1 Introduction to Linear Algebra 0-07-066781-0 Prentice Hall Gilbert Strang Science Delete Book
2 Modern Control System 1-08-890781-0 Wiley Dorf Bishop Programming Delete Book
3 The Lord of the Rings 2-09-066556-2 McGraw-Hill J.R.R Tolkien Novel Delete Book
4 Python 55555 Halvorsen Hans-Petter Programming Delete Book
New Book
© 2019 - BookApp - Priv
BookApp Home Books
New Book
Title:
ISBN:
Publisher:
Author:
Category:
Save
© 2019 - BookApp - Privacy


BookApp Home Books
Edit Book
Title:
Python
ISBN:
55555
Publisher:
Halvorsen
Author:
Hans-Petter
Category:
Programming
Save
© 2019 - BookApp - Privacy

Web Pages

Web Page	CRUD Operation
Books	**R**ead
NewBook	**C**reate
EditBook	**(R**ead +) **U**pdate
DeleteBook	**D**elete

CRUD:
Create
Read
Update
Delete

Web Pages

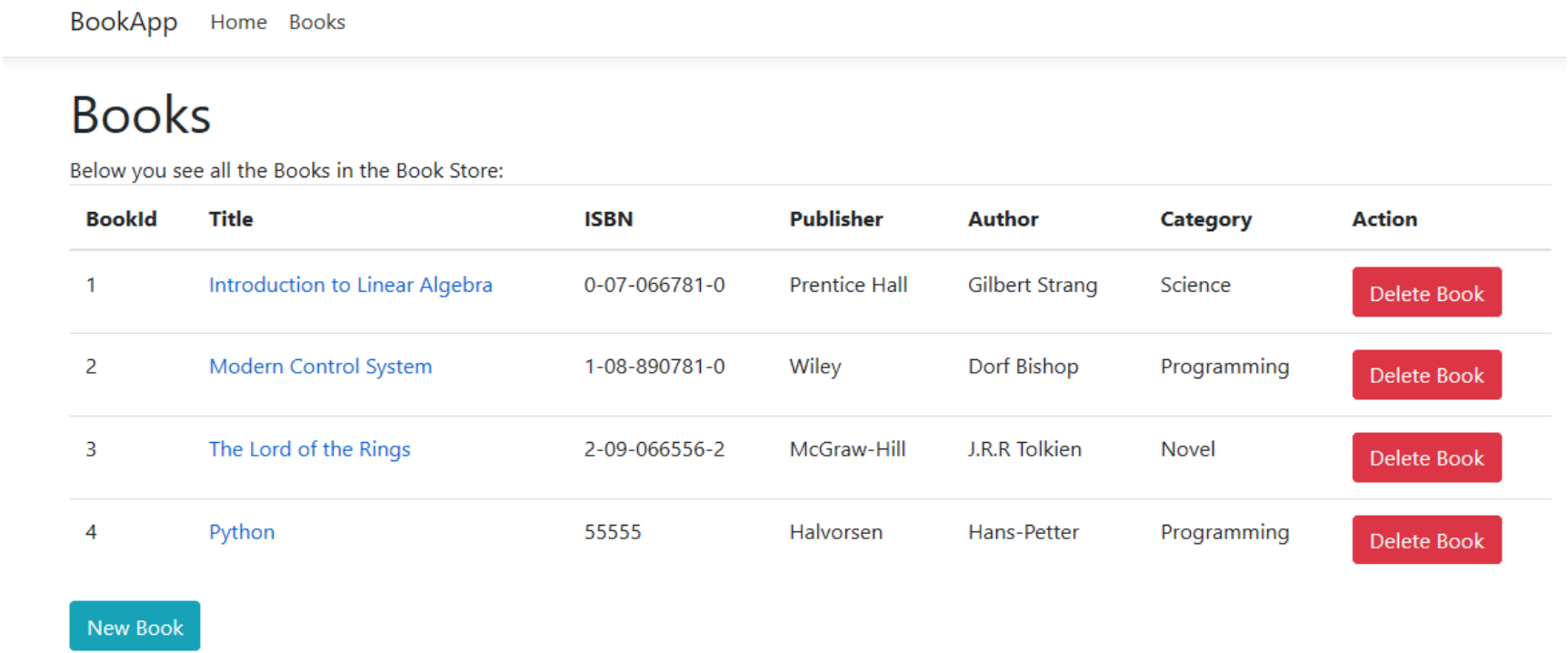
- Books
  - Books.cshtml (Razor Page) + Books.cshtml.cs (Page Model)
- NewBook
  - NewBook.cshtml + NewBook.cshtml.cs
- EditBook
  - EditBook.cshtml + EditBook.cshtml.cs
- DeleteBook
  - DeleteBook.cshtml + DeleteBook.cshtml.cs

Books

Hans-Petter Halvorsen

Table of Contents

Books


BookApp
Home
Books
Books
Below you see all the Books in the Book Store:
BookId
Title
ISBN
Publisher
Author
Category
Action
1
Introduction to Linear Algebra
0-07-066781-0
Prentice Hall
Gilbert Strang
Science
Delete Book
2
Modern Control System
1-08-890781-0
Wiley
Dorf Bishop
Programming
Delete Book
3
The Lord of the Rings
2-09-066556-2
McGraw-Hill
J.R.R Tolkien
Novel
Delete Book
4
Python
55555
Halvorsen
Hans-Petter
Programming
Delete Book
New Book
© 2019 - BookApp - Privacy

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);

```
...

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.cs

Page Model File

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>
```

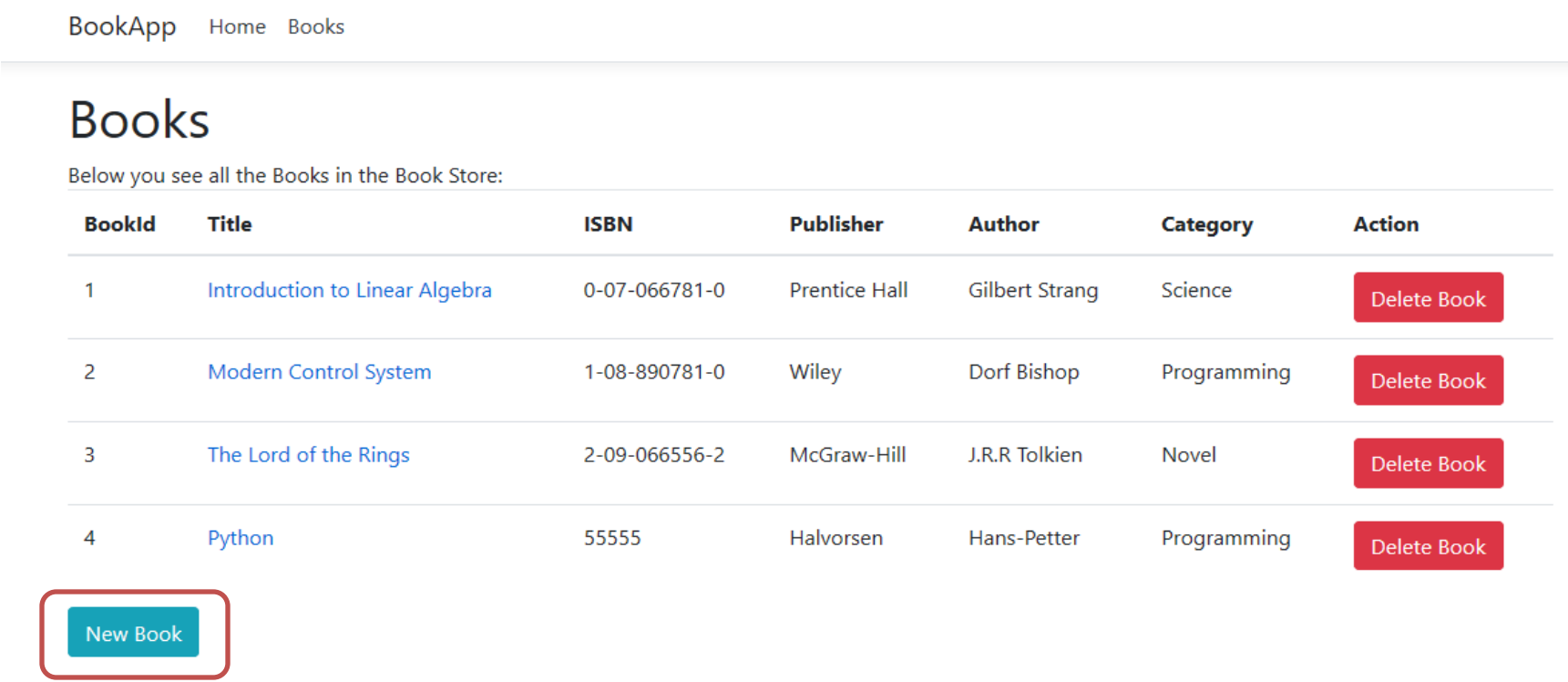
Run the Application

New Book

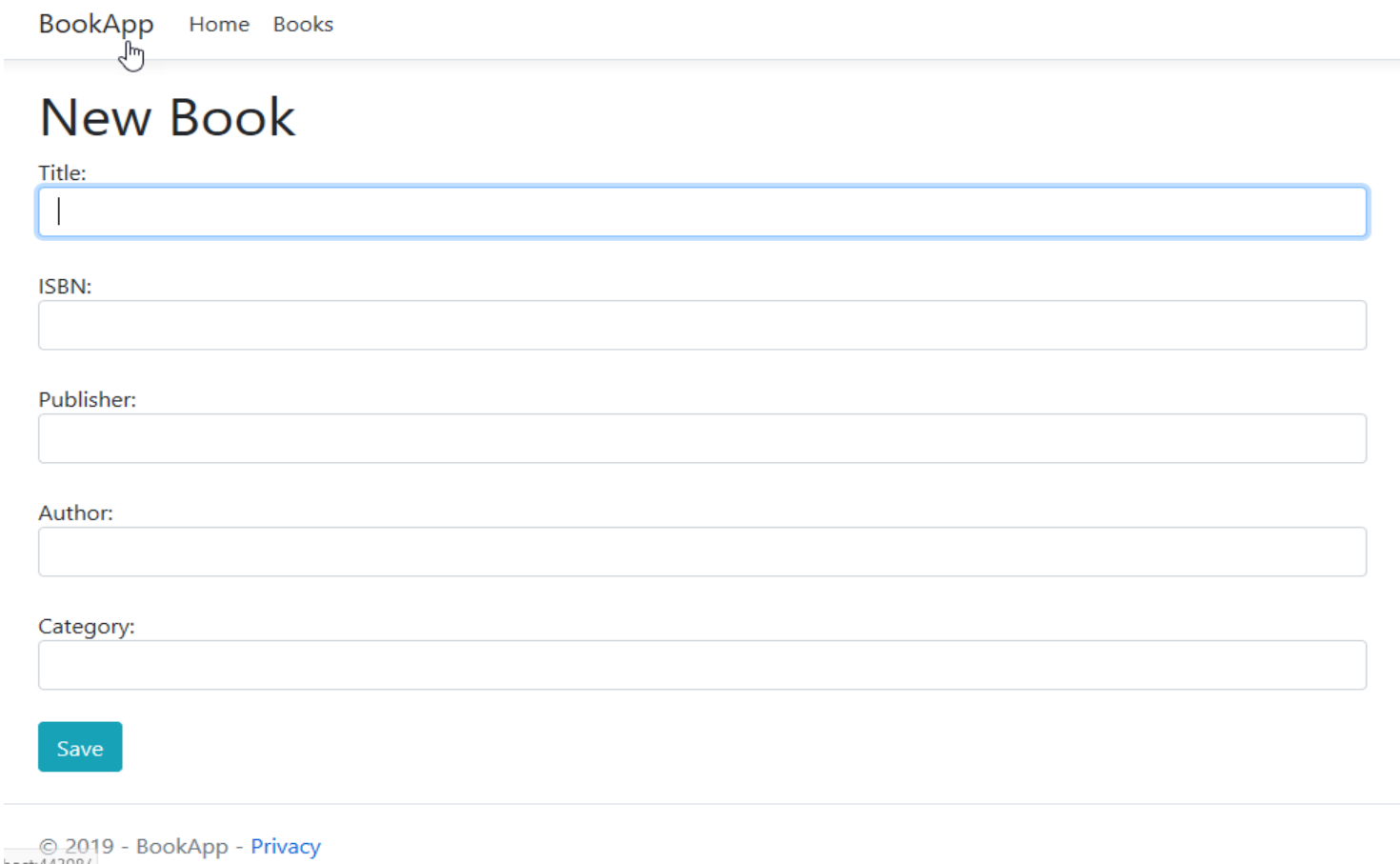
Hans-Petter Halvorsen

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
Introduction to Linear Algebra
0-07-066781-0
Prentice Hall
Gilbert Strang
Science
Delete Book
2
Modern Control System
1-08-890781-0
Wiley
Dorf Bishop
Programming
Delete Book
3
The Lord of the Rings
2-09-066556-2
McGraw-Hill
J.R.R Tolkien
Novel
Delete Book
4
Python
55555
Halvorsen
Hans-Petter
Programming
Delete Book
New Book
© 2019 - BookApp - Privacy

New Book


BookApp
Home
Books
New Book
Title:
ISBN:
Publisher:
Author:
Category:
Save
© 2019 - BookApp - Privacy

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>
```

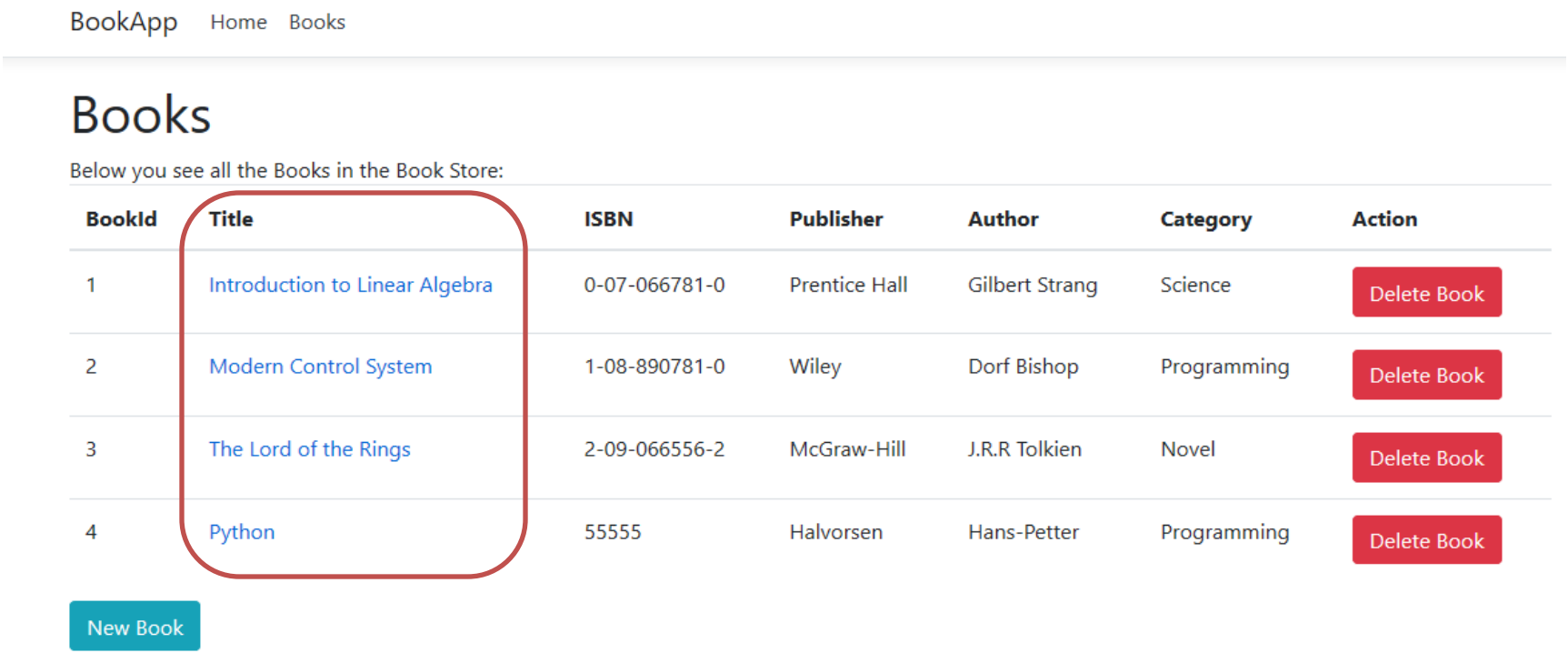
Run the Application

Edit Book

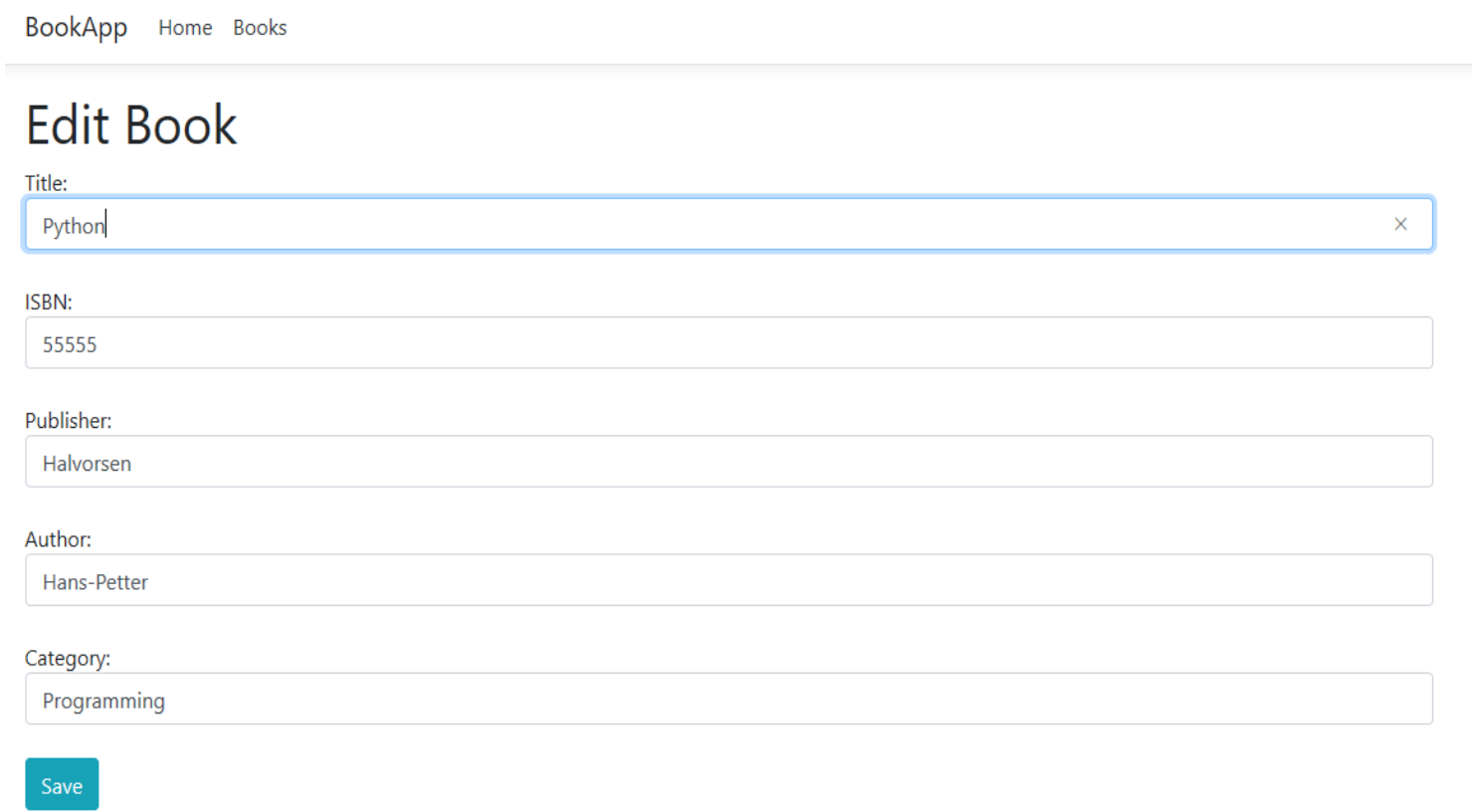
Hans-Petter Halvorsen

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
Introduction to Linear Algebra
0-07-066781-0
Prentice Hall
Gilbert Strang
Science
Delete Book
2
Modern Control System
1-08-890781-0
Wiley
Dorf Bishop
Programming
Delete Book
3
The Lord of the Rings
2-09-066556-2
McGraw-Hill
J.R.R Tolkien
Novel
Delete Book
4
Python
55555
Halvorsen
Hans-Petter
Programming
Delete Book
New Book
© 2019 - BookApp - Privacy

Edit Book


BookApp
Home
Books
Edit Book
Title:
Python
ISBN:
55555
Publisher:
Halvorsen
Author:
Hans-Petter
Category:
Programming
Save
© 2019 - BookApp - Privacy

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>
```

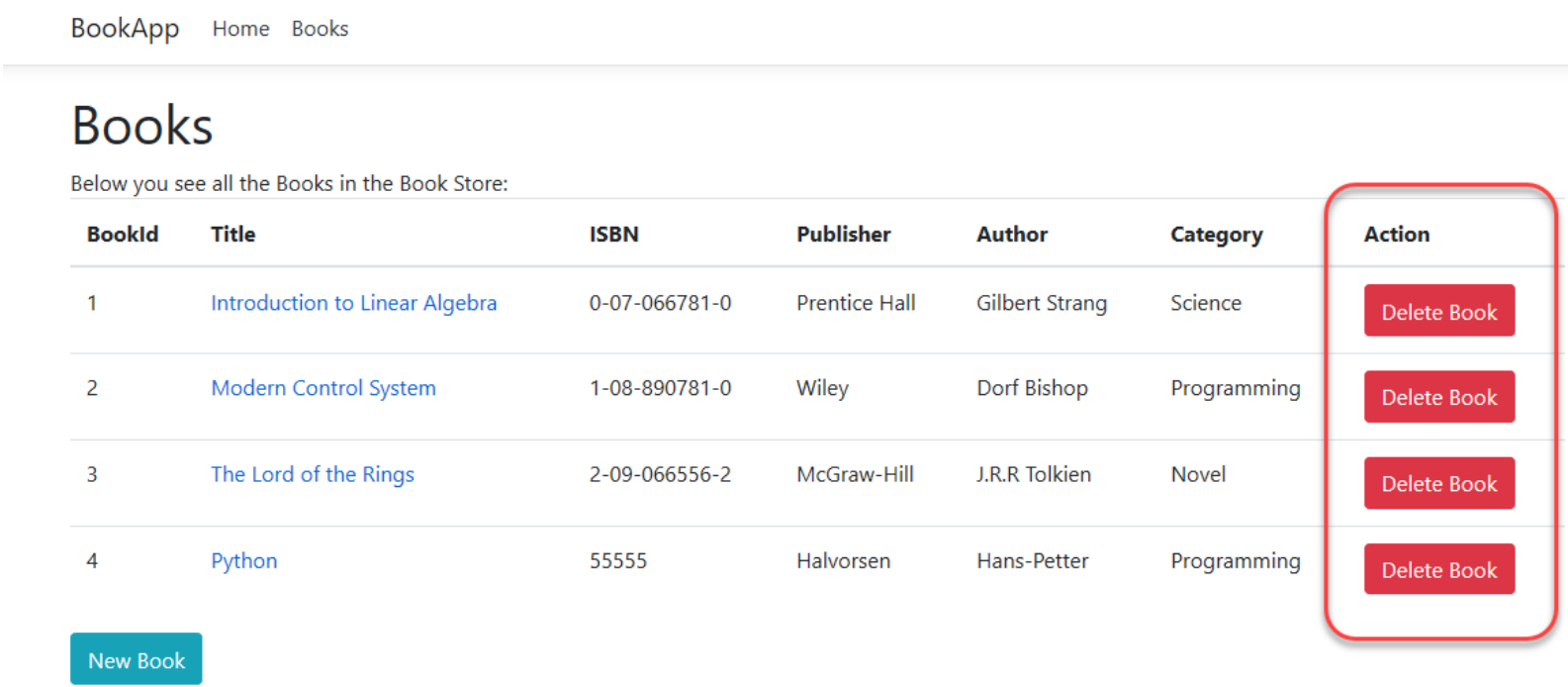
Run the Application

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
Introduction to Linear Algebra
0-07-066781-0
Prentice Hall
Gilbert Strang
Science
Delete Book
2
Modern Control System
1-08-890781-0
Wiley
Dorf Bishop
Programming
Delete Book
3
The Lord of the Rings
2-09-066556-2
McGraw-Hill
J.R.R Tolkien
Novel
Delete Book
4
Python
55555
Halvorsen
Hans-Petter
Programming
Delete Book
New Book
© 2019 - BookApp - Privacy

```
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

"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

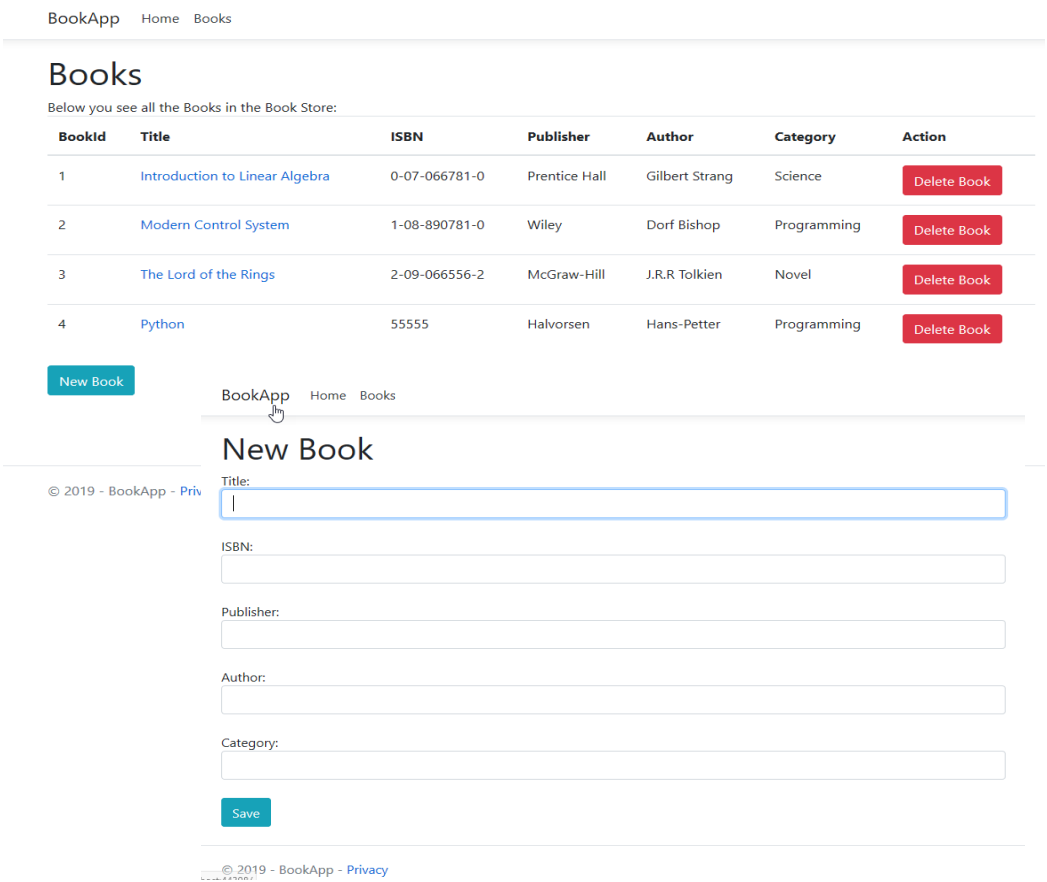
(No Code)

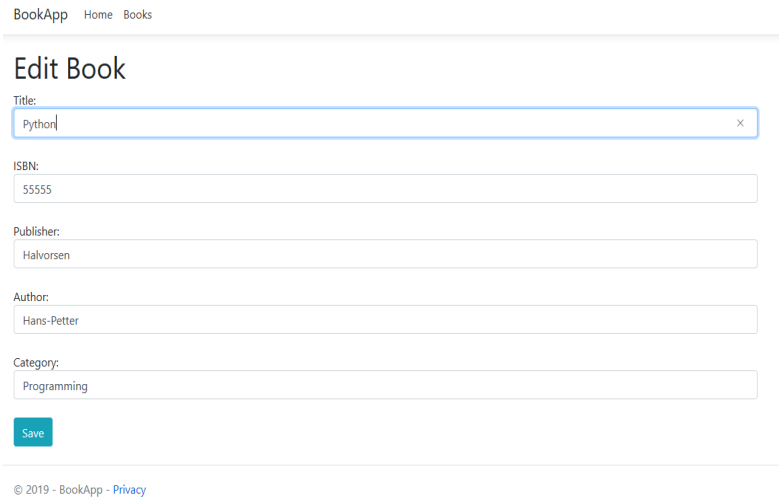
Web Page with Razor

```
@page

@model BookApp.Pages.DeleteBookModel

@{

}
```

Run the Application

Final Application





CRUD:
Create
Read
Update
Delete

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog