CRUD Operation With Razor Pages In ASP.NET Core 2.0

Saineshwar Bageri Updated date Sep 27 2017 80.9k 14 14

DatabaseScript.rar | **RazorPagesDemo.rar**

In this article, we are going to learn how to perform CRUD operations with Razor Pages in ASP.NET CORE 2.0. Before getting started let's get a basic idea of what Razor pages are.

Razor Pages is a new feature of ASP.NET Core MVC that makes coding page-focused scenarios easier and more productive.

Definition taken from here.

Where do we use Razor pages?

There are some pages in the application which are not too big where you are still required to create a controller and add action Method, along with that we need to add View.

In this part we can use Razor Pages which has code behind it . We just need to add a Razor Page and on view “Customer.cshtml” you can design your view and on the same page you can write code for handling requests such as get and Post, but if you think you want to separate it then you can use code behind “Customer.cshtml.cs”

ASP.NET Core

Pre prerequisite for Razor Pages Application

1. Install .Net Core 2.0 SDK
   URL - *https://www.microsoft.com/net/download/core*

   ASP.NET Core

   ASP.NET Core

2. Install Visual Studio 2017 version 15.3 or later
   URL - *https://www.visualstudio.com/downloads/*

Visual Studio Community 2017

Free, fully-featured IDE for students, open-source and individual developers

ASP.NET Core

Creating Razor page project

For creating a project just choose File Menu from Visual studio IDE then New, inside that, choose Project.

ASP.NET Core

After choosing a project, a new dialog will pop up with the name “New Project”. In that, we are going to choose Visual C# Project Templates - Web - ASP.NET Core Web Application. Then, we are going to name the project as “RazorPagesDemo”.

ASP.NET Core

After naming the project we are going click on OK button to create a project.

A new dialog will pop up for choosing templates for Creating “ASP.NET Core Web Application” in that template we are going to Create ASP.Net Core Razor Pages application. That's why we are going to choose “Web Application” and next we will have the option to choose framework 1.Net core or 2.Net Framework, and also ASP.NET Core Version. In that, we are going to choose “.Net Core” and “ASP.NET Core 2.0” as ASP.NET Core Version as click on OK button to create a project.

ASP.NET Core

After clicking on OK button it will start to create a project.

ASP.NET Core

Project Structure

You will only able to see Pages folder in this folder all Razor pages are stored.

ASP.NET Core

One thing which is back again is a little code behind to make thinks little easier.

ASP.NET Core

Yes Razor pages has Code behind.

ASP.NET Core

In the first steps we are going to add a model folder and inside that folder we are going to add Customer Model.

Creating Model folder and Adding Customer Model in it

In this part, we are going to add a Model folder in “RazorPagesDemo” Project after adding folder next we are going to add Customer class (Model) in Models folder.

For details see below snapshot,

ASP.NET Core

After adding Customer Model now next we are going to add a property to this class.

Adding Properties and DataAnnotations to Customer Model

In Razor pages, you can use the same DataAnnotations which are there in MVC.

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace RazorPagesDemo.Models
{
    [Table("CustomerTB")]
    public class Customer
    {
        [Key]
        public int CustomerID { get; set; }
        [Required(ErrorMessage = "Enter Name")]
        public string Name { get; set; }
        [Required(ErrorMessage = "Enter Address")]
        public string Address { get; set; }
        [Required(ErrorMessage = "Enter Country Name")]
        public string Country { get; set; }
        [Required(ErrorMessage = "Enter City Name")]
        public string City { get; set; }
        [Required(ErrorMessage = "Enter City Phoneno")]
        public string Phoneno { get; set; }
    }
}
```

After completing with adding model and DataAnnotations next we are going to add Razor page.

Adding Razor Page to project

For adding Razor page just right click on Pages folder then select Add - inside that select New Item.

ASP.NET Core

After selecting New Item a new dialog will pop up with name “New item” in that we are going to select Razor Page Item and name it as

ASP.NET Core

After clicking on add button below is Pages folder structure in this part you can see “Customer.cshtml” View with “Customer.cshtml.cs” code behind which will have all handlers’ part to handle the request.

ASP.NET Core

Understanding “Customer.cshtml” View

This “Customer.cshtml” file looks more like a Razor View.

In Customer.cshtml view the first thing you are going to see is @page directive which tells Razor view engine that this page is Razor page, not MVC View and it makes a page to handle request directly without going to the controller.

ASP.NET Core

Next, you can see @model is a CustomerModel is a code-behind class name.

ASP.NET Core

The CustomModel file name is the same name as Razor page file “Customer.cshtml.cs” just “.cs” appended at last.

This CustomModel class inherits from PageModel which make this class to handle the request.

Next thing you can see in CustomModel is OnGet Method (handler) which handles get request.

Let’s start will simple example then we are going to start with CRUD operation.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace RazorPagesDemo.Pages
{
    public class CustomerModel : PageModel
    {
        public string WelcomeMessage { get; set; }

        public void OnGet()
        {
            WelcomeMessage = "WelCome to Razor Pages by Saineshwar Bageri";
        }
    }
}
```

In this part, we have simply added a string with name WelcomeMessage and assigned value to it.

Next, on view, we are going to display a message in the following format.

ASP.NET Core

Now Save Application and run.

And to access page just enter Page Name "*Customer*".

URL - *http://localhost:######/Customer*

Wow, we have created our first razor page.

ASP.NET Core

CRUD Operation with Razor Pages

The first thing we are going to do is Create Customer; for doing that we have added Customer Model in the Models folder.

Next, we are going to declare that model in *CustomerModel* class as below.

Code snippet of CustomerModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using RazorPagesDemo.Models;

namespace RazorPagesDemo.Pages
{
    public class CustomerModel : PageModel
    {

        public Customer Customer { get; set; }

        public void OnGet()
        {

        }
    }
}
```

Now let’s Design View.

Adding input Controls on Customer.cshtml View

In this part, we have used New MVC tag helper to create input fields and we have added all model properties on View.

Note

if you want to learn about details of New Tag Helpers visit this link.

https://blogs.msdn.microsoft.com/cdndevs/2015/08/06/a-complete-guide-to-the-mvc-6-tag-helpers/

```
@page
@using RazorPagesDemo.Models @*namespace*@
@model CustomerModel
<script src="~/lib/jquery/dist/jquery.js"></script>
<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>


<div class="container">
    <h3>Add Customer</h3>
    <hr />
    <br />
    <form method="post">
        <div class="row">
            <div class="col-md-4">
                <label asp-for="Customer.Name"></label>
                <input asp-for="Customer.Name" class="form-control" />
                <span class="alert-danger" asp-validation-for="Customer.Name"></span>
            </div>
        </div>
        <div class="row">
        <input asp-for="Customer.Address" class="form-control" />
        <span class="alert-danger" asp-validation-for="Customer.Address"></span>
      </div>
    </div>
    <div class="row">
      <div class="col-md-4">
        <label asp-for="Customer.Country"></label>
        <input asp-for="Customer.Country" class="form-control" />
        <span class="alert-danger" asp-validation-for="Customer.Country"></span>
      </div>
    </div>

    <div class="row">
      <div class="col-md-4">
        <label asp-for="Customer.City"></label>
        <input asp-for="Customer.City" cl class="form-control" />
        <span class="alert-danger" asp-validation-for="Customer.City"></span>
      </div>
    </div>

    <div class="row">
      <div class="col-md-4">
        <label asp-for="Customer.Phoneno"></label>
        <input asp-for="Customer.Phoneno" class="form-control" />
        <span class="alert-danger" asp-validation-for="Customer.Phoneno"></span>
      </div>
    </div>
    <br />
    <input type="submit" value="Save" class="btn btn-primary" />
    <a class="btn btn-default" href="/allcustomer">Cancel</a>
  </form>
</div>
```

Now save Application and run project.

URL - *http://localhost:######/Customer*

Eg. ####### (port number)

Snapshot of Customer page

ASP.NET Core

After entering URL the first request goes to OnGet handler.

While Debugging

ASP.NET Core

Note - Handlers

There are 2 default handlers in Razor pages

1. OnGet()
2. OnPost()

If you want you can create your own handlers you can create I will show in the upcoming example.

After completing with Designing Razor Page file (“Customer.cshtml”) now let’s add another handler to handle post request.

Adding OnPost Handler

In this part, we are going to add Post Handler to get all data which is filled by the user.

We have added OnPost handler but it is not filling data of model what we have posted.

While Debugging

ASP.NET Core

For binding data, we need to add [BindProperty] attribute on the (Customer) property we declare in CustomerModel class.

ASP.NET Core

If we try to Post model again then we are going to get model populated with values. Let's save these values in the database.

Database Part

I have created a database with the name "CustomerDB" and in that, it has "CustomerTB" table.

ASP.NET Core

First thing for saving data in Database we need an ORM we are going to use "Entity framework core"

Installing package for Entity framework core from NuGet

To install the package, just right click on the project (RazorPagesDemo) and then select Manage NuGet package. The below dialog of NuGet Package Manager will pop up.

In the browse tab, type "Microsoft.EntityFrameworkCore.SqlServer" in the search box and just click on Install button.

ASP.NET Core

Adding Connection string and Setting up DbContext

After adding a reference, now add a connection string in appsetting.json file.

ASP.NET Core

Now, let’s add a class with the name DatabaseContext in Models folder.

For adding a model, just right click on Models folder. Then, select Add - Class. An "Add New Item" dialog will pop up with default class selected. Name the class as DatabaseContext and click on Add button

ASP.NET Core

After adding a DatabaseContext class, next, we are going to inherit DbContext class.

After inheriting with DbContext, next we are creating a constructor which takes DbContextOptions as an input parameter and also inherits the base class constructor (: base(options)) [DbContext].

Code snippet of DatabaseContext class

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace RazorPagesDemo.Models
{
    public class DatabaseContext : DbContext
```

```
        {

        }
    }
}
```

Next, we are going to add a new Service in Startup.cs class for injecting dependency.

Now, whenever you use DatabaseContext class, DbContext instance will be injected there.

ASP.NET Core

After adding CustomerTB Model, in our next step, we are going to add DbSet of all models in DatabaseContext class.

Adding DbSet for CustomerTB Model in DatabaseContext class

Now, let's add DbSet for CustomerTB Model in DatabaseContext class, as shown below.

ASP.NET Core

After adding DbSet in DatabaseContext class next we are going to add a constructor in CustomerModel class.

Setting up Dependency injection at CustomerModel class

In this part wherever we use DatabaseContext class, DbContext instance will be injected there.

ASP.NET Core

Now using ("_Context") object we can save data in the database, let's make a change in OnPost handler to save data in the database.

ASP.NET Core

Code snippet of CustomerModel class

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

namespace RazorPagesDemo.Pages
{
    public class CustomerModel : PageModel
    {
        DatabaseContext _Context;
        public CustomerModel(DatabaseContext databasecontext)
        {
            _Context = databasecontext;
        }

        [BindProperty]
        public Customer Customer { get; set; }

        public void OnGet()
        {

        }

        public ActionResult OnPost()
        {
            var customer = Customer;
            if (!ModelState.IsValid)
            {
                return Page(); // return page
            }
```

```
                var result = _Context.Add(customer);
                _Context.SaveChanges(); // Saving Data in database

                return RedirectToPage("AllCustomer");
            }
        }
}
```

Now save Application and run project.

And to access page just enter Page Name “*Customer*”.

URL - *http://localhost:######/Customer* Eg. ####### (port number)

Now on save button data will be saved in the database.

ASP.NET Core

Database Output

ASP.NET Core

Now we have completed adding part let’s add another Razor Page to display all customers (AllCustomer).

Adding All Customer Razor page

In this part, we are going to add another razor page to display All Customer which is saved in the database.

Adding new razor page in same way as we added customer Razor page

ASP.NET Core

After adding AllCustomer.cshtml Razor page next we are going to adding Constructor and OnGet Handler

Adding Constructor and OnGet Handler

In this part onGet Handler we are going to get all customer details from the database and assign those values to CustomerList and this CustomerList on razor page we are going to iterate and display customer details.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using RazorPagesDemo.Models;
using System.Collections.Generic;
using System.Linq;

namespace RazorPagesDemo.Pages
{
    public class AllCustomerModel : PageModel
    {
        DatabaseContext _Context;
        public AllCustomerModel(DatabaseContext databasecontext)
        {
            _Context = databasecontext;
        }

        public List<Customer> CustomerList { get; set; }

        public void OnGet()
        {
            var data = (from customerlist in _Context.CustomerTB
                        select customerlist).ToList();

            CustomerList = data;
        }
    }
}
```

On AllCustomer razor page view we are going to declare @page directive after that namespace and at last

@model which is “*AllCustomerModel*”

ASP.NET Core

Now we are going to Iterate data (CustomerList).

Code snippet of AllCustomerModel.cshtml

```
@page
@using RazorPagesDemo.Models
@model AllCustomerModel

<h2>Index</h2>

<p>
    <a asp-page="Customer">Create New Customer</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayName("Name")
            </th>
            <th>
                @Html.DisplayName("Address")
            </th>
            <th>
                @Html.DisplayName("Country")
            </th>
            <th>
                @Html.DisplayName("City")
            </th>
            </th>
            <th>Edit | Delete</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.CustomerList)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Address)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Country)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.City)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Phoneno)
                </td>
                <td>
                    <a asp-page="./EditCustomer" asp-route-id="@item.CustomerID">Edit</a> |
                    <a asp-page="./AllCustomer" onclick="return confirm('Are you sure you want to delete this item?');" asp-page-handler="Delete" asp-route-id="@item.CustomerID">Delete</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

Edit link

In edit link, we have just assign EditCustomer razor page name to “asp-page” property and “asp-route-id” we have assigned CustomerID to it, We haven't added EditCustomer razor page yet but we will soon.

```
<a asp-page="./EditCustomer" asp-route-id="@item.CustomerID">Edit</a>
```

Delete link

In Delete link we have just assigned AllCustomer razor page names to “asp-page” property and we have assigned CustomerID to “asp-route-id; " next we are going to assign “asp-page-handler” property which is a new one especially for Razor pages. Here we are going to add Handler name “Delete” which we are going to create in AllCustomer Razor page.

```
<a asp-
page="./AllCustomer" onclick="return confirm('Are you sure you want to delete this item?');"
asp-page-handler="Delete" asp-route-id="@item.CustomerID">Delete</a>
```

Adding Delete Handler

In Razor pages, we have 2 default handlers, OnGet and OnPost but we can add custom handlers also to handle OnGet and OnPost request

Below is a snapshot of it.

ASP.NET Core

Code snippet of Delete Handler

In this part, we are going add Delete handler “*OnGetDelete*” which take CustomerID as input.

remove method to delete it and finally we redirect after deleting to the AllCustomer page.

```
public ActionResult OnGetDelete(int? id)
        {
            if (id != null)
            {
                var data = (from customer in _Context.CustomerTB
                            where customer.CustomerID == id
                            select customer).SingleOrDefault();

                _Context.Remove(data);
                _Context.SaveChanges();
            }

            return RedirectToPage("AllCustomer");
        }
```

Now save the application and run it.

Access AllCustomer page.

ASP.NET Core

Now if you see delete link by hovering it you can see we are passing id (CustomerId) and along with that we are also passing handler name (“*Delete*”)

e.g. *http://localhost:49989/AllCustomer?id=2&handler=Delete*

While debugging

ASP.NET Core

EditCustomer Razor page.

Adding EditCustomer page

In this part, we are going to add another Razor page to Edit Customer Details which is saved in the database.

Adding new Razor page in the same way as we add customer razor page

ASP.NET Core

After adding EditCustomer.cshtml Razor page next we are going to adding Constructor and OnGet Handler

Adding Constructor and OnGet Handler

In this part onGet Handler, we are going to get CustomerID (id) from query string from that we are going to get Customer Details from the database and assign those values to Customer Model this model we are going to send to Edit Customer View.

Code snippet of Edit Customer Model

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using RazorPagesDemo.Models;

namespace RazorPagesDemo.Pages
{
    public class EditCustomerModel : PageModel
    {
        DatabaseContext _Context;
        public EditCustomerModel(DatabaseContext databasecontext)
        {
```

```


        [BindProperty]
        public Customer Customer { get; set; }

        public void OnGet(int? id)
        {
            if (id != null)
            {
                var data = (from customer in _Context.CustomerTB
                            where customer.CustomerID == id
                            select customer).SingleOrDefault();

                Customer = data;
            }
        }
    }
}
```

Next, we are going to design Edit View for displaying records for editing.

Code snippet of Edit Customer Page

In this page we are creating edit page means we are going to show record and allow the user to edit details and update it.

The first thing you are going to see is @page"{id: int}" directive, which tells that page need "{id: int}"int id (CustomerID) then only it will accept requests else it will returns an HTTP 404 (not found) error.

It id (CustomerID) we are going to send from All Customer View Edit link.

Note

Link example: - http://localhost:49989/EditCustomer/1

```
@page "{id:int}"
@using RazorPagesDemo.Models
@model EditCustomerModel

<div class="container">
    <h3>Edit Customer</h3>
    <hr />
    <br />
    <form method="post">
        <input asp-for="Customer.CustomerID" type="hidden" />
        <div class="row">
            <div class="col-md-4">
                <label asp-for="Customer.Name"></label>
                <input asp-for="Customer.Name" class="form-control" />
            </div>
        </div>
        <div class="row">
            <div class="col-md-4">
                <label asp-for="Customer.Address"></label>
                <input asp-for="Customer.Address" class="form-control" />
            </div>
        </div>
        <div class="row">
            <div class="col-md-4">
                <label asp-for="Customer.Country"></label>
                <input asp-for="Customer.Country" class="form-control" />
            </div>
        </div>

        <div class="row">
            <div class="col-md-4">
                <label asp-for="Customer.City"></label>
                <input asp-for="Customer.City" cl class="form-control" />
        <div class="row">
            <div class="col-md-4">
                <label asp-for="Customer.Phoneno"></label>
                <input asp-for="Customer.Phoneno" class="form-control" />
            </div>
        </div>
        <br />
        <input type="submit" value="Save" class="btn btn-primary" />
        <a class="btn btn-default" href="/allcustomer">Cancel</a>
    </form>
</div>
```

Now save the application and run.

Access All Customer page

ASP.NET Core

Now hover edit link you can see URL which is generated (*http://localhost:49989/EditCustomer/1*) now click on Edit link it will display EditCustomer Page.

While debugging

ASP.NET Core

ASP.NET Core

Now we have completed Edit page OnGet Handler implementation which helps to display customer data in edit mode, next we need to update data; for doing that we need to Add Onpost Handler which will handle post request.

Code snippet of OnPost Request

are going to return Page() which will show Error Message.

If it is valid then we are going to update data in the database and redirect page to AllCustomer Page.

Code snippet of Complete EditCustomerModel

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

using RazorPagesDemo.Models;
using System.Linq;

namespace RazorPagesDemo.Pages
{
    public class EditCustomerModel : PageModel
    {
        DatabaseContext _Context;
        public EditCustomerModel(DatabaseContext databasecontext)
        {
            _Context = databasecontext;
        }


        [BindProperty]
        public Customer Customer { get; set; }

        public void OnGet(int? id)
        {
            if (id != null)
            {
                var data = (from customer in _Context.CustomerTB
                            where customer.CustomerID == id
                            select customer).SingleOrDefault();

                Customer = data;
            }

        public ActionResult OnPost()
        {
            var customer = Customer;
            if (!ModelState.IsValid)
            {
                return Page();
            }

            _Context.Entry(customer).Property(x => x.Name).IsModified = true;
            _Context.Entry(customer).Property(x => x.Phoneno).IsModified = true;
            _Context.Entry(customer).Property(x => x.Address).IsModified = true;
            _Context.Entry(customer).Property(x => x.City).IsModified = true;
            _Context.Entry(customer).Property(x => x.Country).IsModified = true;
            _Context.SaveChanges();
            return RedirectToPage("AllCustomer");
        }
    }
}
```

Snapshot of Edit Customer Page

ASP.NET Core

We have updated Name of Customer. Now let's view All customer view -- that value is updated.

ASP.NET Core

Wow, we have successfully Updated Customer Name in the database.

While debugging

ASP.NET Core

In Razor page validation is similar to MVC validation and there is no change in it. Just add DataAnnotations attribute on Model properties and on view starts its validation control.

ASP.NET Core
ASP.NET Core

Finally, we have learned, about Razor Pages and along with that how to do CRUD operations with Razor Pages in a step by step way. I hope you liked my article.

Next Recommended Article
CRUD Operation In ASP.NET Core 2.0 Using Dapper ORM

ASP.NET CORE | ASP.NET CORE 2.0 | CRUD Operation | Razor Pages in ASP.NET CORE 2.0

Saineshwar Bageri TOP 100 · 70 · 17.6m · 5 · MVP 1

Indian Software Developer and Microsoft MVP from C# Corner MVP working on .Net Web Technology (Asp.net , C# , Sqlserver , MVC , Windows ,Console Application, javascript , jquery , json , ORM Dapper) and also... Read more

http://dotnet-sai.blogspot.in

14 14 View Previous Comments

Type your comment here and press Enter Key (Minimum 10 characters)

robinson loreto
•1879 •13 •0
Nov 05, 2019
0 0 Reply

After creating a new customer how woild I redirect to the details page of that customer rather than go back to the list?
Michael Collazo
•1885 •7 •0
Aug 22, 2019
0 0 Reply

How can I bind country lookup table to a dropdown and store integer value?
kalpna
•1874 •18 •0
Sep 17, 2018
0 0 Reply

Very Nice. I was start my first project after 8 yrs of career break and this helped me a lot. Thank you for step by step instructions.
kalpna
•1874 •18 •0
Sep 17, 2018
0 0 Reply

You Nailed it, bro. Awesome very well explained very helpfully. Thank you So much, bro
Nasir Ullah
•1887 •5 •0
Sep 10, 2018
2 0 Reply

Very good article for beginners
Parveen Kumar
•1762 •130 •0
Aug 29, 2018
2 0 Reply

I start my first core project with you, thanks man!
Kawssar Chowdhury
•1883 •9 •0
Aug 17, 2018
2 0 Reply

DropDownList Razor Page?
بارآسا بارآسا
•1889 •3 •0
Jan 26, 2018
1 0 Reply

Saineshwar Bageri! if we have More than one buttons on page then how we can handle functionality of different buttons onPost Handler in code

1824 68 0

1 0 Reply

Good Articale..........

Anilananda Chakrabortty

Oct 09, 2017

1500 392 34.7k

1 0 Reply