CRUD Operation In ASP.NET Core 2.0 Using Dapper ORM

Shrimant Telgave Updated date Aug 20 2018 89.2k 13 16

GroupMeetingASP.NETCoreDapperWebApp.rar

In this article, we will learn CRUD operation in ASP.NET Core 2.0 using Dapper ORM, step by step.

We will use Visual Studio 2017 to develop the web application using ASP.NET Core 2.0 with Razor pages using Dapper. If you don't have a basic idea of ASP.NET Core and how to set up ASP.NET Core Environment, then before proceeding further, please refer to my previous articles on ASP.NET Core 2.0 for a better understanding, as mentioned below.

- What ASP.NET Core Is And Advantages Of Using It
- CRUD Operation In ASP.NET Core 2.0 With Razor Pages Using ADO.NET

Prerequisites

- Install .NET Core 2.0.0 or above SDK from the step to install .NET Core 2.0.
- Install Visual Studio 2017 for the step to install Visual Studio 2017.
- SQL Server 2008 or above.

Dapper is a simple object mapper for .NET and owns the title of "King of Micro ORM" in terms of speed. An ORM is an Object Relational Mapper, which is responsible for mapping between database and programming language.

There are three steps to working with Dapper as follows,

- Create an IDbConnection object.
- Write a query to perform CRUD operations.
- Pass query as a parameter in Execute method.

More detail - dapper-tutorial

We will create one small "Group Meeting" web application using ASP.NET Core 2.0. using Dapper ORM as follow.

First of all, create database scripts.

Scripts 1

To create the database.

```
01.  CREATE DATABASE ProjectMeeting
```

Scripts 2

To create the database table named as “GroupMeeting”.

```
01. USE [ProjectMeeting]
02. GO
03.
04. /****** Object:  Table [dbo].[GroupMeeting]    Script Date: 18-08-2018 23:02:42 ******/
05. SET ANSI_NULLS ON
06. GO
```

```
GO

SET ANSI_PADDING ON
GO

CREATE TABLE [dbo].[GroupMeeting](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ProjectName] [varchar](50) NULL,
    [GroupMeetingLeadName] [varchar](50) NULL,
    [TeamLeadName] [varchar](50) NULL,
    [Description] [varchar](50) NULL,
    [GroupMeetingDate] [date] NULL,
 CONSTRAINT [PK_GroupMeeting-2] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, A
) ON [PRIMARY]

GO

SET ANSI_PADDING OFF
GO
```

Scripts 3

Create the stored procedure to get all the group meeting details.

```
Create procedure [dbo].[GetGroupMeetingDetails]
AS
BEGIN
     SELECT * FROM GROUPMEETING
END
```

Create the stored procedure to get the group meeting by Id.

```
Create procedure [dbo].[GetGroupMeetingByID](@Id int)
AS
BEGIN
     SELECT * FROM GROUPMEETING where id=@Id
END
```

Scripts 5

Create the stored procedure to create a new group meeting,

```
create procedure [dbo].[InsertGroupMeeting]
(
    @ProjectName varchar(50),
    @GroupMeetingLeadName varchar(50),
    @TeamLeadName varchar(50),
    @Description varchar(50),
    @GroupMeetingDate date
)
As
BEGIN

 INSERT INTO GroupMeeting(ProjectName,GroupMeetingLeadName,TeamLeadName,Description,GroupMeetingDat
 VALUES(@ProjectName,@GroupMeetingLeadName,@TeamLeadName,@Description,@GroupMeetingDate)

END
```

Scripts 6

Create the stored procedure to update the group meeting.

```
    @Id int,
    @ProjectName varchar(50),
    @GroupMeetingLeadName varchar(50),
    @TeamLeadName varchar(50),
    @Description varchar(50),
    @GroupMeetingDate date
)
As
BEGIN
     UPDATE GroupMeeting
     SET ProjectName =@ProjectName,
     GroupMeetingLeadName =@GroupMeetingLeadName,
     TeamLeadName = @TeamLeadName,
     Description = @Description,
     GroupMeetingDate =@GroupMeetingDate
     Where Id=@Id
END
```

Scripts 7

Create the stored procedure to delete the group meeting.

```
create procedure [dbo].[DeleteGroupMeeting]
(
    @Id int
)
As
BEGIN
    DELETE FROM GroupMeeting WHERE Id=@Id
END
```

To Insert dummy records into the database table execute the following scripts.

```
USE [ProjectMeeting]
```

```
INSERT INTO [dbo].[GroupMeeting]
           ([ProjectName]
           ,[GroupMeetingLeadName]
           ,[TeamLeadName]
           ,[Description]
           ,[GroupMeetingDate])
     VALUES
           ('Online Laptop booking',
           'Madhav S',
           'Kishor D',
           'Online Laptop booking Software',
           GETDATE()),
          ('Internal Interprice Commnumication',
           'Abhijit L',
           'Dnyanesh D',
           'Interprice Commnumication',
           GETDATE()),
           ('Health Care Management',
           'Jon M',
           'Randy L',
           'Health Care management project',
           GETDATE())
GO
```

Now, we have completed our database related changes. Let's we can go with code base changes using Visual Studio 2017.

Step 1

Open the Visual Studio 2017.

Step 2

Step 3

Select .NET Core from the left side and select the 'ASP.NET Core Web Application' from the new open project template. Then provide the meaning name like "GroupMeetingASP.NETCoreWebApp".

Step 4

Select Web Application(MVC) template from the list of templates and click on "OK" button as follow.

Step 5

The default ASP.NET Core MVC structure gets created as follow. The default model, view, controller gets created by default.

Step 6

Right click on Models => Click on Add => Click on Class.

Provide a meaningful name like “GroupMeeting” and click on "Add" button as follow.

To work with dapper we need to install dapper ORM using 'Manage Nuget Package'

1. Right click on Solution Manager => Click on 'Manage Nuget Package' as shown in the figure.

2. Select 'Browse' and type dapper in the search box => Enter =>Select the dapper and Click on Install as shown in the figure.

After successful installation of dapper ORM:

In order to work with Dapper ORM we need to import a namespace as follows.

```
01. using Dapper;
```

Step 8

Write code to create properties for group meeting class as follow. Also, use the Required attribute to validate the class fields.

```
01. using System;
02. using System.Collections.Generic;
03. using System.Linq;
04. using System.Data;
```

```
07. using Dapper;
08.
09. namespace GroupMeetingASP.NETCoreWebApp.Models
10. {
11.     public class GroupMeeting
12.     {
13.         public int Id { get; set; }
14.
15.         [Required(ErrorMessage ="Enter Project Name!")]
16.         public string ProjectName { get; set; }
17.
18.         [Required(ErrorMessage = "Enter Group Lead Name!")]
19.         public string GroupMeetingLeadName { get; set; }
20.
21.         [Required(ErrorMessage = "Enter Team Lead Name!")]
22.         public string TeamLeadName { get; set; }
23.
24.         [Required(ErrorMessage = "Enter Description!")]
25.         public string Description { get; set; }
26.
27.         [Required(ErrorMessage = "Enter Group Meeting Date!")]
28.         public DateTime GroupMeetingDate { get; set; }
29.
30.         static string strConnectionString = "User Id=sa;Password=Shri;Server=DESKTOP-
    2D0R2UP\\SQL2014;Database=ProjectMeeting;";
31.
32. }
```

Step 9

Write code to get all group meeting detail from the database using the stored procedure with dapper ORM. The method name is "GetGroupMeetings()" and return type is IEnumerable<GroupMeeting> or List<GroupMeeting> you can use either one of them.

```
01. public static IEnumerable<GroupMeeting> GetGroupMeetings()
02.         {
```

```
            using (IDbConnection con = new SqlConnection(strConnectionString))
            {
                if (con.State == ConnectionState.Closed)
                    con.Open();

                groupMeetingsList = con.Query<GroupMeeting>("GetGroupMeetingDetails").ToList();
            }

            return groupMeetingsList;
        }
```

Get group meeting details by groupId code as follow,

```
public static GroupMeeting GetGroupMeetingById(int? id)
        {
            GroupMeeting groupMeeting = new GroupMeeting();
            if (id == null)
                return groupMeeting;

            using (IDbConnection con = new SqlConnection(strConnectionString))
            {
                if (con.State == ConnectionState.Closed)
                    con.Open();

                DynamicParameters parameter = new DynamicParameters();
                parameter.Add("@Id", id);
                groupMeeting = con.Query<GroupMeeting>
("GetGroupMeetingByID", parameter, commandType:CommandType.StoredProcedure).FirstOrDefault();
            }

            return groupMeeting;
        }
```

Step 10

```
01.  public static int AddGroupMeeting(GroupMeeting groupMeeting)
02.          {
03.              int rowAffected = 0;
04.              using (IDbConnection con = new SqlConnection(strConnectionString))
05.              {
06.                  if (con.State == ConnectionState.Closed)
07.                      con.Open();
08.
09.                  DynamicParameters parameters = new DynamicParameters();
10.                  parameters.Add("@ProjectName", groupMeeting.ProjectName);
11.                  parameters.Add("@GroupMeetingLeadName", groupMeeting.GroupMeetingLeadName);
12.                  parameters.Add("@TeamLeadName", groupMeeting.TeamLeadName);
13.                  parameters.Add("@Description", groupMeeting.Description);
14.                  parameters.Add("@GroupMeetingDate", groupMeeting.GroupMeetingDate);
15.
16.                  rowAffected= con.Execute("InsertGroupMeeting",parameters,commandType:CommandType.St
17.              }
18.
19.              return rowAffected;
20.          }
```

Code to Update the group meeting using dapper is as follows.

```
01.  public static int UpdateGroupMeeting(GroupMeeting groupMeeting)
02.          {
03.              int rowAffected = 0;
04.
05.              using (IDbConnection con = new SqlConnection(strConnectionString))
06.              {
07.                  if (con.State == ConnectionState.Closed)
08.                      con.Open();
09.
10.                  DynamicParameters parameters = new DynamicParameters();
```

```
                parameters.Add("@GroupMeetingLeadName", groupMeeting.GroupMeetingLeadName);
                parameters.Add("@TeamLeadName", groupMeeting.TeamLeadName);
                parameters.Add("@Description", groupMeeting.Description);
                parameters.Add("@GroupMeetingDate", groupMeeting.GroupMeetingDate);
              rowAffected=  con.Execute("UpdateGroupMeeting",parameters,commandType:CommandType.Stc
            }

            return rowAffected;
        }
```

Code to delete the group meeting using dapper is as follows.

```
public static int DeleteGroupMeeting(int id)
        {
            int rowAffected = 0;
            using (IDbConnection con = new SqlConnection(strConnectionString))
            {
                if (con.State == ConnectionState.Closed)
                    con.Open();
                DynamicParameters parameters = new DynamicParameters();
                parameters.Add("@Id",id);
                rowAffected = con.Execute("DeleteGroupMeeting",parameters,commandType:CommandType.S

            }

            return rowAffected;
        }
```

As of now, we have completed the model related changes Get, Insert, Update, Delete using dapper ORM in the above code.

Step 11

Step 12

Select MVC Controller Empty and click on add button as follows.

Provide a meaningful name like “GroupMeeting” as follows.

Step 13

After adding the controller, you need to import the required namespace. Then added the following code to get all group meeting details and pass to the view as follow.

```
01. using System;
02. using System.Collections.Generic;
03. using System.Linq;
04. using System.Threading.Tasks;
05. using Microsoft.AspNetCore.Mvc;
06. using GroupMeetingASP.NETCoreWebApp.Models;
07.
08. namespace GroupMeetingASP.NETCoreWebApp.Controllers
09. {
10.     public class GroupMeetingController : Controller
11.     {
12.         public IActionResult Index()
13.         {
14.             return View(GroupMeeting.GetGroupMeetings());
15.         }
16.     }
17. }
```

Right click on ActionResult and click on Add view as follow.

Step 15

Write code for displaying group meeting data on Index.cshtml view as follows.

```
@model IEnumerable<GroupMeetingASP.NETCoreWebApp.Models.GroupMeeting>
@{
    ViewData["Title"] = "Index";
}

<h4>Group Meeting Web App</h4><hr />
<h4>
    <a asp-action="AddGroupMeeting">Add GroupMeeting</a>
</h4>
<div>
    <table class="table table-responsive table-bordered panel-primary">
        <thead>
            <th>Project Name</th>
            <th>Group Lead Name</th>
            <th>Team Lead Name</th>
            <th>Description</th>
            <th>Meeting Date</th>
        <th></th>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
            <tr>
                <td>@Html.DisplayFor(model => item.ProjectName)</td>
                <td>@Html.DisplayFor(model => item.GroupMeetingLeadName)</td>
```

```
                    <td>@Html.DisplayFor(model => item.GroupMeetingDate)</td>
                    <td>
                        <a asp-action="EditMeeting" asp-route-id="@item.Id">Edit</a>|
                        <a asp-action="DeleteMeeting" asp-route-id="@item.Id">Delete</a>
                    </td>
                </tr>

                }
            </tbody>
        </table>
</div>
```

Click on Startup.cs file and change the Controller name to "GroupMeeting" controller as follows.

```
app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=GroupMeeting}/{action=Index}/{id?}");
            });
```

After changing the controller name in startup.cs file click on IIS Express to run the application OR F5.

Index page display is as follows.

Step 16

We will create “AddGroupMeeting” view in the GroupMeeting controller and write the following code. We have written two views, "HttpGet" and "HttpPost" as follows.

```
        {
            return View();
        }

        [HttpPost]
        public IActionResult AddGroupMeeting([Bind] GroupMeeting groupMeeting)
        {
            if (ModelState.IsValid)
            {
                if (GroupMeeting.AddGroupMeeting(groupMeeting) > 0)
                {
                    return RedirectToAction("Index");
                }
            }
            return View(groupMeeting);
        }
```

Step 17

We will create “AddGroupMeeting.cshtml” view and write the code to add the group meeting detail as follows.

```
@model GroupMeetingASP.NETCoreWebApp.Models.GroupMeeting
@{
    ViewData["Title"] = "AddGroupMeeting";
}

<h2>Add Group Meeting</h2>
<div class="row">
    <div class="col-md-4">
        <form asp-action="AddGroupMeeting">
            <div class="">
                <label asp-for="ProjectName" class="control-label"></label>
                <input asp-for="ProjectName" class="form-control" />
                <span class="alert-danger" asp-validation-for="ProjectName"></span>
            </div>
```

```
                    <input asp-for="GroupMeetingLeadName" class="form-control" />
                    <span class="alert-danger" asp-validation-for="GroupMeetingLeadName"></span>
                </div>
                <div class="form-group">
                    <label asp-for="TeamLeadName" class="control-label"></label>
                    <input asp-for="TeamLeadName" class="form-control" />
                    <span class="alert-danger" asp-validation-for="TeamLeadName"></span>
                </div>
                <div class="form-group">
                    <label asp-for="Description" class="control-label"></label>
                    <input asp-for="Description" class="form-control" />

                    <span class="alert-danger" asp-validation-for="Description"></span>
                </div>
                <div class="form-group">
                    <label asp-for="GroupMeetingDate" class="control-label"></label>
                    <input asp-for="GroupMeetingDate" class="form-control" />
                    <span class="alert-danger" asp-validation-for="GroupMeetingDate"></span>
                </div>
                <div class="form-group">
                    <input type="submit" value="Create Meeting" class="btn btn-success btn-sm" />
                </div>
            </form>
        </div>
        <div class="col-md-8">

        </div>
</div>
<div>
    <a asp-action="Index">Back To Home</a>
</div>
```

"AddGroupMeeting" view display as follows.

Enter the valid information in the group meeting page and click on “Create Meeting” button. Then it will redirect to the index view page.

Step 18

We will create “EditMeeting” view in the GroupMeeting Controller and write the code to edit the group meeting detail as follows. There are two views of EditMeeting "HttpGet" and "HttpPost". The "HttpGet" edit meeting view has id as a parameter and gets called when clicking on "Edit" button on index view page.

```
01. [HttpGet]
02.         public IActionResult EditMeeting(int? id)
03.         {
04.             if (id == null)
05.             {
06.                 return NotFound();
07.             }
08.             GroupMeeting group = GroupMeeting.GetGroupMeetingById(id);
09.             if (group == null)
10.                 return NotFound();
11.             return View(group);
12.         }
13.
14.         [HttpPost]
15.         public IActionResult EditMeeting(int id, [Bind] GroupMeeting groupMeeting)
16.         {
17.             if (id != groupMeeting.Id)
18.                 return NotFound();
19.
20.             if (ModelState.IsValid)
```

```
                return RedirectToAction("Index");
            }
            return View(groupMeeting);
        }
```

We will create “EditMeeting.cshtml” view and write the code as follows.

```
@model GroupMeetingASP.NETCoreWebApp.Models.GroupMeeting
@{
    ViewData["Title"] = "EditMeeting";
}
<h4>Update the Meeting</h4>
<div class="row">
    <div class="col-md-4">
        <form asp-action="EditMeeting">
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="ProjectName" class="control-label"></label>
                <input asp-for="ProjectName" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="GroupMeetingLeadName" class="control-label"></label>
                <input asp-for="GroupMeetingLeadName" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="TeamLeadName" class="control-label"></label>
                <input asp-for="TeamLeadName" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <input asp-for="Description" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="GroupMeetingDate" class="control-label"></label>
                <input asp-for="GroupMeetingDate" class="form-control" />
```

```
                    <input type="submit" value="Edit Meeting" class="btn btn-success btn-sm" />
                </div>
            </form>
        </div>
        <div class="col-md-8">

        </div>
</div>
<div>
    <a asp-action="Index">Back To Home</a>
</div>
```

Click on "Edit" button on the index view page as follows.

It will open the following page when you click on the "edit" button. Then the group meeting will get updated into the database and redirected to the index view page.

Change the group meeting detail as per your requirement and click on "Edit Meeting" button to update the group meeting details. It will update the group meeting detail and redirect to the home page.

Step 19

We will create “DeleteMeeting” view in the GroupMeeting Controller and write the code to delete the group meeting detail as follows. There are two views of DeleteMeeting "HttpGet" and "HttpPost" as follow.

```
public IActionResult DeleteMeeting(int id)
        {
            GroupMeeting group = GroupMeeting.GetGroupMeetingById(id);
            if (group==null)
```

```
            }

            return View(group);
        }
        [HttpPost]
        public IActionResult DeleteMeeting(int id,GroupMeeting groupMeeting)
        {
            if (GroupMeeting.DeleteGroupMeeting(id) > 0)
            {
                return RedirectToAction("Index");
            }
            return View(groupMeeting);
        }
```

Step 20

Write the “DeleteMeeting.cshtml” view page code as mentioned below.

```
@model GroupMeetingASP.NETCoreWebApp.Models.GroupMeeting
@{
    ViewData["Title"] = "DeleteMeeting";
}

<h3 class="alert">Are you sure you want to delete this?</h3>
<div>
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.ProjectName)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.ProjectName)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.GroupMeetingLeadName)
        </dt>
```

```
            </dd>
            <dt>
                @Html.DisplayNameFor(model => model.TeamLeadName)
            </dt>
            <dd>
                @Html.DisplayFor(model => model.TeamLeadName)
            </dd>
            <dt>
                @Html.DisplayNameFor(model => model.Description)
            </dt>
            <dd>
                @Html.DisplayFor(model => model.Description)
            </dd>
            <dt>
                @Html.DisplayNameFor(model => model.GroupMeetingDate)
            </dt>
            <dd>
                @Html.DisplayFor(model => model.GroupMeetingDate)
            </dd>
        </dl>

        <form asp-action="DeleteMeeting">
            <input type="hidden" asp-for="Id" />
            <input type="submit" value="YES" class="btn btn-danger btn-sm" /> <a asp-
action="Index" class="btn btn-danger">NO</a>
        </form>
</div>
```

Run the application. The defualt index view page will open. Then, click on the "Delete" link as follows.

After clicking on the “delete” button on the Index View page, the following page will open for confirmation to delete the group meeting.

the Index View page which will look like this.

I hope you understand the basic concepts of CRUD operation using ASP.NET Core 2.0 with Razor View pages using Dapper ORM.

Next Recommended Article

ASP.NET Core - CRUD Using Angular 5 And Entity Framework Core

ASP.NET Core | CRUD Operation In ASP.NET Core | Dapper | Razor Pages Using Dapper

Shrimant Telgave TOP 500

473 295.5k 1

Shrimant Telgave is a Senior Software Developer having experience in Design, developing and maintaining large-scale applications. He has good experience in C#, ASP.NET, ASP.NET MVC, SQL Server, Web Services, WCF, Entity ... Read more

https://www.programmingwithshri.com/ https://in.linkedin.com/in/shrimant-telgave-9b488690

16 13

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

You can write crud operation in asp net core-2-0-using-dapper-orm + oop example. I need it!

Duy nguyễn

1881 11 0

Sep 19, 2019

2 2 Reply

473 4k 295.5k

2

It's great if you use all four properties of oop. If it is difficult to implement abstraction and polymorphism

Duy nguyễn

1881 11 0

Sep 22, 2019

1

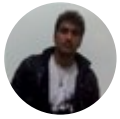
What are the benefits of using dapper? Why we are not using Entity Framework here?

Shailendra Mishra

1874 18 0

Mar 22, 2019

2 2 Reply



Dapper performs better than EF as it does not have to convert c# or linq expression to sql statements. We itself have to write sql statements in ORM layer and than we can map the outcome of sql statements to c# entities. So its execution time is faster as it does not have to convert c# statements to sql queries. Also there is a need of primary key in each table when using Entity framework but there is no need of primary key if not required.

Pawan Tiwari

1157 810 246.7k

Jun 08, 2019

3

Here is the best article to understand the benefit of Dapper ORM: https://www.c-sharpcorner.com/UploadFile/e4e3f7/dapper-king-of-micro-orm-C-Sharp-net/

Shrimant Telgave

473 4k 295.5k

Aug 09, 2019

1

Nice Article.....

Tanaji Patil

1878 14 0

Sep 10, 2018

2 1 Reply

Thank you.

Shrimant Telgave

473 4k 295.5k

Sep 10, 2018

1

ORM library support multi-database provider and support .NET Framework and .NET Standard so it should work cross-platform as well.

Does dapper support cross platform ?

Prakash Pati

1810 82 0

Aug 21, 2018

2 1 Reply

ORM library support multi-database provider and support .NET Framework and .NET Standard so it should work cross-platform as well.

Shrimant Telgave

473 4k 295.5k

Sep 09, 2018

0

Nice Article!!

Jitendra Waghale

1249 671 91k

Aug 20, 2018

1 1 Reply

Thanks jitu.....

Shrimant Telgave

473 4k 295.5k

Aug 20, 2018

0