Lab 3 – Managing data in SQL Server with ASP .Net Core: Part 1

Estimated usage time: 1 Hour 50 Minutes

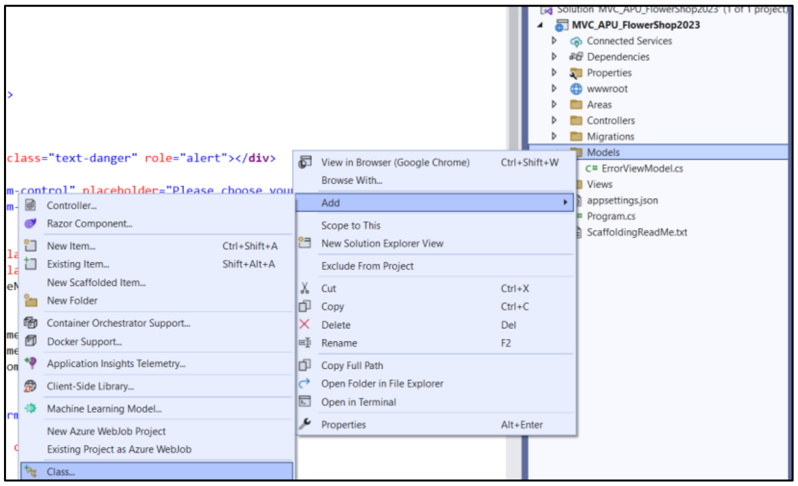
Lab 3.1. Create tables in SQL Server with ASP .Net Core.

This tutorial will teach the student how to create a table in the SQL database and how to add data in the SQL table.

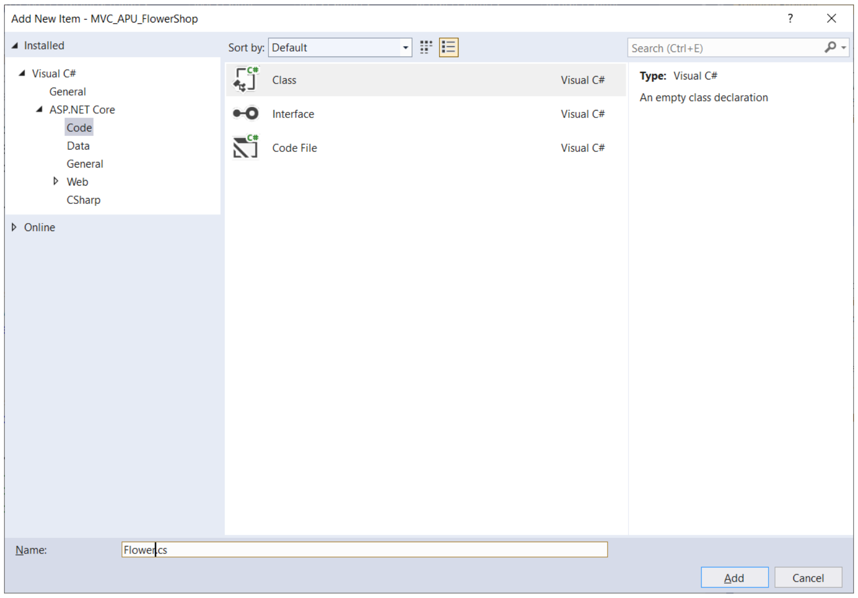
a. Create new table in existing Identity DB

(Estimation of Total Time Used: 20 minutes)

1. Continued from the **Lab 2.2** project.
2. Go to **Solution Explorer** > Right click on the **Models** Folder > Select **Add** > Select **Class**.



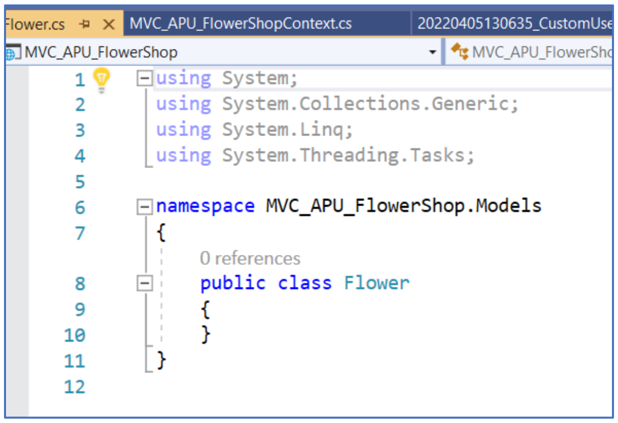
3. In the **Add New Item** dialog, named the class file as **Flower.cs**. Then, tap the **Add** button.



Type: Visual C#
An empty class declaration

Name: Flower.cs

4. An empty **Flower.cs** file is generated as shown below.

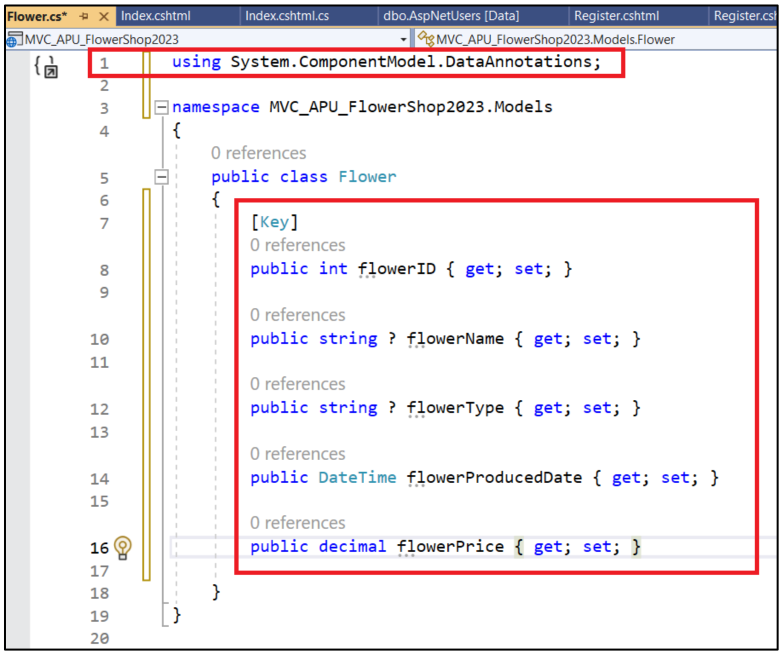


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MVC_APU_FlowerShop.Models
{
    public class Flower
    {
    }
}
```

5. Assume we need to prepare a table as below:

Flower ID	Flower Name	Flower Type	Flower Produced Date	Flower Price

Thus, edit the **Flower.cs** model class file based on the above table structure. Save the file after finish editing the file.

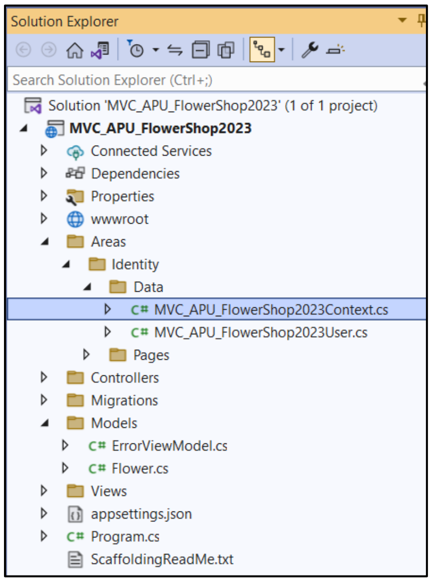


```
Flower.cs*   Index.cshtml   Index.cshtml.cs   dbo.AspNetUsers [Data]   Register.cshtml   Register.cs
MVC_APU_FlowerShop2023                    MVC_APU_FlowerShop2023.Models.Flower
using System.ComponentModel.DataAnnotations;

namespace MVC_APU_FlowerShop2023.Models
{
    0 references
    public class Flower
    {
        [Key]
        0 references
        public int flowerID { get; set; }

        0 references
        public string ? flowerName { get; set; }

        0 references
        public string ? flowerType { get; set; }

        0 references
        public DateTime flowerProducedDate { get; set; }

        0 references
        public decimal flowerPrice { get; set; }
    }
}
```

```
using System.ComponentModel.DataAnnotations;
```

```
    public class Flower
    {
        [Key]
        public int FlowerID { get; set; }

        public string ? FlowerName { get; set; }

        public string ? FlowerType { get; set; }

        public DateTime FlowerProducedDate { get; set; }

        public decimal FlowerPrice { get; set; }
    }
```

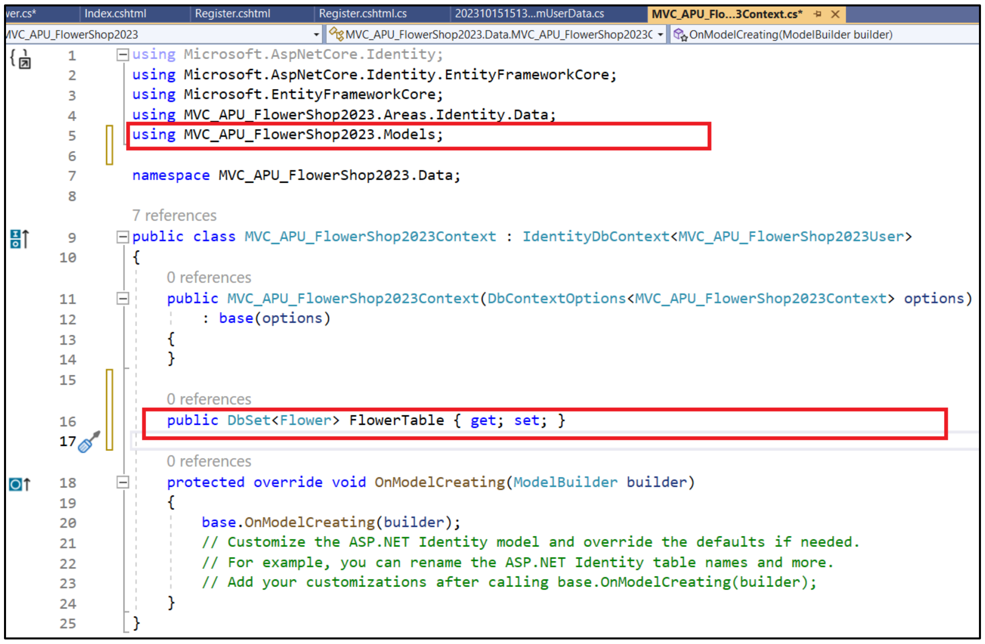
6. Now, open the **MVC_APU_FlowerShop2023Context.cs.**



7. Then, add the new table details to the context class: **MVC_APU_FlowerShop2023Context.cs**.

```
using MVC_APU_FlowerShop2023.Models;
```

```
public DbSet<Flower> FlowerTable { get; set; }
```

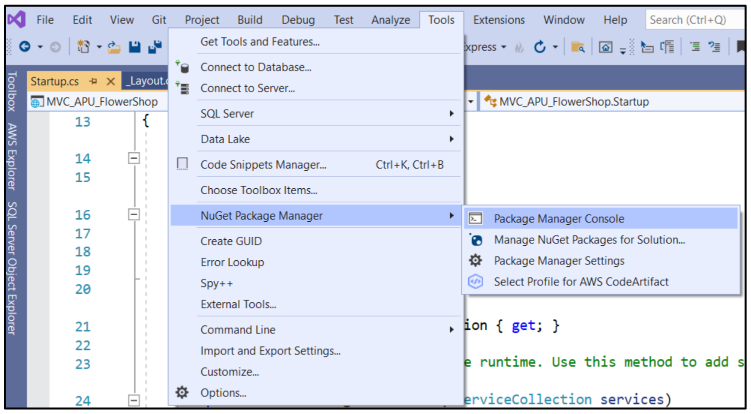

ver.cs*
Index.cshtml
Register.cshtml
Register.cshtml.cs
202310151513...mUserData.cs
MVC_APU_Flo...3Context.cs*
MVC_APU_FlowerShop2023
MVC_APU_FlowerShop2023.Data.MVC_APU_FlowerShop2023C
OnModelCreating(ModelBuilder builder)

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using MVC_APU_FlowerShop2023.Areas.Identity.Data;
using MVC_APU_FlowerShop2023.Models;

namespace MVC_APU_FlowerShop2023.Data;

7 references
public class MVC_APU_FlowerShop2023Context : IdentityDbContext<MVC_APU_FlowerShop2023User>
{
    0 references
    public MVC_APU_FlowerShop2023Context(DbContextOptions<MVC_APU_FlowerShop2023Context> options)
        : base(options)
    {
    }

    0 references
    public DbSet<Flower> FlowerTable { get; set; }

    0 references
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);
        // Customize the ASP.NET Identity model and override the defaults if needed.
        // For example, you can rename the ASP.NET Identity table names and more.
        // Add your customizations after calling base.OnModelCreating(builder);
    }
}
```

b. Create Migration Schema Code for New Flower Table

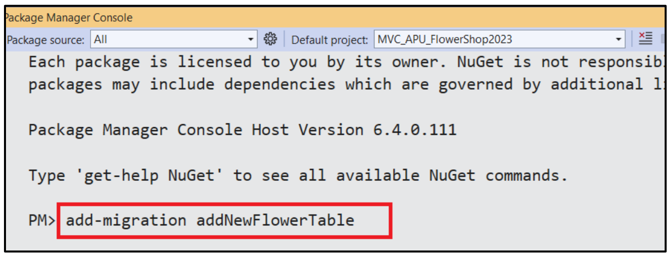
(Estimation of Total Time Used: 15 minutes)

1. Now, start the PMC, click on the **Tools** > **NuGet Package Manager** > **Package Manager Console.**

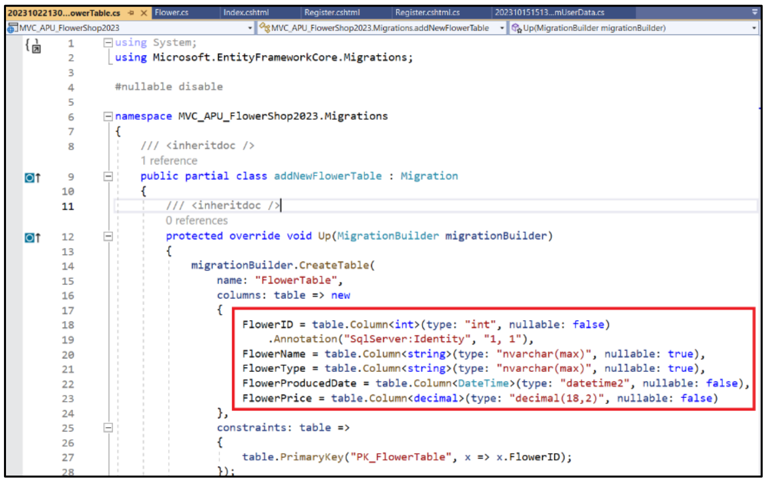


2. In the Visual Studio **Package Manager Console**, type the below commands:

Add-Migration addNewFlowerTable



```
Each package is licensed to you by its owner. NuGet is not responsib
packages may include dependencies which are governed by additional l

Package Manager Console Host Version 6.4.0.111

Type 'get-help NuGet' to see all available NuGet commands.

PM> add-migration addNewFlowerTable
```

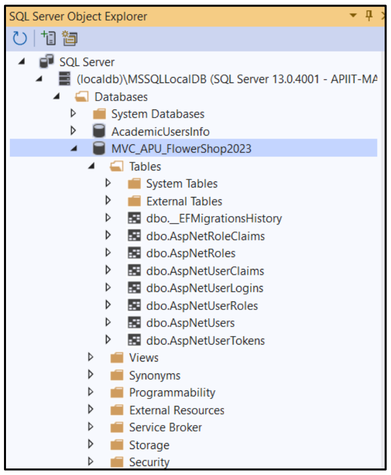

20231022130...owerTable.cs
Flower.cs
Index.cshtml
Register.cshtml
Register.cshtml.cs
202310151513...mUserData.cs
MVC_APU_FlowerShop2023
MVC_APU_FlowerShop2023.Migrations.addNewFlowerTable
Up(MigrationBuilder migrationBuilder)

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace MVC_APU_FlowerShop2023.Migrations
{
    /// <inheritdoc />
    1 reference
    public partial class addNewFlowerTable : Migration
    {
        /// <inheritdoc />
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "FlowerTable",
                columns: table => new
                {
                    FlowerID = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    FlowerName = table.Column<string>(type: "nvarchar(max)", nullable: true),
                    FlowerType = table.Column<string>(type: "nvarchar(max)", nullable: true),
                    FlowerProducedDate = table.Column<DateTime>(type: "datetime2", nullable: false),
                    FlowerPrice = table.Column<decimal>(type: "decimal(18,2)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_FlowerTable", x => x.FlowerID);
                });
```

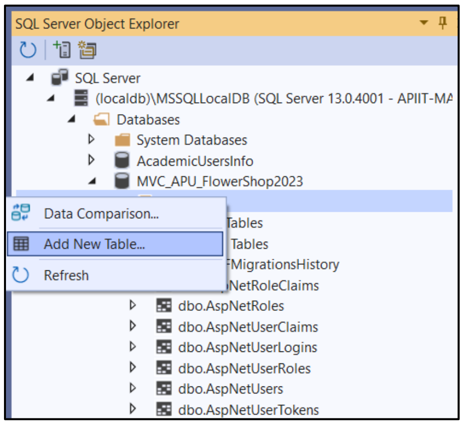
c. Learn How to Manually Setup the Table in SQL Server Without Using Update-Database Command.

(Estimation of Total Time Used: 15 minutes)

1. **SQL Server Object Explorer (SSOX)** in Visual Studio and create the table in the MVC_APU_FlowerShop2023.

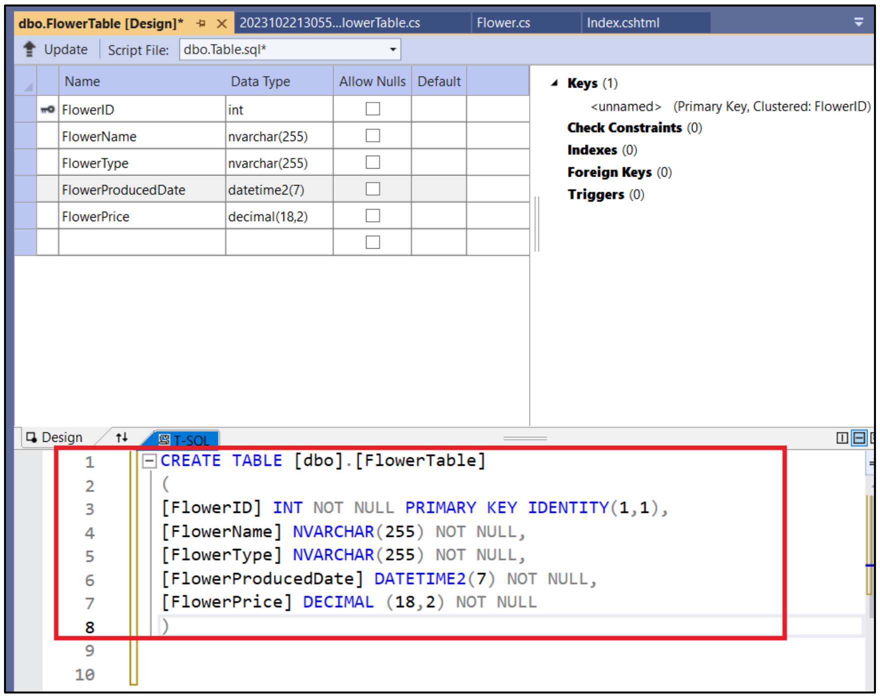


2. Right click the **Databases** > **MVC_APU_FlowerShop2023 > Tables > Add New Table….**



3. Add the below SQL Queries to the T-SQL corner.

```
CREATE TABLE [dbo].[FlowerTable]
(
   [FlowerID] INT NOT NULL PRIMARY KEY IDENTITY(1,1),
   [FlowerName] NVARCHAR(255) NOT NULL,
   [FlowerType] NVARCHAR(255) NOT NULL,
   [FlowerProducedDate] DATETIME2(7) NOT NULL,
   [FlowerPrice] DECIMAL (18,2) NOT NULL
)
```

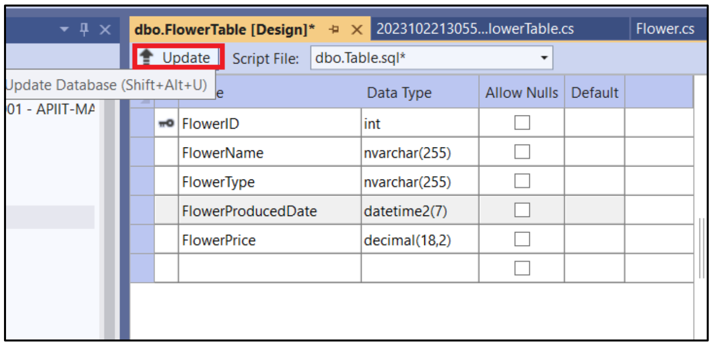

dbo.FlowerTable [Design]*
2023102213055...lowerTable.cs
Flower.cs
Index.cshtml
Update
Script File: dbo.Table.sql*
Design
T-SQL

Name	Data Type	Allow Nulls	Default
FlowerID	int	☐	
FlowerName	nvarchar(255)	☐	
FlowerType	nvarchar(255)	☐	
FlowerProducedDate	datetime2(7)	☐	
FlowerPrice	decimal(18,2)	☐	
		☐	

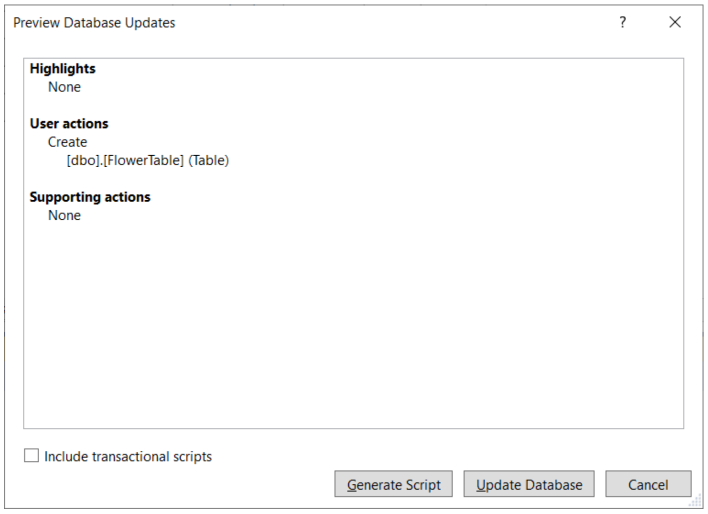
Keys (1)
<unnamed> (Primary Key, Clustered: FlowerID)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```sql
CREATE TABLE [dbo].[FlowerTable]
(
[FlowerID] INT NOT NULL PRIMARY KEY IDENTITY(1,1),
[FlowerName] NVARCHAR(255) NOT NULL,
[FlowerType] NVARCHAR(255) NOT NULL,
[FlowerProducedDate] DATETIME2(7) NOT NULL,
[FlowerPrice] DECIMAL (18,2) NOT NULL
)
```

4. Now, click on the Update button to add the table to the database.


dbo.FlowerTable [Design]*
2023102213055...lowerTable.cs
Flower.cs
Update
Script File: dbo.Table.sql*
Update Database (Shift+Alt+U)
01 - APIIT-MA

Name	Data Type	Allow Nulls	Default
FlowerID	int	☐	
FlowerName	nvarchar(255)	☐	
FlowerType	nvarchar(255)	☐	
FlowerProducedDate	datetime2(7)	☐	
FlowerPrice	decimal(18,2)	☐	
		☐	


Preview Database Updates

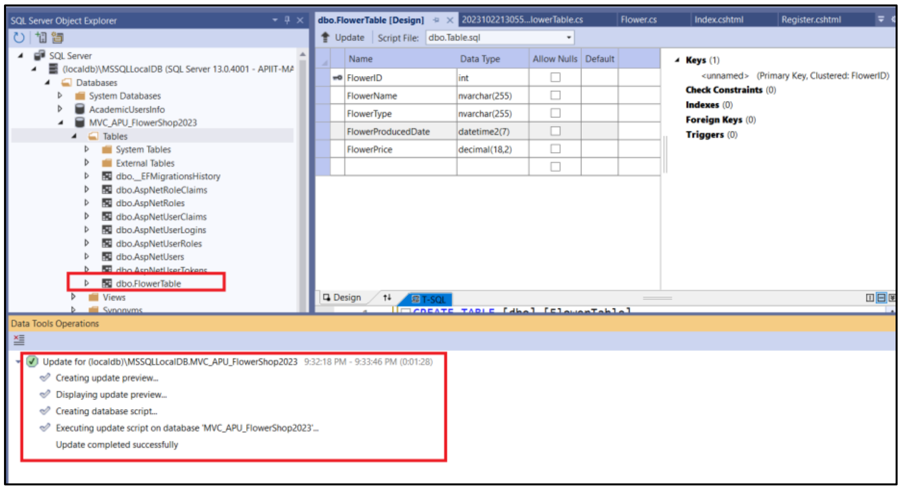
Highlights
None

User actions
Create
[dbo].[FlowerTable] (Table)

Supporting actions
None

☐ Include transactional scripts

5. Finally, click on the Update Database button and the Flower table will be added to your current database.



Name	Data Type	Allow Nulls	Default
FlowerID	int	☐	
FlowerName	nvarchar(255)	☐	
FlowerType	nvarchar(255)	☐	
FlowerProducedDate	datetime2(7)	☐	
FlowerPrice	decimal(18,2)	☐	
		☐	

```
Keys (1)
    <unnamed>  (Primary Key, Clustered: FlowerID)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)
```

```
Update for (localdb)\MSSQLLocalDB.MVC_APU_FlowerShop2023   9:32:18 PM - 9:33:46 PM (0:01:28)
    Creating update preview...
    Displaying update preview...
    Creating database script...
    Executing update script on database 'MVC_APU_FlowerShop2023'...
    Update completed successfully
```

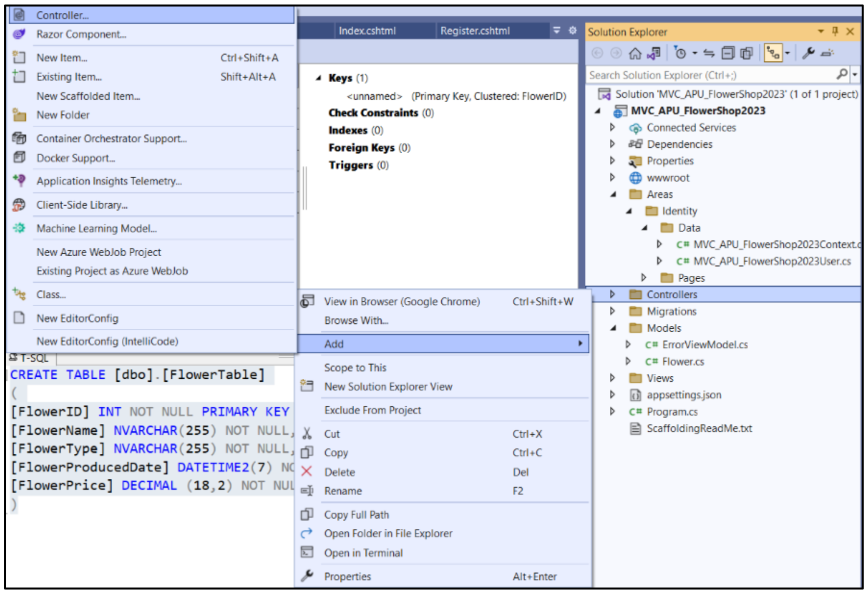
6. Next step we need to learn how to add the data to the Flower Table.

d. Add data into the Flower table.

(Estimation of Total Time Used: 40 minutes)

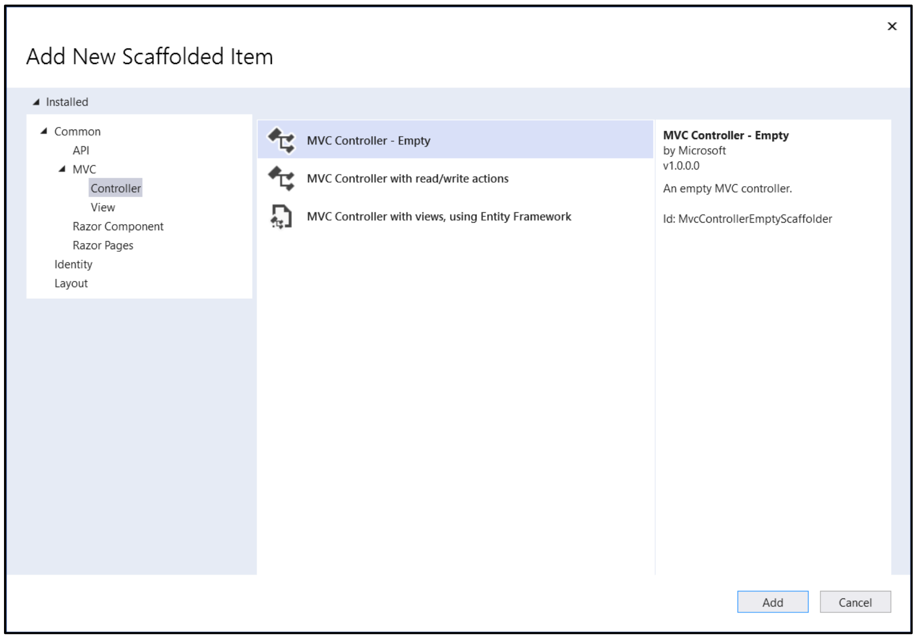
1. Create a controller for managing the flower views.

2. Go to **Solution Explorer** > Right click on the **Controllers** Folder > Select **Add** > Select **Controller**.



```
CREATE TABLE [dbo].[FlowerTable]
(
[FlowerID] INT NOT NULL PRIMARY KEY
[FlowerName] NVARCHAR(255) NOT NULL,
[FlowerType] NVARCHAR(255) NOT NULL,
[FlowerProducedDate] DATETIME2(7) NO
[FlowerPrice] DECIMAL (18,2) NOT NUL
)
```

3. In the **Add New Scaffolded Item** dialog, select **MVC Controller – Empty**. Then select **Add**.

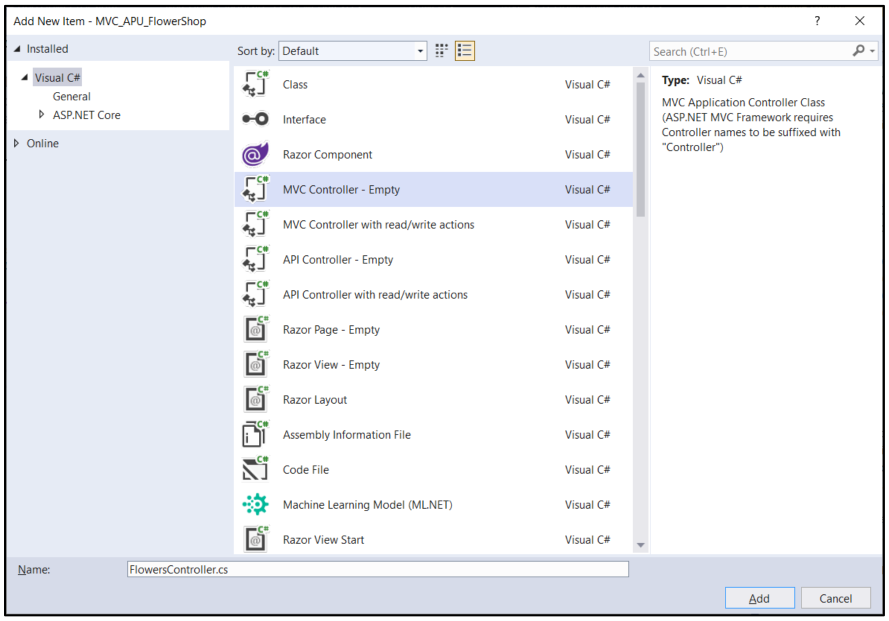


MVC Controller - Empty
by Microsoft
v1.0.0.0

An empty MVC controller.

Id: MvcControllerEmptyScaffolder

4. In the **Add New Item** dialog, select **MVC Controller – Empty** and name the file as **FlowersController.cs**.


Add New Item - MVC_APU_FlowerShop

Installed
- Visual C#
  - General
  - ASP.NET Core
- Online

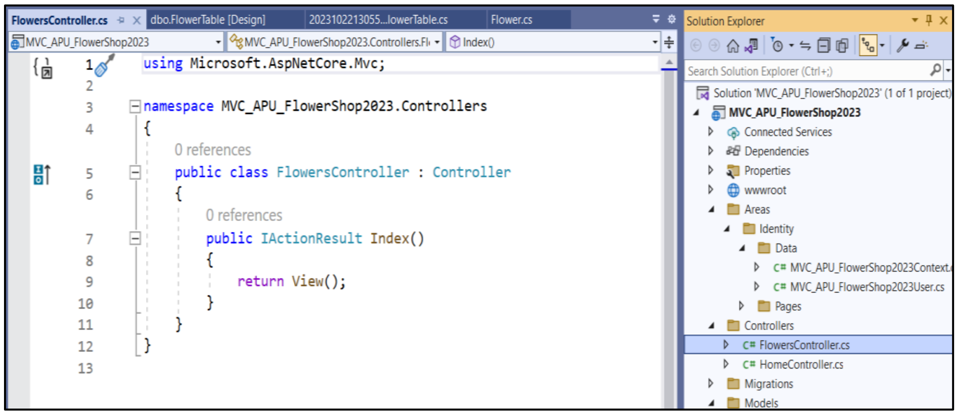
Sort by: Default

Class	Visual C#
Interface	Visual C#
Razor Component	Visual C#
MVC Controller - Empty	Visual C#
MVC Controller with read/write actions	Visual C#
API Controller - Empty	Visual C#
API Controller with read/write actions	Visual C#
Razor Page - Empty	Visual C#
Razor View - Empty	Visual C#
Razor Layout	Visual C#
Assembly Information File	Visual C#
Code File	Visual C#
Machine Learning Model (ML.NET)	Visual C#
Razor View Start	Visual C#

Type: Visual C#
MVC Application Controller Class (ASP.NET MVC Framework requires Controller names to be suffixed with "Controller")

Name: FlowersController.cs

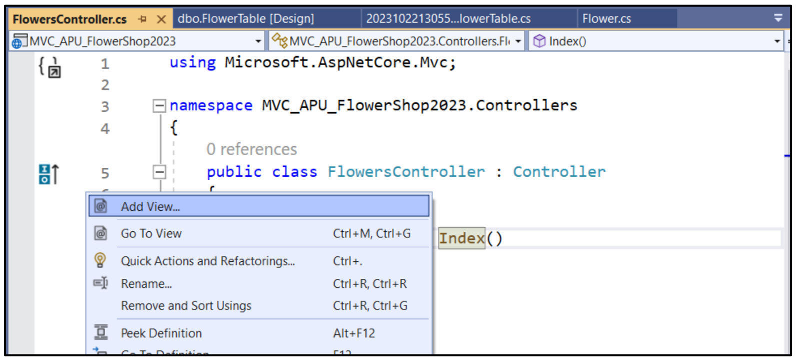
5. Then, select the $\boxed{\textbf{Add}}$ button. A new controller has been added to your **Controllers** folder now.



```
using Microsoft.AspNetCore.Mvc;

namespace MVC_APU_FlowerShop2023.Controllers
{
    public class FlowersController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Solution Explorer
- Solution 'MVC_APU_FlowerShop2023' (1 of 1 project)
  - MVC_APU_FlowerShop2023
    - Connected Services
    - Dependencies
    - Properties
    - wwwroot
    - Areas
      - Identity
        - Data
          - MVC_APU_FlowerShop2023Context
          - MVC_APU_FlowerShop2023User.cs
        - Pages
    - Controllers
      - FlowersController.cs
      - HomeController.cs
    - Migrations
    - Models

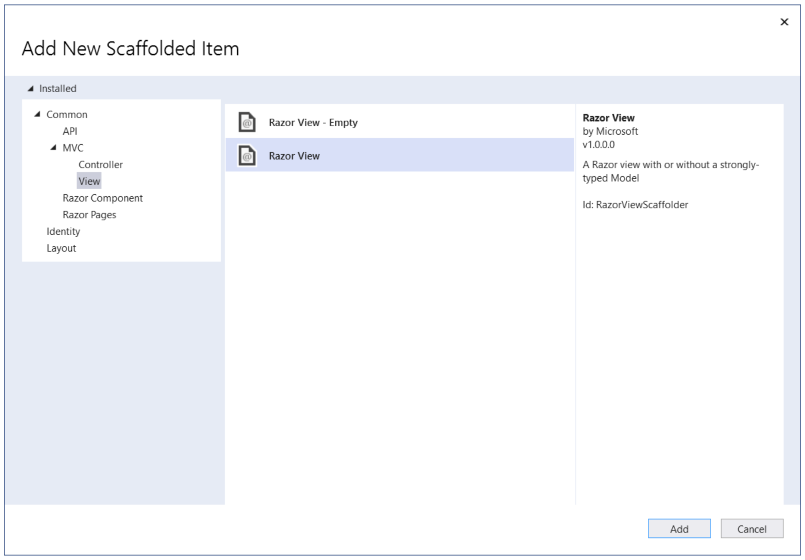
6. Now, highlight the `Index()` function and right click on the highlighted function. Select **Add View**.



```
using Microsoft.AspNetCore.Mvc;

namespace MVC_APU_FlowerShop2023.Controllers
{
    0 references
    public class FlowersController : Controller
    {
        Index()
```

FlowersController.cs
dbo.FlowerTable [Design]
2023102213055...lowerTable.cs
Flower.cs
MVC_APU_FlowerShop2023
MVC_APU_FlowerShop2023.Controllers.Fl
Index()
Add View...
Go To View Ctrl+M, Ctrl+G
Quick Actions and Refactorings... Ctrl+.
Rename... Ctrl+R, Ctrl+R
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12

7. In the **Add New Scaffolded Item** dialog, select **Razor View** and press button **Add**.

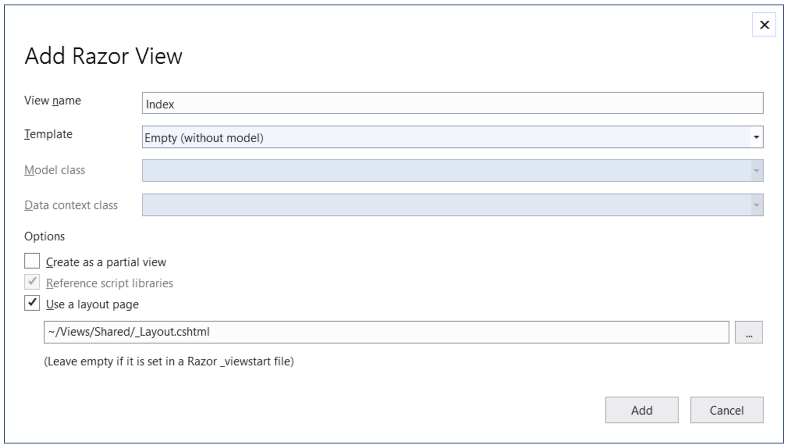


Add New Scaffolded Item

Razor View
by Microsoft
v1.0.0.0
A Razor view with or without a strongly-typed Model

Id: RazorViewScaffolder

8. In the **Add Razor View** dialog, directly create on the **Add** button.



Add Razor View

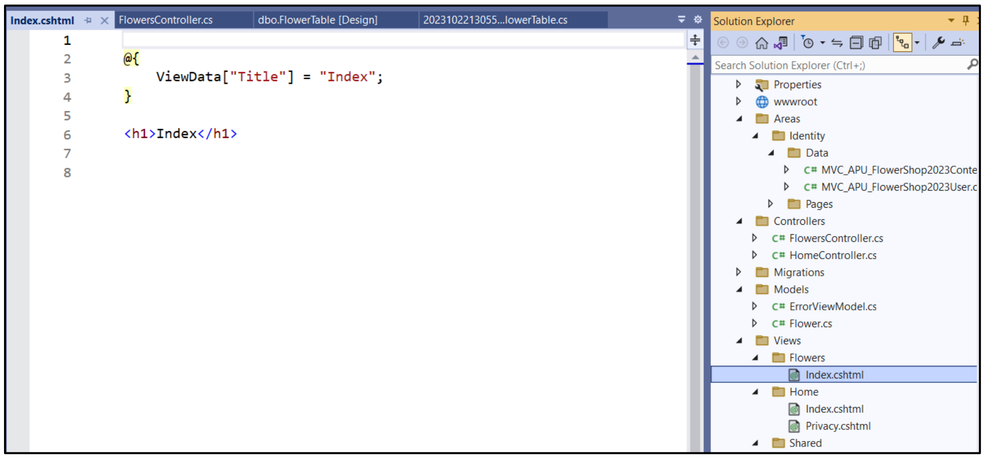
View name: Index
Template: Empty (without model)
Model class:
Data context class:

Options
- [ ] Create as a partial view
- [✓] Reference script libraries
- [✓] Use a layout page

~/Views/Shared/_Layout.cshtml

(Leave empty if it is set in a Razor _viewstart file)

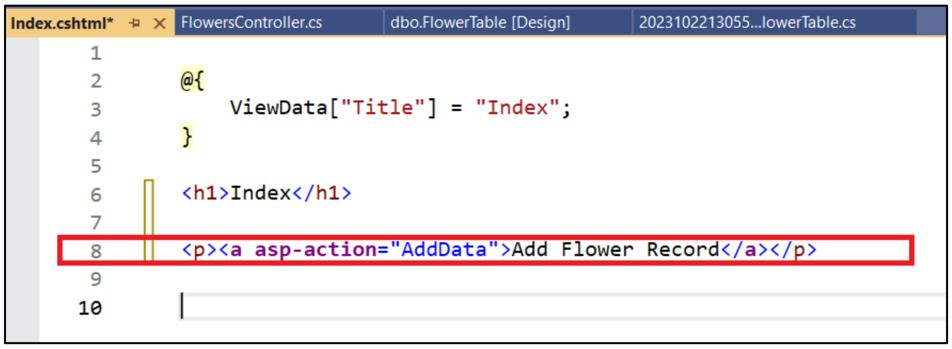
9. The **Index.cshtml** page will be added to a folder named **Flowers**.



```
@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

```

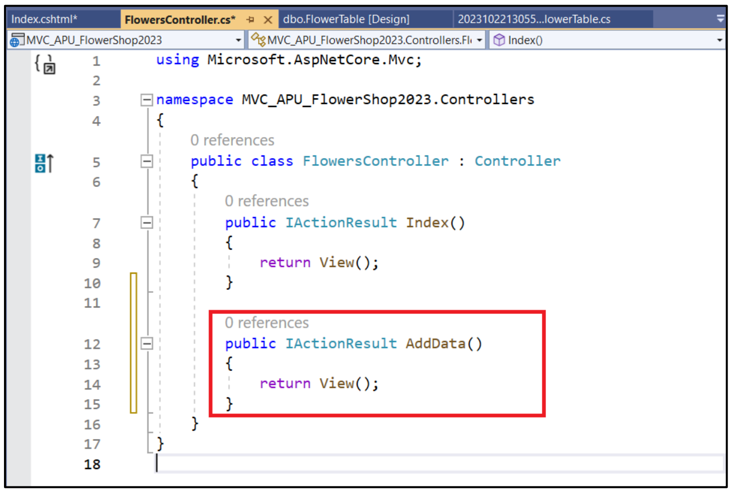
10. Now, in the **Index.cshtml**, add the below sentence to link to your Index.cshtml page with the coming Add Data page.

```
<p><a asp-action="AddData">Add Flower Record</a></p>
```



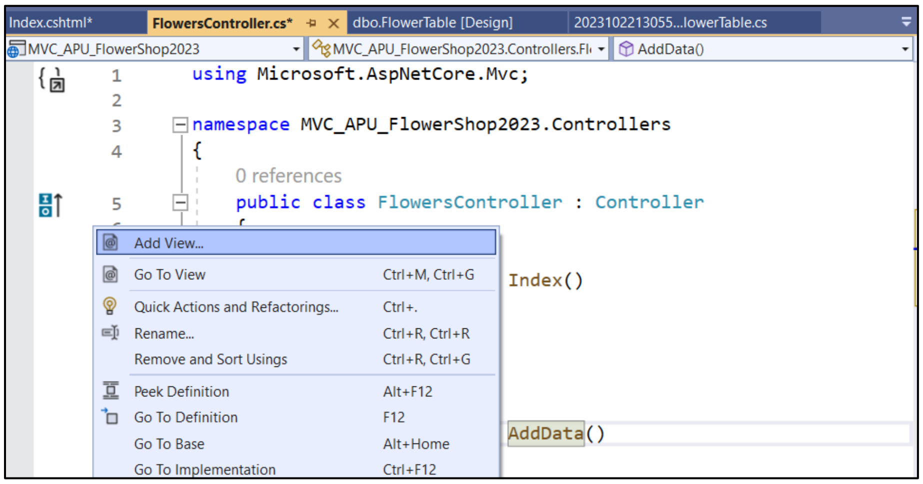
```
@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p><a asp-action="AddData">Add Flower Record</a></p>

```

11. To add data to the Flower table, we need to **create another .cshtml page**. To create that page, we must **write a function named AddData** in the controller file.



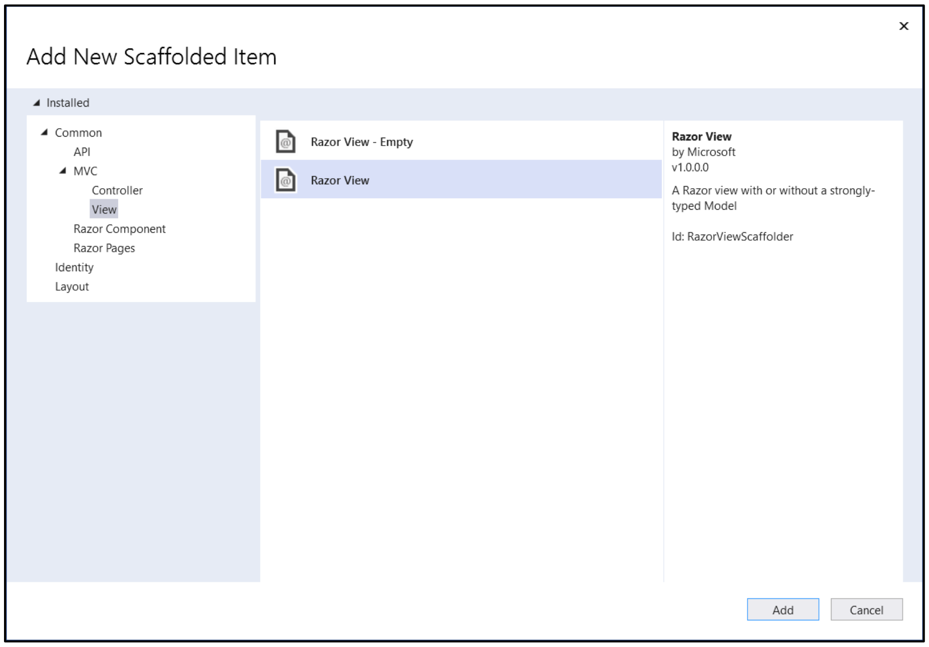
```
using Microsoft.AspNetCore.Mvc;

namespace MVC_APU_FlowerShop2023.Controllers
{
    0 references
    public class FlowersController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View();
        }

        0 references
        public IActionResult AddData()
        {
            return View();
        }
    }
}
```

12. Now, same as the creation steps in **Index.cshtml** page, we must right click on the **AddData()** function.



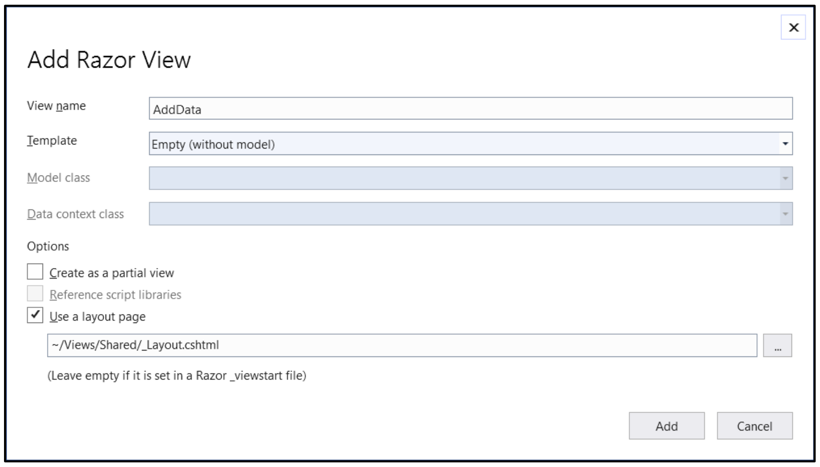
```
using Microsoft.AspNetCore.Mvc;

namespace MVC_APU_FlowerShop2023.Controllers
{
    public class FlowersController : Controller
                    Index()
                    AddData()
```

13. In the **Add New Scaffolded Item** dialog, select **Razor View** and press button **Add**.



Razor View
by Microsoft
v1.0.0.0
A Razor view with or without a strongly-typed Model
Id: RazorViewScaffolder

14. In the **Add Razor View** dialog, directly create on the **Add** button.



Add Razor View

View name: AddData

Template: Empty (without model)

Model class:

Data context class:

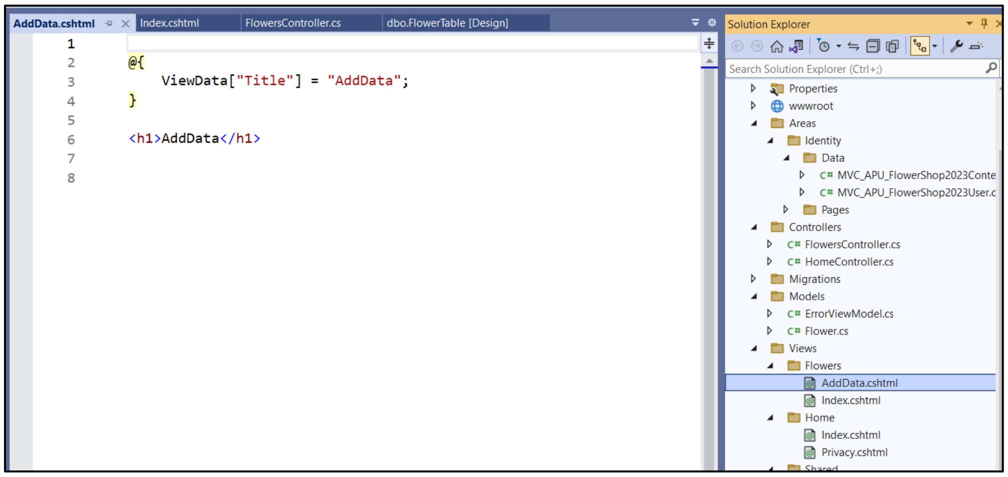
Options

- [ ] Create as a partial view
- [ ] Reference script libraries
- [x] Use a layout page

~/Views/Shared/_Layout.cshtml

(Leave empty if it is set in a Razor _viewstart file)

15. The **AddData.cshtml** page will be added to a folder named as **Flowers**.

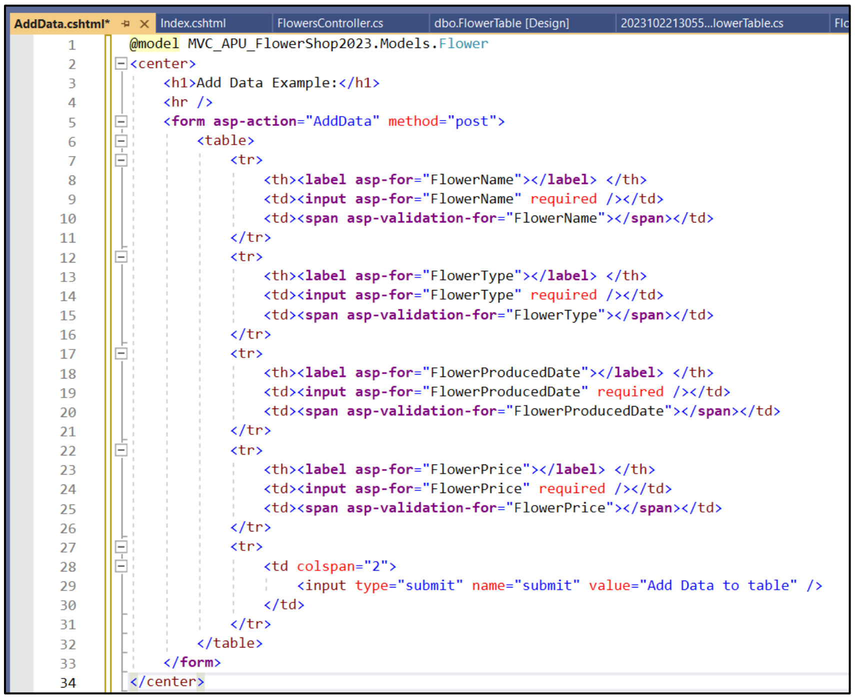


```
@{
    ViewData["Title"] = "AddData";
}

<h1>AddData</h1>
```

```
Properties
wwwroot
Areas
  Identity
    Data
      MVC_APU_FlowerShop2023Conte
      MVC_APU_FlowerShop2023User.c
    Pages
Controllers
  FlowersController.cs
  HomeController.cs
Migrations
Models
  ErrorViewModel.cs
  Flower.cs
Views
  Flowers
    AddData.cshtml
    Index.cshtml
  Home
    Index.cshtml
    Privacy.cshtml
  Shared
```

16. Now, design the Add Data page using the below code:

```
@model MVC_APU_FlowerShop2023.Models.Flower
<center>
    <h1>Add Data Example:</h1>
    <hr />
    <form asp-action="AddData" method="post">
        <table>
            <tr>
                <th><label asp-for="FlowerName"></label> </th>
                <td><input asp-for="FlowerName" required /></td>
                <td><span asp-validation-for="FlowerName"></span></td>
            </tr>
            <tr>
                <th><label asp-for="FlowerType"></label> </th>
                <td><input asp-for="FlowerType" required /></td>
                <td><span asp-validation-for="FlowerType"></span></td>
            </tr>
            <tr>
                <th><label asp-for="FlowerProducedDate"></label> </th>
                <td><input asp-for="FlowerProducedDate" required /></td>
                <td><span asp-validation-for="FlowerProducedDate"></span></td>
            </tr>
            <tr>
                <th><label asp-for="FlowerPrice"></label> </th>
                <td><input asp-for="FlowerPrice" required /></td>
                <td><span asp-validation-for="FlowerPrice"></span></td>
            </tr>
            <tr>
                <td colspan="2">
                    <input type="submit" name="submit" value="Add Data to table" />
                </td>
            </tr>
        </table>
    </form>
    </center>
```



```
@model MVC_APU_FlowerShop2023.Models.Flower
<center>
    <h1>Add Data Example:</h1>
    <hr />
    <form asp-action="AddData" method="post">
        <table>
            <tr>
                <th><label asp-for="FlowerName"></label> </th>
                <td><input asp-for="FlowerName" required /></td>
                <td><span asp-validation-for="FlowerName"></span></td>
            </tr>
            <tr>
                <th><label asp-for="FlowerType"></label> </th>
                <td><input asp-for="FlowerType" required /></td>
                <td><span asp-validation-for="FlowerType"></span></td>
            </tr>
            <tr>
                <th><label asp-for="FlowerProducedDate"></label> </th>
                <td><input asp-for="FlowerProducedDate" required /></td>
                <td><span asp-validation-for="FlowerProducedDate"></span></td>
            </tr>
            <tr>
                <th><label asp-for="FlowerPrice"></label> </th>
                <td><input asp-for="FlowerPrice" required /></td>
                <td><span asp-validation-for="FlowerPrice"></span></td>
            </tr>
            <tr>
                <td colspan="2">
                    <input type="submit" name="submit" value="Add Data to table" />
                </td>
            </tr>
        </table>
    </form>
</center>
```

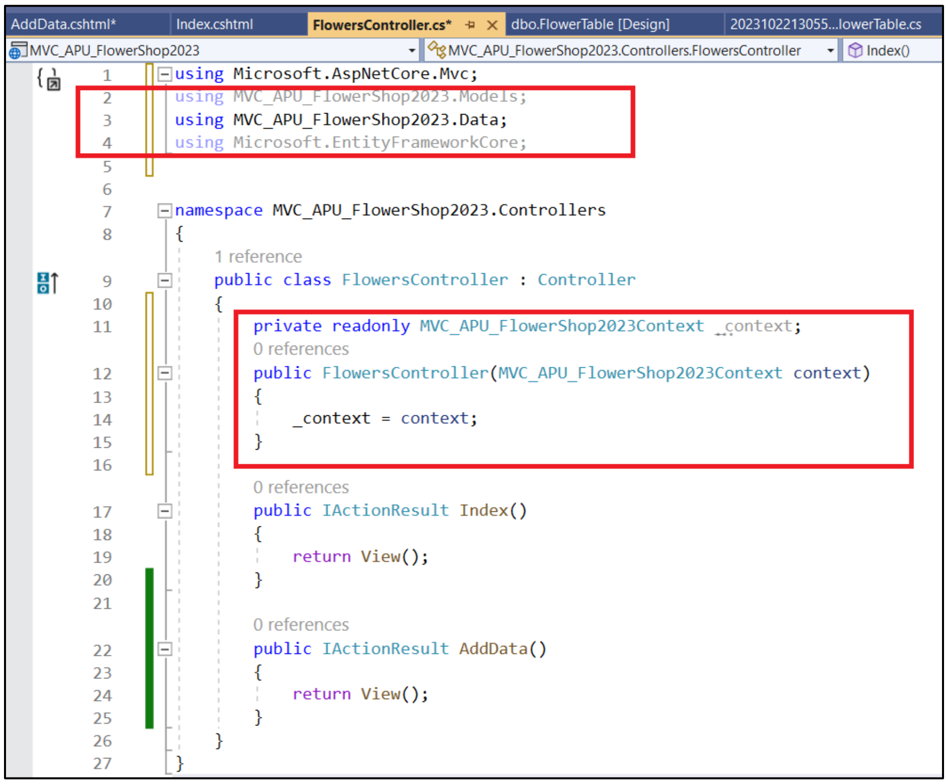
17. Now, go back to **FlowersController.cs** again.

18. First, add the below libraries into the controller.

```
using MVC_APU_FlowerShop2023.Models;
using MVC_APU_FlowerShop2023.Data;
using Microsoft.EntityFrameworkCore;
```

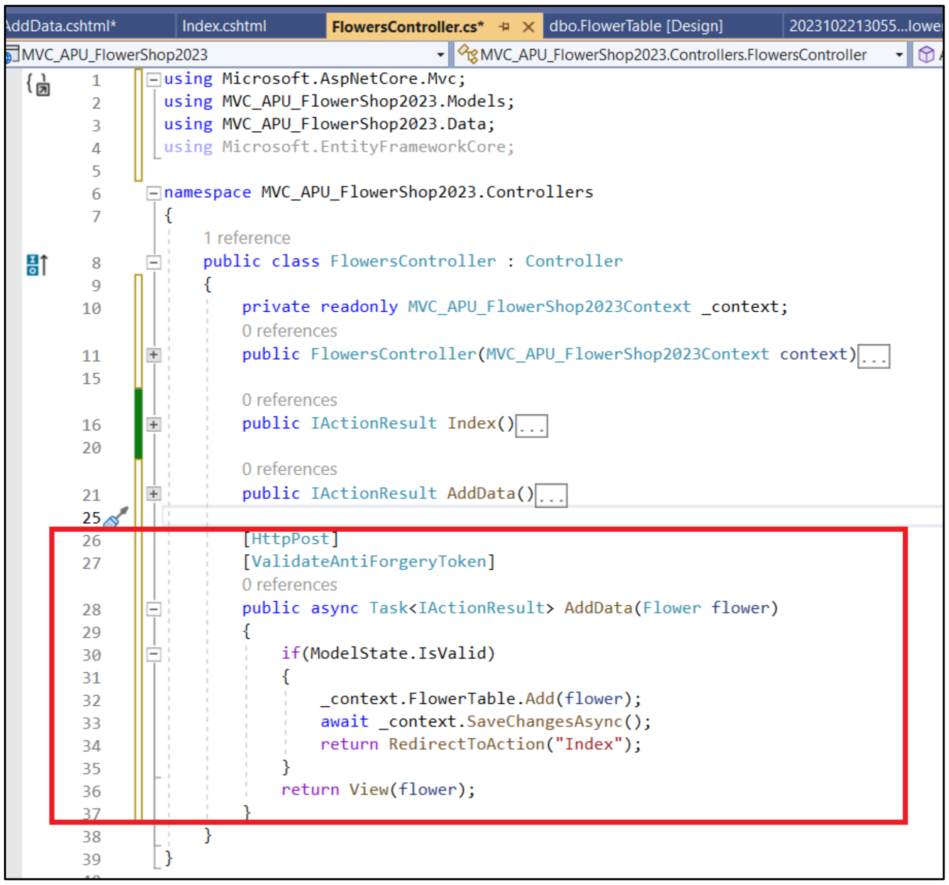
19. To connect to the database, add the below lines inside the controller.

```
        private readonly MVC_APU_FlowerShop2023Context _context;

        public FlowersController(MVC_APU_FlowerShop2023Context context)
        {
            _context = context;
        }
```



```
using Microsoft.AspNetCore.Mvc;
using MVC_APU_FlowerShop2023.Models;
using MVC_APU_FlowerShop2023.Data;
using Microsoft.EntityFrameworkCore;


namespace MVC_APU_FlowerShop2023.Controllers
{
    1 reference
    public class FlowersController : Controller
    {
        private readonly MVC_APU_FlowerShop2023Context _context;
        0 references
        public FlowersController(MVC_APU_FlowerShop2023Context context)
        {
            _context = context;
        }

        0 references
        public IActionResult Index()
        {
            return View();
        }

        0 references
        public IActionResult AddData()
        {
            return View();
        }
    }
}
```

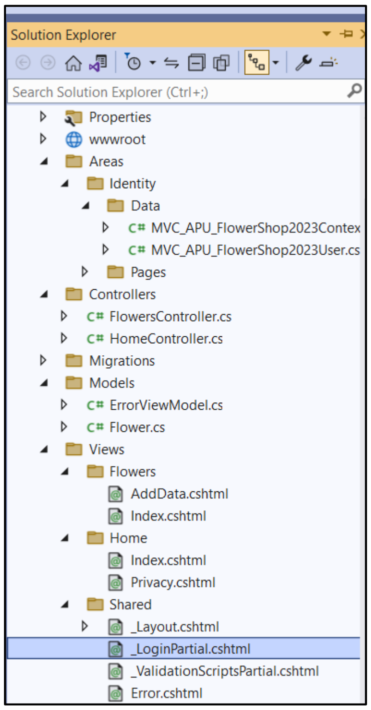
20. Now, add another **AddData()** function to receive the user inputs and store the inputs to the database.



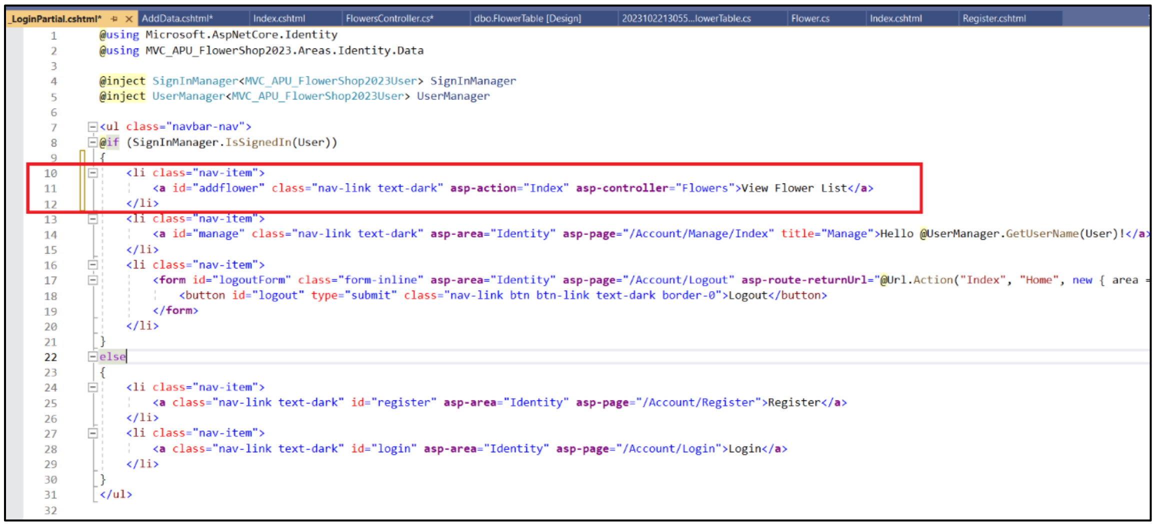
```
using Microsoft.AspNetCore.Mvc;
using MVC_APU_FlowerShop2023.Models;
using MVC_APU_FlowerShop2023.Data;
using Microsoft.EntityFrameworkCore;

namespace MVC_APU_FlowerShop2023.Controllers
{
    1 reference
    public class FlowersController : Controller
    {
        private readonly MVC_APU_FlowerShop2023Context _context;
        0 references
        public FlowersController(MVC_APU_FlowerShop2023Context context)...

        0 references
        public IActionResult Index()...

        0 references
        public IActionResult AddData()...

        [HttpPost]
        [ValidateAntiForgeryToken]
        0 references
        public async Task<IActionResult> AddData(Flower flower)
        {
            if(ModelState.IsValid)
            {
                _context.FlowerTable.Add(flower);
                await _context.SaveChangesAsync();
                return RedirectToAction("Index");
            }
            return View(flower);
        }
    }
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AddData(Flower flower)
        {
            if (ModelState.IsValid)
            {
                _context.Add(flower);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(flower);
}
```

21. Now, to test the add data action, we must add the Flowers page link to the navigation bar. Thus, we must open the **_LoginPartial.cshtml** and attach the **Flowers** > **Index.cshtml** link inside the navigation bar.

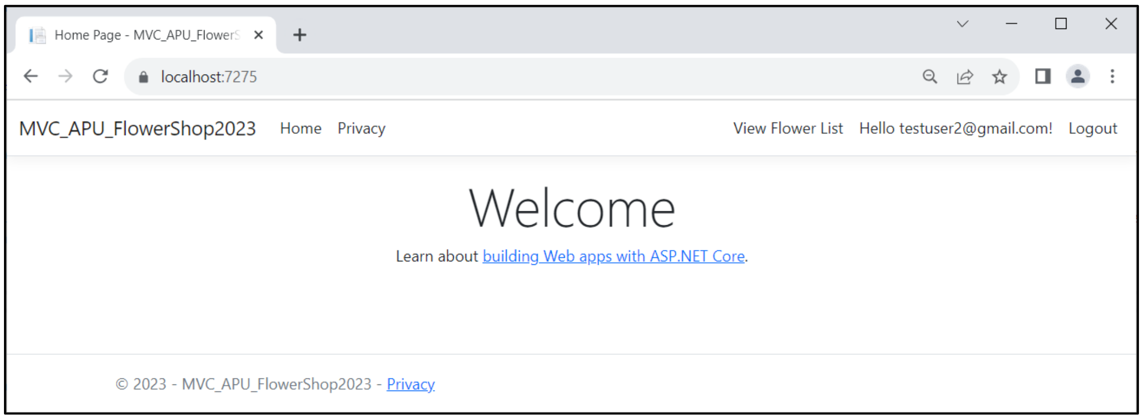


```
<li class="nav-item">
<a id="addflower" class="nav-link text-dark" asp-action="Index" asp-controller="Flowers">
View Flower List</a>
</li>
```



```
@using Microsoft.AspNetCore.Identity
@using MVC_APU_FlowerShop2023.Areas.Identity.Data

@inject SignInManager<MVC_APU_FlowerShop2023User> SignInManager
@inject UserManager<MVC_APU_FlowerShop2023User> UserManager

<ul class="navbar-nav">
@if (SignInManager.IsSignedIn(User))
{
    <li class="nav-item">
        <a id="addflower" class="nav-link text-dark" asp-action="Index" asp-controller="Flowers">View Flower List</a>
    </li>
    <li class="nav-item">
        <a id="manage" class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Hello @UserManager.GetUserName(User)!</a>
    </li>
    <li class="nav-item">
        <form id="logoutForm" class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("Index", "Home", new { area
            <button id="logout" type="submit" class="nav-link btn btn-link text-dark border-0">Logout</button>
        </form>
    </li>
}
else
{
    <li class="nav-item">
        <a class="nav-link text-dark" id="register" asp-area="Identity" asp-page="/Account/Register">Register</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" id="login" asp-area="Identity" asp-page="/Account/Login">Login</a>
    </li>
}
</ul>
```

22. Now, once every step for creating and adding the data is done, we can start testing our application now.


Home Page - MVC_APU_FlowerS

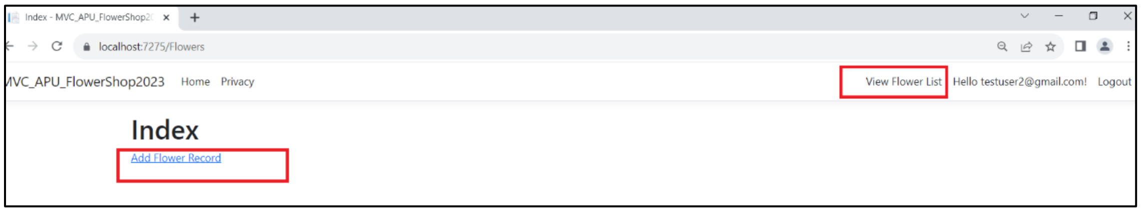
localhost:7275

MVC_APU_FlowerShop2023 Home Privacy View Flower List Hello testuser2@gmail.com! Logout

Welcome

Learn about building Web apps with ASP.NET Core.

© 2023 - MVC_APU_FlowerShop2023 - Privacy

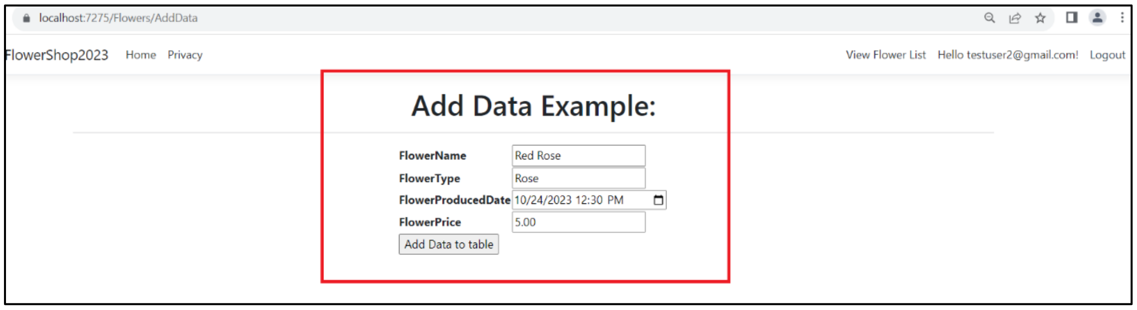

Index - MVC_APU_FlowerShop2

localhost:7275/Flowers

MVC_APU_FlowerShop2023 Home Privacy View Flower List Hello testuser2@gmail.com! Logout

Index

Add Flower Record


localhost:7275/Flowers/AddData

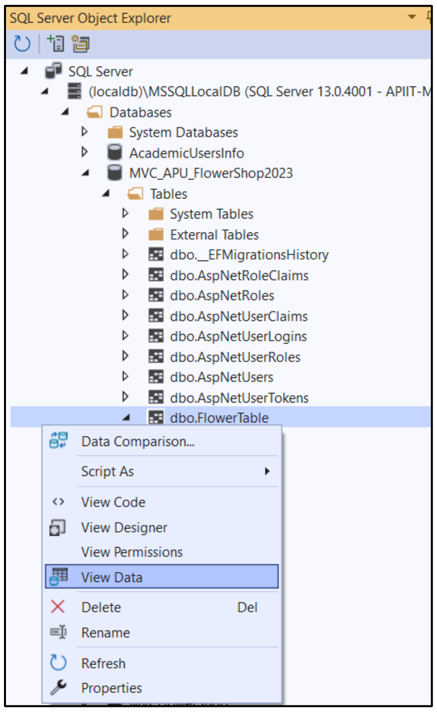
FlowerShop2023 Home Privacy View Flower List Hello testuser2@gmail.com! Logout

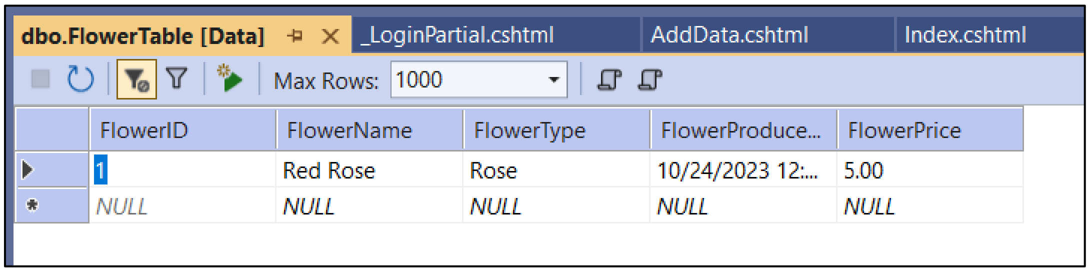
Add Data Example:

FlowerName	Red Rose
FlowerType	Rose
FlowerProducedDate	10/24/2023 12:30 PM
FlowerPrice	5.00

Add Data to table

23. Once added the data, let's look at the database table. If you successfully add the data, the data will appear in the Flower table now.





	FlowerID	FlowerName	FlowerType	FlowerProduce...	FlowerPrice
▶	1	Red Rose	Rose	10/24/2023 12:...	5.00
*	NULL	NULL	NULL	NULL	NULL

Lab 3.2. View Data from the tables of SQL Server with ASP .Net Core.

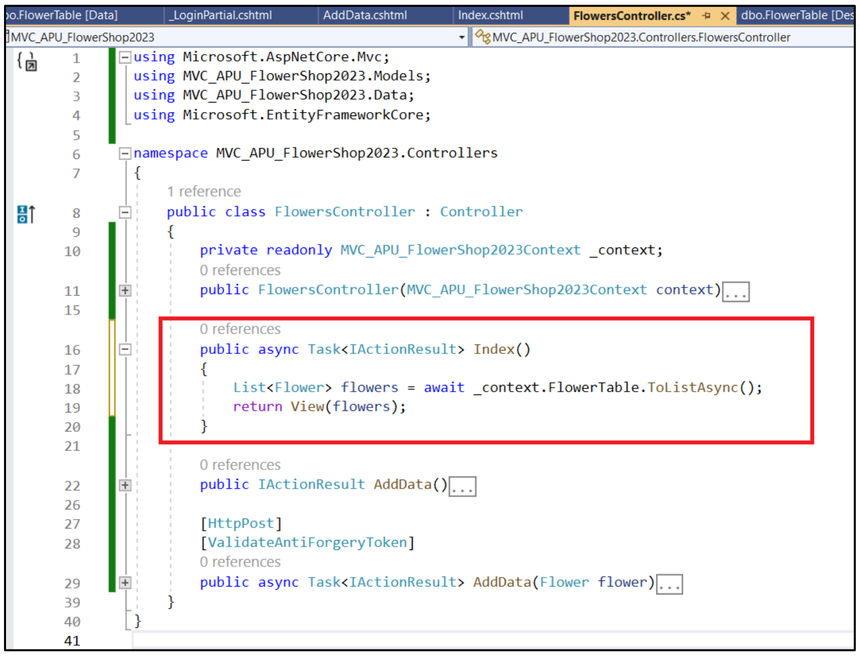
This tutorial will teach the student how to view data from a table in the SQL database.

e. View Data from the Flower Table

(Estimation of Total Time Used: 20 minutes)

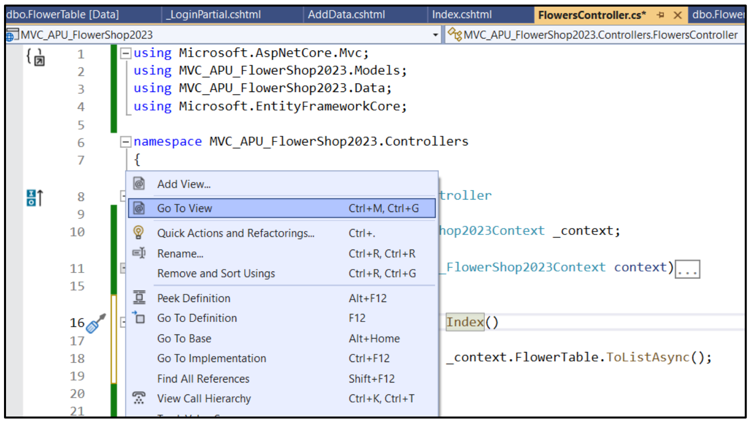
1. Now, go back to **FlowersController.cs** again.
2. Modify the **Index()** function by following the below code.

```
public async Task<IActionResult> Index()
{
    List<Flower> flowers = await _context.FlowerTable.ToListAsync();
    return View(flowers);
}
```



```
using Microsoft.AspNetCore.Mvc;
using MVC_APU_FlowerShop2023.Models;
using MVC_APU_FlowerShop2023.Data;
using Microsoft.EntityFrameworkCore;

namespace MVC_APU_FlowerShop2023.Controllers
{
    1 reference
    public class FlowersController : Controller
    {
        private readonly MVC_APU_FlowerShop2023Context _context;
        0 references
        public FlowersController(MVC_APU_FlowerShop2023Context context)...

        0 references
        public async Task<IActionResult> Index()
        {
            List<Flower> flowers = await _context.FlowerTable.ToListAsync();
            return View(flowers);
        }

        0 references
        public IActionResult AddData()...

        [HttpPost]
        [ValidateAntiForgeryToken]
        0 references
        public async Task<IActionResult> AddData(Flower flower)...
    }
}
```

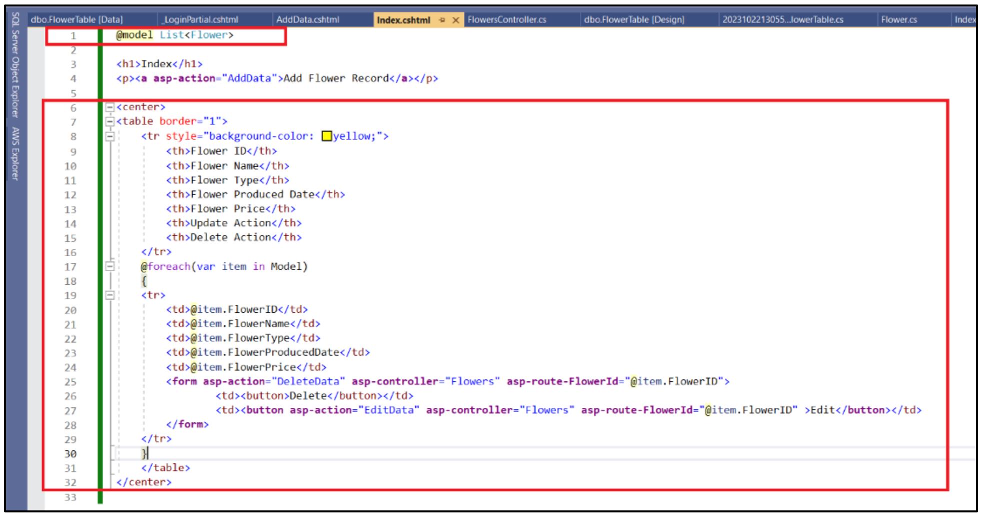
3. Now, go back to the **Index()** function by right clicking on the **Index()** function.


dbo.FlowerTable [Data]
_LoginPartial.cshtml
AddData.cshtml
Index.cshtml
FlowersController.cs*
dbo.Flowe
MVC_APU_FlowerShop2023
MVC_APU_FlowerShop2023.Controllers.FlowersController
Add View...
Go To View Ctrl+M, Ctrl+G
Quick Actions and Refactorings... Ctrl+.
Rename... Ctrl+R, Ctrl+R
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
Go To Base Alt+Home
Go To Implementation Ctrl+F12
Find All References Shift+F12
View Call Hierarchy Ctrl+K, Ctrl+T

```
using Microsoft.AspNetCore.Mvc;
using MVC_APU_FlowerShop2023.Models;
using MVC_APU_FlowerShop2023.Data;
using Microsoft.EntityFrameworkCore;

namespace MVC_APU_FlowerShop2023.Controllers
{
                                      troller

                                      hop2023Context _context;

                                       _FlowerShop2023Context context)[...]

                                       Index()

                                       _context.FlowerTable.ToListAsync();
```

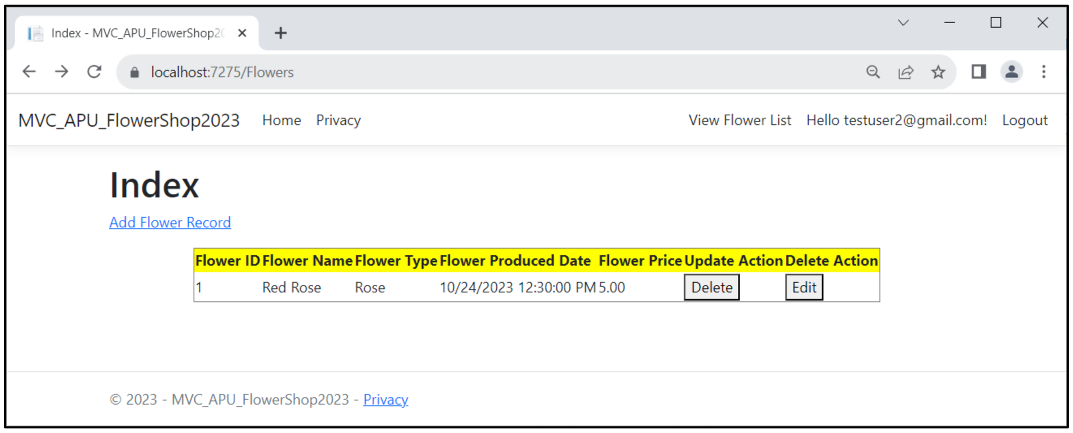
4. Modify the **Index.cshtml** file with the below code so that it can display the current data from the Flower table.

```
@model List<Flower>

<center>
<table border="1">
    <tr style="background-color: yellow;">
        <th>Flower ID</th>
        <th>Flower Name</th>
        <th>Flower Type</th>
        <th>Flower Produced Date</th>
        <th>Flower Price</th>
        <th>Update Action</th>
        <th>Delete Action</th>
    </tr>
    @foreach(var item in Model)
    {
    <tr>
        <td>@item.FlowerID</td>
        <td>@item.FlowerName</td>
        <td>@item.FlowerType</td>
        <td>@item.FlowerProducedDate</td>
        <td>@item.FlowerPrice</td>
        <form asp-action="DeleteData" asp-controller="Flowers" asp-route-
FlowerId="@item.FlowerID">
                <td><button>Delete</button></td>
                <td><button asp-action="EditData" asp-controller="Flowers" asp-route-
FlowerId="@item.FlowerID" >Edit</button></td>
        </form>
    </tr>
    }
    </table>
</center>
```



```
@model List<Flower>

<h1>Index</h1>
<p><a asp-action="AddData">Add Flower Record</a></p>

<center>
<table border="1">
    <tr style="background-color: yellow;">
        <th>Flower ID</th>
        <th>Flower Name</th>
        <th>Flower Type</th>
        <th>Flower Produced Date</th>
        <th>Flower Price</th>
        <th>Update Action</th>
        <th>Delete Action</th>
    </tr>
    @foreach(var item in Model)
    {
    <tr>
        <td>@item.FlowerID</td>
        <td>@item.FlowerName</td>
        <td>@item.FlowerType</td>
        <td>@item.FlowerProducedDate</td>
        <td>@item.FlowerPrice</td>
        <form asp-action="DeleteData" asp-controller="Flowers" asp-route-FlowerId="@item.FlowerID">
                <td><button>Delete</button></td>
                <td><button asp-action="EditData" asp-controller="Flowers" asp-route-FlowerId="@item.FlowerID" >Edit</button></td>
        </form>
    </tr>
    }
    </table>
</center>
```

5. Now, click on the **Start Without Debugging** button and restart the website again. You should see the data displayed in the current **Index.cshtml** in the Flower Folder.



Index

Add Flower Record

Flower ID	Flower Name	Flower Type	Flower Produced Date	Flower Price	Update Action	Delete Action
1	Red Rose	Rose	10/24/2023 12:30:00 PM	5.00	Delete	Edit

© 2023 - MVC_APU_FlowerShop2023 - Privacy

Summary:

In this tutorial, we have learnt how to create a new table and add data into the new table through the existing ASP .Net Core project. Besides, in the next tutorial, we have also learnt how to read records from the table through the existing ASP.NET Core project.

In the next tutorial, we will learn how to update and delete records from / to the table using the existing ASP .NET Core project.