

Lab 3 – Managing data in SQL Server with ASP .Net Core: Part 2

Estimated usage time: 1 Hour

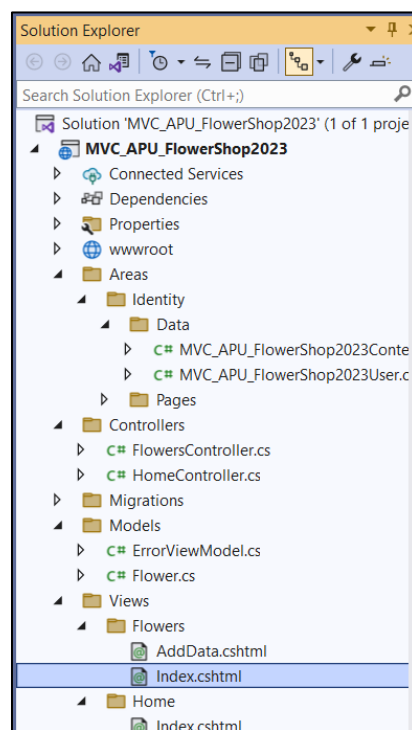
Lab 3.3. Update data in SQL Server through ASP .Net Core.

This tutorial will teach the student how to update data in the SQL database.


a. Create a page for updating data in existing Identity DB

(Estimation of Total Time Used: 30 minutes)

1. Continued from the **Lab 3.1** project.
2. Go to **Solution Explorer** > Go to **Views** Folder > Go to **Flowers** Folder > Double clicks on the **Index.cshtml** File.



3. Copy the function name from the button asp-action = "" (as shown in the red highlighted below).

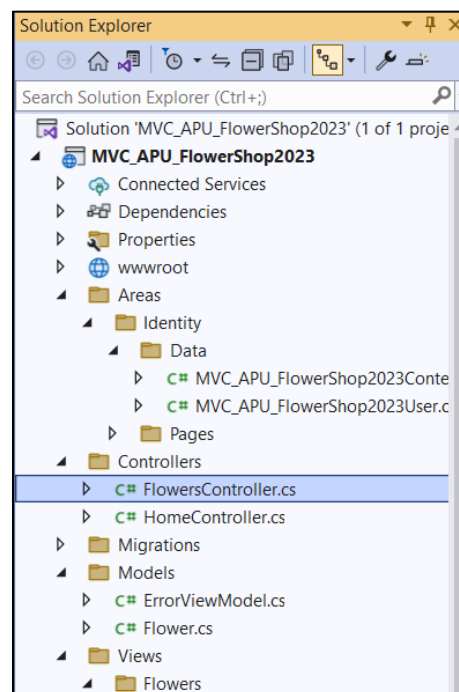


```

1  @model List<Flower>
2
3  <h1>Index</h1>
4  <p><a asp-action="AddData">Add Flower Record</a></p>
5
6  <center>
7  <table border="1">
8      <tr style="background-color: yellow;">
9          <th>Flower ID</th>
10         <th>Flower Name</th>
11         <th>Flower Type</th>
12         <th>Flower Produced Date</th>
13         <th>Flower Price</th>
14         <th>Update Action</th>
15         <th>Delete Action</th>
16     </tr>
17     @foreach(var item in Model)
18     {
19         <tr>
20             <td>@item.FlowerID</td>
21             <td>@item.FlowerName</td>
22             <td>@item.FlowerType</td>
23             <td>@item.FlowerProducedDate</td>
24             <td>@item.FlowerPrice</td>
25             <form asp-action="DeleteData" asp-controller="Flowers" asp-route-FlowerId="@item.FlowerID">
26                 <button>Delete</button></td>
27                 <td><button asp-action="EditData" asp-controller="Flowers" asp-route-FlowerId="@item.FlowerID">Edit</button></td>
28             </form>
29         </tr>
30     }
31 </table>
32 </center>
33

```

4. Now, go to **Solution Explorer** > go to **Controllers** Folder > Double clicks on the **FlowersController.cs** File.



5. Add the function with name of “**EditData()**” in the **FlowersController.cs**.

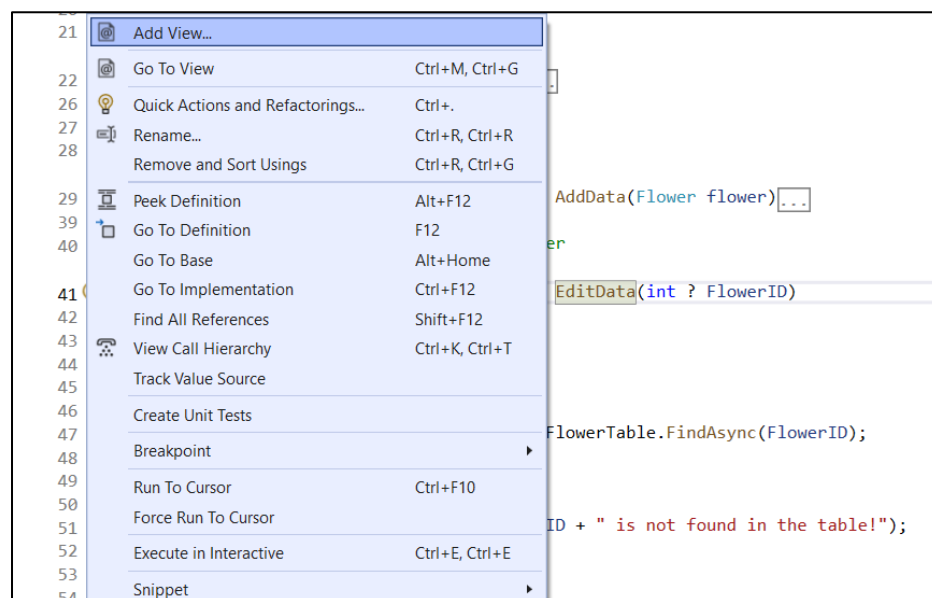
```
//load an update page for the user
0 references
public async Task<IActionResult> EditData(int ? FlowerID)
{
    if(FlowerID == null)
    {
        return NotFound();
    }
    var flower = await _context.FlowerTable.FindAsync(FlowerID);

    if(flower == null)
    {
        return BadRequest(FlowerID + " is not found in the table!");
    }
    return View(flower);
}
```

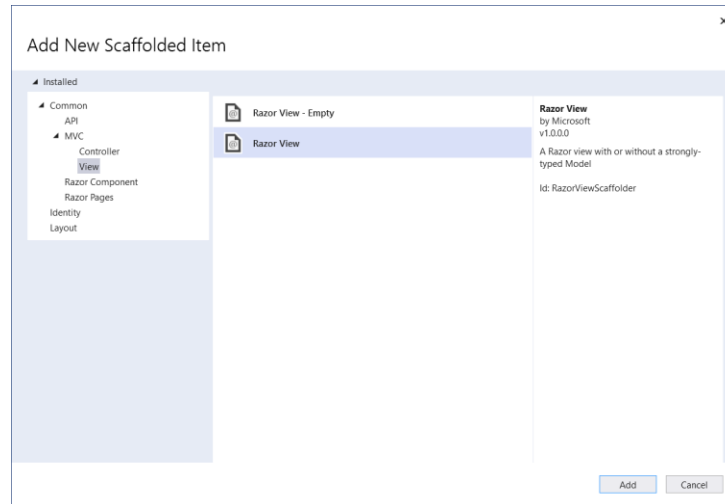
```
//load an update page for the user
public async Task<IActionResult> EditData(int ? FlowerID)
{
    if(FlowerID == null)
    {
        return NotFound();
    }
    var flower = await _context.FlowerTable.FindAsync(FlowerID);

    if(flower == null)
    {
        return BadRequest(FlowerID + " is not found in the table!");
    }
    return View(flower);
}
```

6. Now, highlight the `EditData()` function and right click on the highlighted function. Select **Add View** and generate a new page for this function.



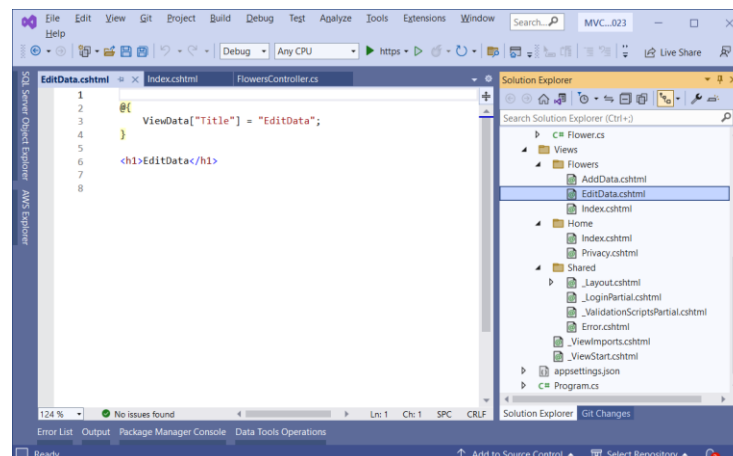
7. In the **Add New Scaffolded Item** dialog, select **Razor View** and press button **Add**.



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



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



10. Now, in the **EditData.cshtml**, add the below sentence to display the current existing data in the page.

```
@model MVC_APU_FlowerShop2023.Models.Flower

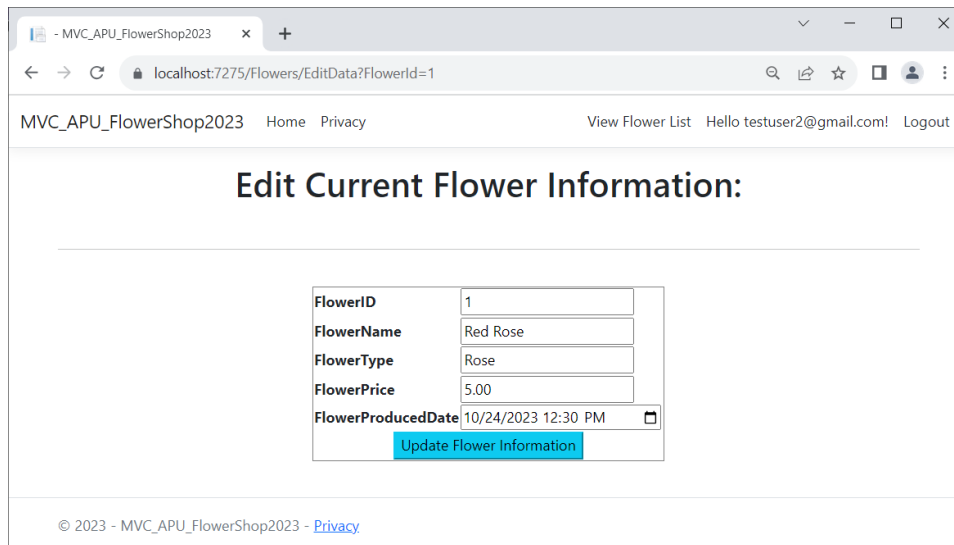
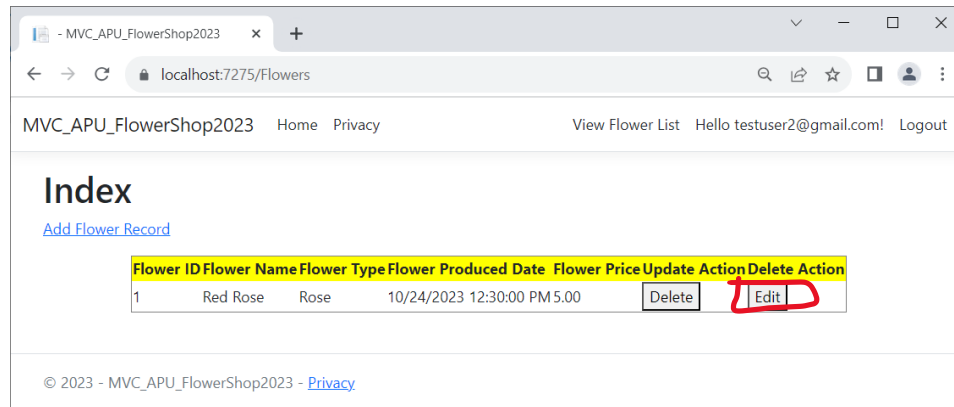
<center>
  <h1>Edit Current Flower Information:</h1>
  <br /><hr /><br />
  <form method="post" asp-action="UpdateData" asp-controller="Flowers">
    <table border='1'>
      <tr>
        <th><label asp-for="FlowerID"></label></th>
        <td><input asp-for="FlowerID" readonly /></td>
      </tr>
      <tr>
        <th><label asp-for="FlowerName"></label></th>
        <td><input asp-for="FlowerName" /></td>
        <td><span asp-validation-for="FlowerName"></span></td>
      </tr>
      <tr>
        <th><label asp-for="FlowerType"></label></th>
        <td><input asp-for="FlowerType" /></td>
        <td><span asp-validation-for="FlowerType"></span></td>
      </tr>
      <tr>
        <th><label asp-for="FlowerPrice"></label></th>
        <td><input asp-for="FlowerPrice" /></td>
        <td><span asp-validation-for="FlowerPrice"></span></td>
      </tr>
      <tr>
        <th><label asp-for="FlowerProducedDate"></label></th>
        <td><input asp-for="FlowerProducedDate" /></td>
        <td><span asp-validation-for="FlowerProducedDate"></span></td>
      </tr>
      <tr>
        <td colspan="3" style="text-align:center">
          <button type="submit" class="btn-info">Update Flower
Information</button>
        </td>
      </tr>
    </table>
  </form>
</center>
```

```

EditData.cshtml | Index.cshtml | FlowersController.cs
1  @model MVC_APU_FlowerShop2023.Models.Flower
2
3  <center>
4      <h1>Edit Current Flower Information:</h1>
5      <br /><br />
6      <form method="post" asp-action="UpdateData" asp-controller="Flowers">
7          <table border='1'>
8              <tr>
9                  <th><label asp-for="FlowerID"></label></th>
10                 <td><input asp-for="FlowerID" readonly /></td>
11             </tr>
12             <tr>
13                 <th><label asp-for="FlowerName"></label></th>
14                 <td><input asp-for="FlowerName" /></td>
15                 <td><span asp-validation-for="FlowerName"></span></td>
16             </tr>
17             <tr>
18                 <th><label asp-for="FlowerType"></label></th>
19                 <td><input asp-for="FlowerType" /></td>
20                 <td><span asp-validation-for="FlowerType"></span></td>
21             </tr>
22             <tr>
23                 <th><label asp-for="FlowerPrice"></label></th>
24                 <td><input asp-for="FlowerPrice" /></td>
25                 <td><span asp-validation-for="FlowerPrice"></span></td>
26             </tr>
27             <tr>
28                 <th><label asp-for="FlowerProducedDate"></label></th>
29                 <td><input asp-for="FlowerProducedDate" /></td>
30                 <td><span asp-validation-for="FlowerProducedDate"></span></td>
31             </tr>
32             <tr>
33                 <td colspan="3" style="text-align:center">
34                     <button type="submit" class="btn-info">Update Flower Information</button>
35                 </td>
36             </tr>
37         </table>
38     </form>
39 </center>

```

11. Now, click on the **Start Without Debugging** button, you will see the below page appear after you clicked on the **Edit** button in the **Index.cshtml** page.



12. To update the latest data in the Flower table, we need to **write a function named ProcessUpdateData** in the controller file to process the update action.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> UpdateData(Flower flower)
{
    try
    {
        if (ModelState.IsValid)
        {
            _context.FlowerTable.Update(flower);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index", "Flowers");
        }
        return View("EditData", flower);
    }
    catch(Exception ex)
    {
        return BadRequest("Error: " + ex.Message);
    }
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> UpdateData(Flower flower)
{
    try
    {
        if (ModelState.IsValid)
        {
            _context.FlowerTable.Update(flower);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index", "Flowers");
        }
        return View("EditData", flower);
    }
    catch(Exception ex)
    {
        return BadRequest("Error: " + ex.Message);
    }
}
```


13. To test the function, we can change the current flower information and see if it changes in the database.

Edit Current Flower Information:

FlowerID	1
FlowerName	Blue Rose
FlowerType	Rose
FlowerPrice	3.99
FlowerProducedDate	10/26/2023 02:30 PM

Update F

October 2023

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

Clear Today

12 30 PM

01 31 AM

02 32

03 33

04 34

05 35

06 36

Index

[Add Flower Record](#)

Flower ID	Flower Name	Flower Type	Flower Produced Date	Flower Price	Update Action	Delete Action
1	Blue Rose	Rose	10/26/2023 2:30:00 PM	3.99	Delete	Edit

Lab 3.4. Delete Data from the tables of SQL Server with ASP .Net Core.

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

b. Delete Data from the Flower Table

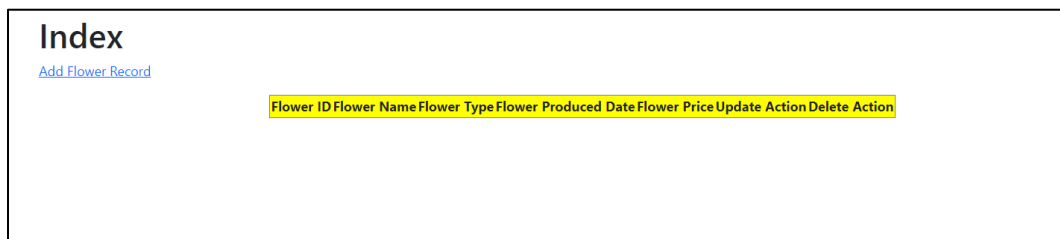
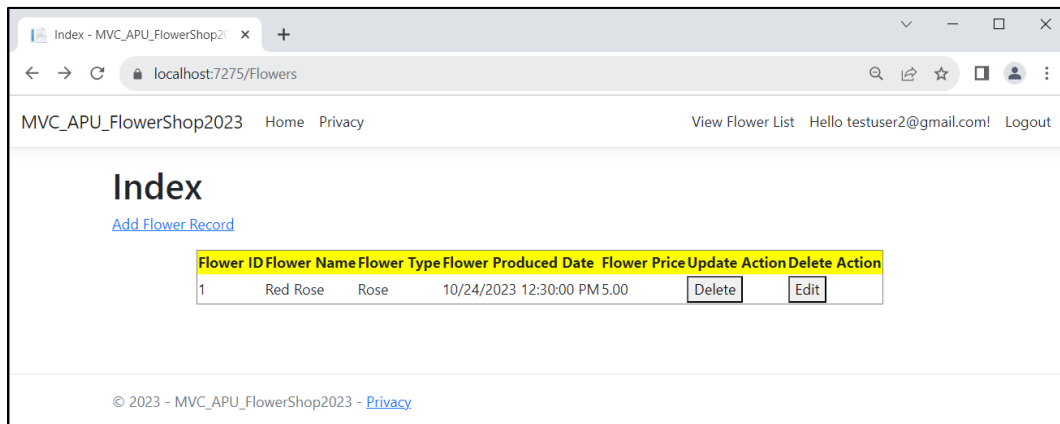
(Estimation of Total Time Used: 30 minutes)

1. Now, go back to **FlowersController.cs** again.
2. Add a new **DeleteData()** function by following the below code.

```
//delete data from the page
public async Task<IActionResult> DeleteData(int ? FlowerID)
{
    if(FlowerID ==null)
    {
        return NotFound();
    }
    var flower = await _context.FlowerTable.FindAsync(FlowerID);
    if(flower == null)
    {
        return BadRequest(FlowerID + " is not found in the list!");
    }
    _context.FlowerTable.Remove(flower);
    await _context.SaveChangesAsync();
    return RedirectToAction("Index", "Flowers");
}
```

```
//delete data from the page
0 references
public async Task<IActionResult> DeleteData(int ? FlowerID)
{
    if(FlowerID ==null)
    {
        return NotFound();
    }
    var flower = await _context.FlowerTable.FindAsync(FlowerID);
    if(flower == null)
    {
        return BadRequest(FlowerID + " is not found in the list!");
    }
    _context.FlowerTable.Remove(flower);
    await _context.SaveChangesAsync();
    return RedirectToAction("Index", "Flowers");
}
```

- Now, click on the **Start Without Debugging** button and restart the website again.
- To test the function, we can click on the **Delete** button and see if the selected flower information is deleted from the database.



Summary:

In this tutorial, we have learnt how to edit data in database through the existing ASP .Net Core project. Besides, in the next tutorial, we have also learnt how to delete record from the table through the existing ASP.NET Core project.

In the next tutorial, we will learn how to migrate the tables from local database to Amazon Relational Database Services through Visual Studio.