

ASP.NET Core MVC 6.0

Education and Training Solutions 2022

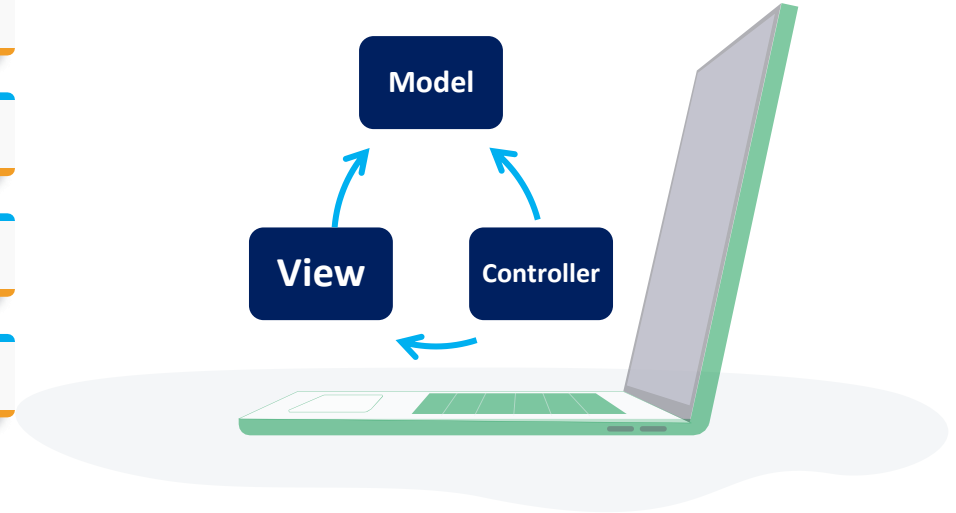


1 Generate ASP.NET Core MVC Model Classes

2 Route

3 CRUD Operation

4 Controller and View



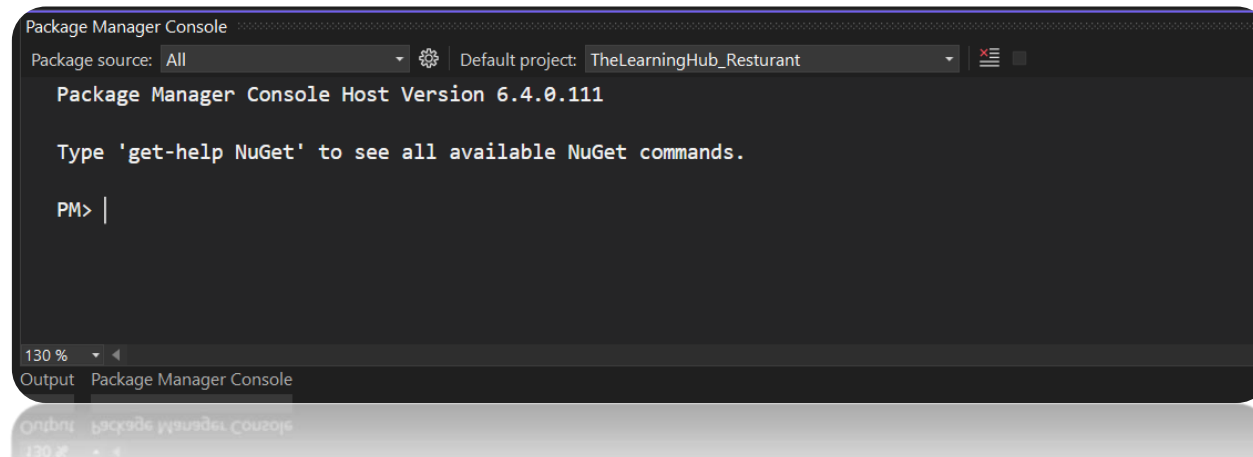
Generate ASP.NET Core MVC Model Classes

To Generate ASP.NET Core MVC Model Classes Go to: Tools => NuGet Package Manager => Package Manager Console => Paste the following command:

Command:

```
Scaffold-DbContext "Data  
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DAT  
A=(SID=x)))";User Id=****;Password=****;" Oracle.EntityFrameworkCore -outputdir  
Models
```

****: put your username and password in the SQL Oracle server



To generate specific tables from the database as models, use this command:

```
Scaffold-DbContext "Data  
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))(CONNECT_DATA=(SID=x)))";User Id=****;Password=****;" Oracle.EntityFrameworkCore -tables  
CATEGORY,PRODUCT,USERLOGIN,CUSTOMER,PRODUCTCUSTOMER,ROLE -FORCE -outputdir Models
```

Note that all table names should be in uppercase format

In appsettings.json:

```
"ConnectionStrings": {  
  "DefaultConnection": "Data  
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=15  
21))(CONNECT_DATA=(SID=x)))User Id=***;Password=***;Persist  
Security Info=True;"  
},
```

In Program.cs:

```
builder.Services.AddDbContext<ModelContext>(options =>  
options.UseOracle(builder.Configuration.GetConnectionString(  
"DefaultConnection")));
```

0 references

```
public static void Main(string[] args)  
{  
    var builder = WebApplication.CreateBuilder(args);  
  
    builder.Services.AddDbContext<ModelContext>(options =>  
options.UseOracle(builder.Configuration.GetConnectionString("DefaultConnection")));
```

```
options.UseOracle(builder.Configuration.GetConnectionString("DefaultConnection"));
```



Route

- Route defines the URL pattern and handler information, Routing maps the URL to a physical file or class (controller class in MVC).
- Configure routes in program.cs :

```
app.MapControllerRoute(  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

Route Name

URL Pattern

The URL pattern is considered only after the domain name part in the URL.

➤ Would look like localhost:xxxx/{controller}/{action}/{id?}.

? : mean it is optional

Example:

`https://localhost:44392/Home/Index`

Domain Name Controller Action Method

`https://localhost:44392/student/edit/id` Id parameter value

CRUD Operations

Overview of CRUD Operation:

CRUD is an acronym that comes from the computer programming world. It refers to the four functions that are represented necessary to implement a persistent storage application.

CRUD: Create, Read, Update, and Delete.

Controller and View

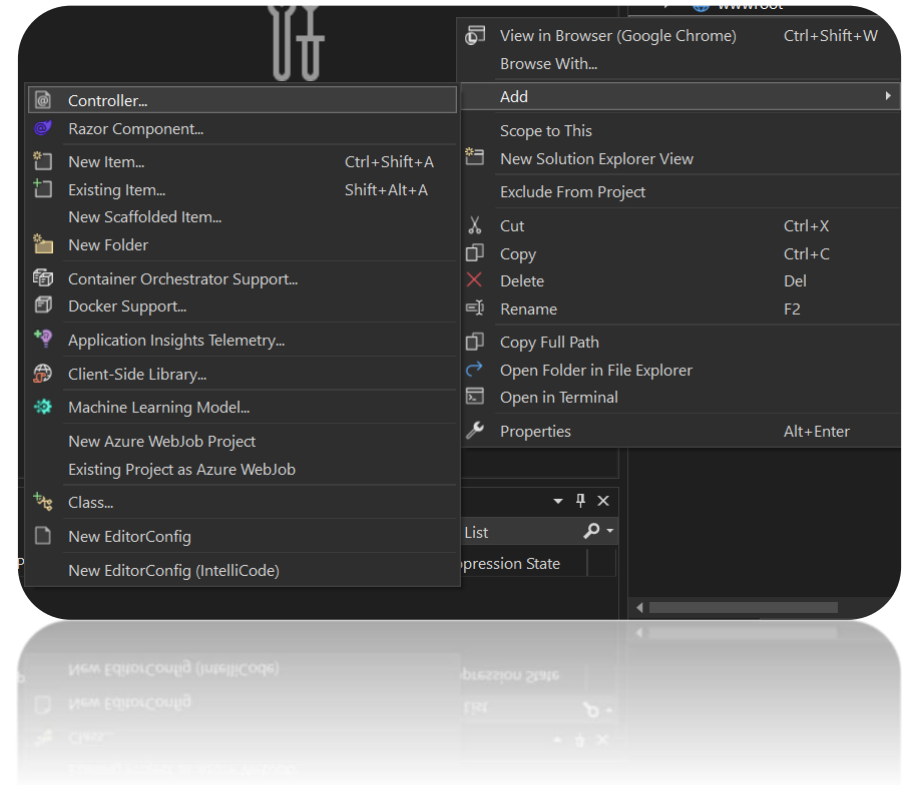
- **The Controller** Handles incoming browser requests, retrieves necessary model data, and returns appropriate responses.
- Contains public methods called Action methods.
- Every **controller** class name must end with the word "**Controller**".

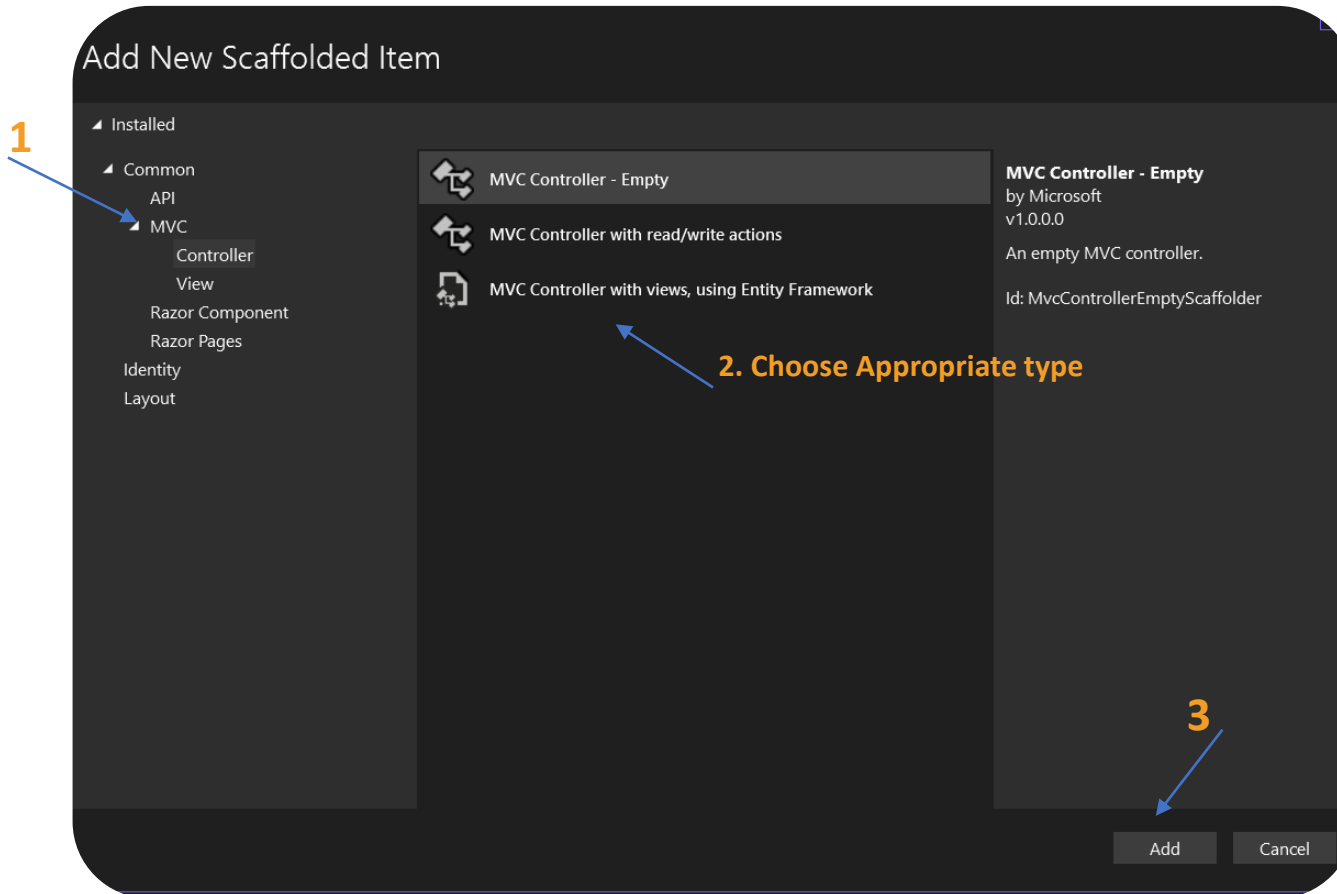


- Action method must be **public**. It cannot be a private or protected, static method.
- ActionResult is a base class of all the result types that return from the Action method.
- The base Controller class contains methods that return the appropriate result type.
- **ActionVerbs**: Handle different type of http request { HttpGet, HttpPost, HttpPut}.

Add new Controller

In Visual Studio, right-click on the Controller folder -> select Add -> click on Controller.





- Choose which **model class** you need to create the controller for, and the **data context** class.

Add MVC Controller with views, using Entity Framework

Model class:

Data context class:

Views

☒ Generate views

☐ Reference script libraries

☐ Use a layout page

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

Controller name:

Controller name:

Category controller

Base controller class

```
public class CategoriesController : Controller
{
    private readonly ModelContext _context;
    private readonly IWebHostEnvironment _webHostEnviroment;

    0 references
    public CategoriesController(ModelContext context, IWebHostEnvironment webHostEnviroment)
    {
        _context = context;
        _webHostEnviroment = webHostEnviroment;
    }

    // GET: Categories
    3 references
    public async Task<IActionResult> Index()
    {
        return _context.Categories != null ?
            View(await _context.Categories.ToListAsync()) :
            Problem("Entity set 'ModelContext.Categories' is null.");
    }
}
```

Return type

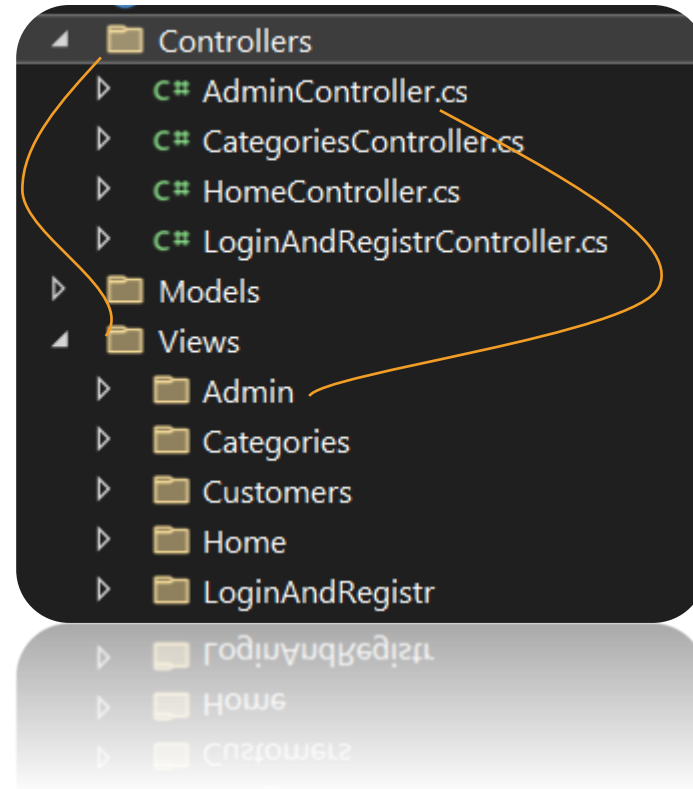
Action Method

View() defined in base controller class

```
    Problem("Entity set 'ModelContext.Categories' is null.");
    View(await _context.Categories.ToListAsync()) :
    return _context.Categories != null ?
```

The View

- View is used to display data using the model class object.
- The Shared folder contains views, layout views, and partial views, which will be shared among multiple controllers.
- A controller can render one or more views.



Each controller has a view
with the same name.

- Razor View: allows you to write a mix between HTML tags and C#|VB code (viewname.**vbhtml** |viewname. **cshtml**).
- Razor syntax is [Compact](#), [Easy to Learn](#), and [IntelliSense](#).
- Razor Inline expression using @ with C# code, Multi-statement Code block using @{ } with C# code.

Razor Syntax





Html Tag

Html Helper

Razor Syntax

```
index.cshtml*  CategoriesController.cs  appsettings.json
1  @model IEnumerable<TheLearningHub_Resturant.Models.Category>
2  @{
3      ViewData["Title"] = "Index";
4      Layout = "~/Views/Shared/_AdminLayout.cshtml";
5  }
6  <h1>Index</h1>
7
8  <p>
9      <a asp-action="Create">Create New</a>
10  </p>
11  <table class="table">
12      <thead>
13          <tr>
14              <th>
15                  @Html.DisplayNameFor(model => model.CategoryName)
16              </th>
17              <th>
18                  @Html.DisplayNameFor(model => model.ImagePath)
19              </th>
20              <th></th>
21          </tr>
22      </thead>
23      <tbody>
24          @foreach (var item in Model) {
25              <tr>
26                  <td>
27                      @Html.DisplayNameFor(modelItem => item.CategoryName)
28                  </td>
29              </tr>
30          }
31      </tbody>
32  </table>
33  @foreach (var item in Model) {
```


Result:

Index		
Create New		
CategoryName	ImagePath	
Fast Food		Edit Details Delete
Salad		Edit Details Delete
Main Meal		Edit Details Delete
Dessert		Edit Details Delete

- To define a List of objects, use [IEnumerable](#), usually using it in the index view page to display all data in the table.

Example:

```
@model IEnumerable<TheLearningHub_Resturant.Models.Category>
```

- In Edit, Details, delete and create a view page which returns the data from one object (row) declare :

```
@model TheLearningHub_Resturant.Models.Category
```

Create a Model-View-Controller for all tables in the database:

1. Category.
2. Product.
3. Customer.
4. UserLogin.
5. Roles



Break



Thank You

