go deploy

Screenshots

Module 1: Exploring ASP.NET Core

Lab: Exploring ASP.NET Core

Scenario

You are working as a junior developer at Adventure Works. You have been asked by a senior developer to investigate the possibility of creating a web-based ASP.NET Core application for your organization's customers, similar to the one that the senior developer has seen on the Internet. Such an application will promote a community of cyclists who use Adventure Works equipment, and the community members will be able to share their experiences. This initiative is intended to increase the popularity of Adventure Works Cycles, and thereby to increase their sales. You have been asked to begin the planning of the application. You have also been asked to examine programming models available to ASP.NET Core developers. To do this, you need to create basic web applications using three different models: Razor Pages, Web API, and MVC.

Exercise 1: Exploring a Razor Pages Application

Scenario

In this exercise, you will create a simple Razor Pages application, and explore its structure.

The main tasks for this exercise are as follows:

- Creating a Razor Pages application
- Exploring the application structure
- Adding simple functionality
- Running the application

Task 1: Creating a Razor Pages application

- [ ] 1. Start **Visual Studio**.
- [ ] 2. In the **Start Page - Microsoft Visual Studio** window, on the **File** menu, point to **New**, and then click **Project**.
- [ ] 3. In the **Create a new project** dialog box, in the languages dropdown, ensure that **C#** is selected.
- [ ] 4. In the **Create a new project** dialog box, choose **ASP.NET Core Web App** and click **Next**.

☐ 6. In the **Additional information** dialog box, select **.NET 6.0 (Long-term support)** from the dropdown at the top of the dialog box.

☐ 7. In the **Additional information** dialog box, ensure that the **Configure for HTTPS** check box is **not** selected, and leave the other settings as their default values.

☐ 8. Click on **Create**.

☐ 9. In the **ActorsRazorPages - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**, and wait for the **Microsoft Edge** browser window to launch.

> ⚠ **Note:** If prompted to trust the IIS Express SSL certificate, click **Yes**

☐ 10. In **Microsoft Edge**, in the navigation bar, click **Privacy** to review its content.

☐ 11. Close **Microsoft Edge**.

Task 2: Explore the application structure

☐ 1. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Pages**, and then click **_ViewStart.cshtml**.

☐ 2. In the **_ViewStart.cshtml** code window, note the value of Layout is **"_Layout"**.

> ⚠ **Note:** This indicates that all the files inside the **Pages** folder use the same layout file, **~/Pages/Shared/_Layout.cshtml**.

☐ 3. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, click **Privacy.cshtml**.

☐ 4. In the **Privacy.cshtml** code window, examine the Razor code, and verify that there are no links to **.css** files.

☐ 5. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, expand **Shared**, then click **_Layout.cshtml**.

☐ 6. In the **_Layout.cshtml** code window, in the **head** element, note that there is a link to **~/css/site.css**.

☐ 7. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **ActorsRazorPages**, expand **wwwroot**, expand **css**, and then click **site.css**

> ⚠ **Note:** This is the CSS **style sheet** file that is applied in **_Layout.cshtml**.

Pages, point to **Add**, and then click **Razor Page...**.

- [ ] 2. In the **Add New Scaffolded Item** dialog box, click **Razor Page - Empty**, and then click **Add**.

- [ ] 3. In the **Add New Item - ActorsRazorPages** dialog box, in the **Name** text box, type TestPage.cshtml, and then click **Add**.

- [ ] 4. In the **ActorsRazorPages - Microsoft Visual Studio** window, in the **TestPage.cshtml** code window, replace the content below **@model** line with the following code:

```
@{
    ViewData["Title"] = "Test Page";
}

<h1>@ViewData["Title"]</h1>
<h2>This is a Test Page</h2>
```

- [ ] 5. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Pages**, expand **Shared** and then click **_Layout.cshtml**.

- [ ] 6. In the **_Layout.cshtml** code window, locate the following code:

```
<li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-page="/Privacy">Privacy</a>
</li>
```

- [ ] 7. Place the cursor after the located code, press Enter, and then type the following code:

```
<li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-page="/TestPage">TestPage</a>
</li>
```

- [ ] 8. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **ActorsRazorPages**, point to **Add**, and then click **New Folder**.

- [ ] 9. In the **NewFolder** box, type Models, and then press Enter.

- [ ] 10. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **Class...**.

- [ ] 11. In the **Add New Item - ActorsRazorPages** dialog box, in the **Name** text box, type Actor.cs, and then click **Add**.

- [ ] 12. In the **Actor.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
    public string LastName { get; set; }
    public string KnownFor { get; set; }
    public bool OscarWinner { get; set; }
    public string ImageName { get; set; }
```

- [ ] 13. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **New Item...**.
- [ ] 14. In the **Add New Item - ActorsRazorPages** dialog box, click **Interface**.
- [ ] 15. In the **Add New Item - ActorsRazorPages** dialog box, in the **Name** text box, type **IData.cs**, and then click **Add**.
- [ ] 16. In the **IData** interface code block, check that the word **interface** is preceded by **public**, and add it if needed.
- [ ] 17. In the **IData** interface code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
List<Actor> ActorsList { get; set; }
List<Actor> ActorsInitializeData();
Actor GetActorById(int? id);
```

- [ ] 18. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then select **Existing Item...**.
- [ ] 19. In the dialog box navigate to **D:\Allfiles\Mod01\Labfiles\01_ActorsRazorPages_begin**, click **Data.cs**, and then click **Add**.

> **Note:** Examine the **Data.cs** class content. You will see warnings about non-nullable properties (this feature was recently added to the C# language). This will not prevent your code from running, but you can avoid the warnings. Right-click on the **ActorsRazorPages** project in **Solution Explorer**, and choose **Properties**. In the properties, under the **Build** tab, you will find a setting **Nullable**. Set this to **Disable** to suppress the warnings.

- [ ] 20. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **wwwroot**, point to **Add**, and then click **New Folder**.
- [ ] 21. In the **NewFolder** box, type **images**, and then press Enter.
- [ ] 22. Right-click on the **images** folder you just created, and point to **Add**, and then click **Existing Item...**.
- [ ] 23. In the dialog box, navigate to **D:\Allfiles\Mod01\Labfiles\01_ActorsRazorPages_begin\Images**, set the file type filter in the dialog to **All Files (*.*)**, select all the images, and then click **Add**.

go deploy

- [ ] 25. In the **NewFolder** box, type Actors, and then press Enter.
- [ ] 26. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, right-click **Actors**, point to **Add**, and then click **Razor Page...**.
- [ ] 27. In the **Add New Scaffolded Item** dialog box, click **Add**.
- [ ] 28. In the **Add New Item - ActorsRazorPages** dialog box, in the **Name** text box, type Index, and then click **Add**.
- [ ] 29. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Index.cshtml**, click **Index.cshtml.cs**, select the following code, and then press Delete.

```
public void OnGet()
{
}
```

- [ ] 30. In the **Index.cshtml.cs** code window, place the cursor at the end of the **using Microsoft.AspNetCore.Mvc.RazorPages** namespace code, press Enter, and then type the following code:

```
using ActorsRazorPages.Models;
```

- [ ] 31. In the **Index.cshtml.cs** code block, place the cursor after the second **{** (opening brace) sign, press **Enter**, and then type the following code:

```
private IData _data;

public IndexModel(IData data)
{
    _data = data;
}

public IList<Actor> Actors { get; set; }

public void OnGet()
{
    Actors = _data.ActorsInitializeData();
}
```

- [ ] 32. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, under **Actors**, click **Index.cshtml**.
- [ ] 33. In the **ActorsRazorPages - Microsoft Visual Studio** window, in the **Index.cshtml** code window, replace the content below the **@model** line with the following code:

```
}

<h2>Index</h2>

<table class="table">
      <thead>
            <tr>
                  <th>
                        @Html.DisplayNameFor(model => model.Actors[0].FirstName)
                  </th>
                  <th>
                        @Html.DisplayNameFor(model => model.Actors[0].LastName)
                  </th>
                  <th></th>
            </tr>
      </thead>
      <tbody>
            @foreach (var item in Model.Actors)
            {
            <tr>
                  <td>
                        @Html.DisplayFor(modelItem => item.FirstName)
                  </td>
                  <td>
                        @Html.DisplayFor(modelItem => item.LastName)
                  </td>
                  <td>
                        <a asp-page="./Details" asp-route-id="@item.Id">Details</a>
                  </td>
            </tr>
            }
      </tbody>
</table>
```

- [ ] 34. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, right-click **Actors**, point to **Add**, and then click **Existing Item...**.

- [ ] 35. In the dialog box, navigate to **D:\Allfiles\Mod01\Labfiles\01_ActorsRazorPages_begin\Pages**, set the file type filter in the dialog to **All Files (*.*)**, select **Details.cshtml.cs** and **Details.cshtml**, and then click **Add**.

  Note: Examine the **Details.cshtml.cs**, and the **Details.cshtml** files content.

- [ ] 36. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, click **Program.cs**.

- [ ] 37. Add the following code at the beginning of the file:

- [ ] 38. Place the cursor after the line **builder.Services.AddRazorPages();**, press Enter, and then type the following code:

```
builder.Services.AddSingleton<IData, Data>();
```

- [ ] 39. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, double-click **_Layout.cshtml**.

- [ ] 40. In the **_Layout.cshtml** code window, locate the following code:

```
<li>
      <a class="nav-link text-dark" asp-area="" asp-page="/TestPage">TestPage</a>
</li>
```

- [ ] 41. Place the cursor after the located code, press Enter, and then type the following code:

```
<li>
      <a class="nav-link text-dark" asp-area="" asp-page="/Actors/Index">Actors</a>
</li>
```

Task 4: Run the application

- [ ] 1. In the **ActorsRazorPages - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.

- [ ] 2. In the **ActorsRazorPages - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**.

- [ ] 3. View the **Test Page** and **Actors** pages you have added.

> ⚠ **Note**: The browser window displays the title **Test Page** and the text **"This is a Test Page"**.

> ⚠ **Note**: The browser window displays the **Index.cshtml** page under the **Actors** folder.

- [ ] 4. In the **Actors** window, click the **Details** link for either of the actors.

> ⚠ **Note**: The browser window displays the **Details.cshtml** page under the **Actors** folder.

- [ ] 5. Verify that the **site.css** file is used to apply styles to all the pages.

- [ ] 6. Close the **Microsoft Edge** window.

go deploy ☰

✓ **Results**: At the end of this exercise, you have built a simple but fully functional Razor Pages application in Visual Studio.

Exercise 2: Exploring a Web API Application

Scenario

In this exercise, you will create a simple Web API application, and explore its structure.

The main tasks for this exercise are as follows:

- Creating a Web API application
- Exploring the application structure
- Adding simple functionality
- Running the application

Task 1: Creating a Web API application

- [ ] 1. Start **Visual Studio**.
- [ ] 2. In the **Start Page - Microsoft Visual Studio** window, on the **File** menu, point to **New**, and then click **Project**.
- [ ] 3. In the **Create a new project** dialog box, in the languages dropdown, ensure that **C#** is selected.
- [ ] 4. In the **Create a new project** dialog box, choose **ASP.NET Core Web API** and click **Next**.
- [ ] 5. In the **Project Name** text box, type CakeStoreApi and click **Next**.
- [ ] 6. In the **Additional information** dialog box, select **.NET 6.0 (Long-term support)** from the dropdown at the top of the dialog box.
- [ ] 7. In the **Additional information** dialog box, ensure that the **Configure for HTTPS** check box is **not** selected, and leave the other settings as their default values.

Task 2: Explore the application structure

- [ ] 1. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Controllers**, and then click **WeatherForecastController.cs**.

> ⚠ **Note:** The **Get()** method returns an array of forecasts.

- [ ] 2. In the **CakeStoreApi - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without**

Note: The browser displays a Swagger page to show the API methods, parameters (if any), and responses. Swagger is a tool for displaying APIs that conform to the OpenAPI specification.

3. In **Microsoft Edge**, click on the button for the **Get** method to see that it takes no parameters and responds with a JSON object.

4. On the Swagger page, click on the **Try it out** button, and then click on **Execute**.

Note: The Swagger page shows the request and response. The response body is a JSON array of weather forecasts.

5. In **Microsoft Edge**, click **Close**.

Task 3: Add simple functionality

1. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **CakeStoreApi**, point to **Add**, and then click **New Folder**.

2. In the **NewFolder** text box, type Models, and then press Enter.

3. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **Class...**.

4. In the **Add New Item - CakeStoreApi** dialog box, in the **Name** text box, type CakeStore, and then click **Add**.

5. In the **CakeStore.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
public int Id { get; set; }
public string CakeType { get; set; }
public int Quantity { get; set; }
```

6. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **New Item...**.

7. In the **Add New Item - CakeStoreApi** dialog box, click **Interface**.

8. In the **Add New Item - CakeStoreApi** dialog box, in the **Name** text box, type IData, and then click **Add**.

9. In the **IData** interface code block, check that the word **interface** is preceded by **public**, and add it if needed.

go deploy

```
List<CakeStore> CakesList { get; set; }
List<CakeStore> CakesInitializeData();
CakeStore GetCakeById(int? id);
```

- [ ] 11. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and select **Existing Item...**.
- [ ] 12. In the dialog box, navigate to **D:\Allfiles\Mod01\Labfiles\02_CakeStoreApi_begin**, click **Data.cs**, and then click **Add**.

> **Note:** Examine the **Data.cs** content.

- [ ] 13. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Controllers**, point to **Add**, and then click **Controller**.
- [ ] 14. In the **Add New Scaffolded Item** page, choose **API** from the left-hand list tree, and then choose **API Controller - Empty**, and then click **Add**.
- [ ] 15. In the **Add New Item - CakeStoreApi** dialog box, in the **Name** text box, type CakeStoreApiController, and then click **Add**.
- [ ] 16. In the **CakeStoreApiController.cs** code block, ensure that the cursor is at the end of the **using Microsoft.AspNetCore.Mvc** namespace code, press Enter, and then type the following code:

```
using CakeStoreApi.Models;
```

- [ ] 17. In the **CakeStoreApiController.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
        public CakeStoreApiController(IData data)
        {
                _data = data;
        }

        [HttpGet("/api/CakeStore")]
        public ActionResult<List<CakeStore>> GetAll()
        {
                return _data.CakesInitializeData();
        }

        [HttpGet("/api/CakeStore/{id}", Name = "GetCake")]
        public ActionResult<CakeStore> GetById(int? id)
        {
                var item = _data.GetCakeById(id);
                if (item == null)
                {
                        return NotFound();
                }
                return new ObjectResult(item);
        }
```

Note: The content inside the **httpGet** attributes indicates the URL that the user should write to get to the relevant action.

18. In the **CakeStoreApi - Microsoft Visual Studio** window, in **Solution Explorer**, click **Program.cs**.

19. Add the following code at the beginning of the file:

```
using CakeStoreApi.Models;
```

20. Place the cursor after the line **builder.Services.AddSwaggerGen();**, press Enter, and then type the following code:

```
builder.Services.AddSingleton<IData, Data>();
```

Task 4: Run the application

1. In the **CakeStoreApi - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.

2. In the **CakeStoreApi - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**.

3. In **Microsoft Edge**, click on the button for the CakeStoreApi **Get** method to see that it takes no parameters and responds with a JSON object.

⚠ **Note:** The Swagger page shows the request and response. The response body is a JSON array containing the cake inventory. Note that there is a second version of the get request that takes an integer parameter.

- [ ] 5. In **Microsoft Edge**, click on the button for the CakeStoreApi **Get** method to collapse it and click on the second **Get** method which takes an additional **id** parameter and responds with a JSON object.
- [ ] 6. Click on the **Try it out** button, enter a value of 2 for the id, and then click on **Execute**.

⚠ **Note:** The Swagger page shows the response is a single item from the cake inventory corresponding to id=2, which is the Strawberry cake.

- [ ] 7. In **Microsoft Edge**, note the web address in the address bar, which should be **localhost:[some_port]/swagger/index.html**. Change this address to **localhost:[some_port]/api/CakeStore/2** (replacing **[some_port]** with the actual port number) and hit the return key.

⚠ **Note:** The browser displays the returned **JSON** for the Strawberry cake inventory item. You have called the API directly using the browser. Try using the address **localhost:[some_port]/api/CakeStore/** (i.e. without the id).

- [ ] 8. In **Microsoft Edge**, click **Close**.
- [ ] 9. In the **CakeStoreApi - Microsoft Visual Studio** window, on the **File** menu, click **Exit**.

✓ **Results**: At the end of this exercise, you have built a simple Web API application using ASP.NET Core in Visual Studio.

Exercise 3: Exploring an MVC Application

In this exercise, you will create a simple MVC application, and explore its structure.

The main tasks for this exercise are as follows:

- Creating an MVC application
- Explore the application structure
- Add simple functionality
- Run the application

Task 1: Creating an MVC application

☐ 1. Start **Visual Studio**.

☐ 2. In the **Start Page - Microsoft Visual Studio** window, on the **File** menu, point to **New**, and then click **Project**.

☐ 3. In the **Create a new project** dialog box, in the languages dropdown, ensure that **C#** is selected.

☐ 4. In the **Create a new project** dialog box, choose **ASP.NET Core Web App (Model-View-Controller)** and click **Next**.

☐ 5. In the **Project Name** text box, type **AnimalsMvc** and click **Next**.

☐ 6. In the **Additional information** dialog box, select **.NET 6.0 (Long-term support)** from the dropdown at the top of the dialog box.

☐ 7. In the **Additional information** dialog box, ensure that the **Configure for HTTPS** check box is **not** selected, and leave the other settings as their default values.

☐ 8. Click on **Create**.

☐ 9. In the **AnimalsMvc - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**.

☐ 10. In **Microsoft Edge**, in the navigation bar, click **Privacy** to review its content.

☐ 11. In **Microsoft Edge**, click **Close**.

☐ 12. In the **AnimalsMvc (Running) - Microsoft Visual Studio** window, on the **Debug** menu, click **Stop Debugging**.

Task 2: Explore the application structure

☐ 1. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, note that there is no Pages folder. Instead, expand **Views**, and then click **_ViewStart.cshtml**.

go deploy

3. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Views/Home**, click **Privacy.cshtml**.

4. In the **Privacy.cshtml** code window, examine the Razor code, and note that there are no links to **.css** files.

5. In the **ActorsRazorPages - Microsoft Visual Studio** window, in **Solution Explorer**, under **Pages**, expand **Shared**, then click **_Layout.cshtml**.

6. In the **_Layout.cshtml** code window, in the **head** element, note that there is a link to **~/css/site.css**.

   Note: This is the CSS **style sheet** file that is applied in **_Layout.cshtml** in the same way as in the Razor Pages exercise. The layout file is slightly different. Note, for example, the navigation links are different, because this is an MVC application.

7. Close the **Microsoft Edge** window.

Task 3: Add simple functionality

1. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, expand **Controllers**, click **HomeController.cs**, and then locate the following code:

```
public IActionResult Error()
{
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpCont
}
```

2. Place the cursor after the **}** (closing brace) sign of the located code, press Enter, and then type the following code:

```
public IActionResult TestPage()
{
        return View();
}
```

3. Right-click the code you just added, and then click **Add View...**.

4. In the **Add New Scaffolded Item** dialog box, ensure that the **Razor View** template is selected (not Razor View - Empty), and click **Add**.

5. In the **Add Razor View** dialog box, ensure that the View name textbox contains the name **TestPage**, and that the Template is **Empty (without model)**.

go deploy

> **Note:** It will take a moment while Visual Studio scaffolds out the view.

7. In the **TestPage.cshtml** code window, select the following code:

```
<h1>TestPage</h1>
```

8. Replace the selected code with the following code:

```
<h2>This is a Test Page</h2>
```

9. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, under **Views**, expand **Shared**, and then click **_Layout.cshtml.**

10. In the **_Layout.cshtml** code window, locate the following code:

```
<li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="P
</li>
```

11. Place the cursor after the located code, press Enter, and then type the following code:

```
<li>
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="T
</li>
```

12. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **Class...**.

13. In the **Add New Item - AnimalsMvc** dialog box, in the **Name** text box, type **Animal**, and then click **Add**.

14. In the **Animal.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
public int Id { get; set; }
public string Name { get; set; }
public string ImageName { get; set; }
public string UniqueInformation { get; set; }
public string Category { get; set; }
```

15. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **New Item...**.

17. In the **Add New Item - AnimalsMvc** dialog box, in the **Name** text box, type IData, and then click **Add**.

18. In the **IData** interface code block, ensure that the interface is declared as a **public interface**.

19. In the **IData** interface code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
List<Animal> AnimalsList { get; set; }
List<Animal> AnimalsInitializeData();
Animal GetAnimalById(int? id);
```

20. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Models**, point to **Add**, and then click **Existing Item...**.

21. In the dialog box, navigate to **D:\Allfiles\Mod01\Labfiles\03_AnimalsMvc_begin**, click **Data.cs**, and then click **Add**.

   ⚠ **Note:** Examine the **Data.cs** content.

22. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, right-click on the **Models**, point to **Add**, and then click **Class...**.

23. In the **Add New Item - AnimalsMvc** dialog box, in the **Name** text box, type IndexViewModel, and then click **Add**.

24. In the **IndexViewModel.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
public List<Animal> Animals { get; set; }
```

25. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **Controllers**, point to **Add**, and then click **Controller**.

26. In the **Add New Scaffolded Item** dialog box, click **MVC Controller - Empty**, and then click **Add**.

27. In the **Add New Item - AnimalsMvc** dialog box, in the **Name:** text box, type AnimalsController, and then click **Add**.

28. In the **AnimalsController.cs** code window, ensure that the cursor is at the end of the **using Microsoft.AspNetCore.Mvc** namespace code, press Enter, and then type the following code:

```
using AnimalsMvc.Models;
```

```
public IActionResult Index()
{
        return View();
}
```

30. In the **AnimalsController.cs** code block, place the cursor after the second **{** (opening brace) sign, press Enter, and then type the following code:

```
private IData _tempData;
public AnimalsController(IData tempData)
{
        _tempData = tempData;
}

public IActionResult Index()
{
        List<Animal> animals = _tempData.AnimalsInitializeData();
        IndexViewModel indexViewModel = new IndexViewModel();
        indexViewModel.Animals = animals;
        return View(indexViewModel);
}

public IActionResult Details(int? id)
{
        var model = _tempData.GetAnimalById(id);
        if (model == null)
        {
                return NotFound();
        }
        return View(model);
}
```

31. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, right-click **wwwroot**, point to **Add**, and then click **New Folder**.

32. In the **NewFolder** box, type images, and then press Enter.

33. Right-click on the **images** folder you just created, and point to **Add**, and then click **Existing Item...**.

34. In the dialog box, navigate to **D:\Allfiles\Mod01\Labfiles\03_AnimalsMvc_begin\Images**, ensure that the file type filter in the dialog is set to **All Files (*.*)**, select all the images, and then click **Add**.

35. In the **AnimalsController.cs** code window, locate the following code, right-click the code, and then click **Add View...**.

```
public IActionResult Index()
```

go deploy

- [ ] 37. In the **Add Razor View** dialog box, ensure that the name in the **View name** text box is **Index**.
- [ ] 38. In the **Add Razor View** dialog box, ensure that the **Empty (without model)** template is selected.
- [ ] 39. In the **Add Razor View** dialog box, ensure that the **Create as a partial view** check box is **unchecked** and the **Use a layout page** check box is **checked**, and then click **Add**.

  Note: It will take a moment while Visual Studio scaffolds out the view.

- [ ] 40. In the **Index.cshtml**, erase all the content in the window, and type the following code:

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

<h2>Index</h2>

<table class="table">
        <thead>
                <tr>
                        <th>
                                @Html.DisplayNameFor(model => model.Animals[0].Name)
                        </th>
                        <th>
                                @Html.DisplayNameFor(model => model.Animals[0].Category)
                        </th>
                        <th></th>
                </tr>
        </thead>
        <tbody>
                @foreach (var item in Model.Animals)
                {
                        <tr>
                                <td>
                                @Html.DisplayFor(modelItem => item.Name)
                                </td>
                                <td>
                                @Html.DisplayFor(modelItem => item.Category)
                                </td>
                                <td>
                                <a asp-action="Details" asp-route-id="@item.Id">Details</a>
                                </td>
                        </tr>
                }
        </tbody>
</table>
```

- [ ] 41. In the **AnimalsMvc - Microsoft Visual Studio** window, in the Solution Explorer, under **Controllers**, click **AnimalsController.cs**.

- [ ] 42. In the **AnimalsController.cs** code window, locate the following code, right-click the code, and then click **Add View...**.

```
public IActionResult Details(int? id)
```

- [ ] 43. In the **Add New Scaffolded Item** dialog box, ensure that the **Razor View** template is selected (not Razor View - Empty), and click **Add**.

- [ ] 44. In the **Add Razor View** dialog box, ensure that the name in the **View name** text box is **Details**.

- [ ] 46. In the **Add Razor View** dialog box, ensure that the **Create as a partial view** check box is **unchecked** and the **Use a layout page** check box is **checked**, and then click **Add**.

- [ ] 47. In the **Details.cshtml**, erase all the content in the window, and type the following code:

```
@model AnimalsMvc.Models.Animal
@{
      ViewData["Title"] = "Details";
}

<h2>Details</h2>

<div>
      <h4>Animal</h4>
      <hr />
      <dl class="dl-horizontal">
            <dt>
                  @Html.DisplayNameFor(model => model.Name)
            </dt>
            <dd>
                  @Html.DisplayFor(model => model.Name)
            </dd>
            <dt>
                  @Html.DisplayNameFor(model => model.Category)
            </dt>
            <dd>
                  @Html.DisplayFor(model => model.Category)
            </dd>
            <dt>
                  @Html.DisplayNameFor(model => model.UniqueInformation)
            </dt>
            <dd>
                  @Html.DisplayFor(model => model.UniqueInformation)
            </dd>
      </dl>

      <div style="padding:10px;">
            @if (Model.ImageName != "")
            {
                  <img src="~/images/@Model.ImageName" alt="Sample Image" height="3
            }

      </div>
</div>
<div>
            <a asp-action="Index">Back to List</a>
</div>
```

- [ ] 48. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, under **Views**, under

☐ 49. In the **_Layout.cshtml** code window, locate the following code:

```
<li>
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="T
</li>
```

☐ 50. Place the cursor after the located code, press Enter, and then type the following code:

```
<li>
      <a class="nav-link text-dark" asp-area="" asp-controller="Animals" asp-action
</li>
```

☐ 51. In the **AnimalsMvc - Microsoft Visual Studio** window, in **Solution Explorer**, click **Program.cs**.

☐ 52. Place the cursor after the line **builder.Services.AddControllersWithViews();**, press Enter, and then type the following code:

```
builder.Services.AddSingleton<IData, Data>();
```

> ⚠ **Note:** We also need to add **using AnimalsMvc.Models;** at the start of this file, but Visual Studio adds this for us automatically.

Task 4: Run the application

☐ 1. In the **AnimalsMvc - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.

☐ 2. In the **AnimalsMvc - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Without Debugging**.

☐ 3. In **Microsoft Edge**, in the navigation bar, click **Test Page** to review its content.

> ⚠ **Note:** The browser window displays the text "This is a Test Page".

☐ 4. In the **Test Page** window, in the navigation bar, click **Animals** to view its content.

> ⚠ **Note:** The browser window displays the **Index.cshtml** page, under the **Animals** folder.

☐ 5. In the **Animals** window, select an animal, and then click **Details** to go to the **Details** page.

> ⚠ **Note:** The browser window displays the **Details.cshtml** page, under the **Animals** folder.

☐ 7. Close **Microsoft Edge**.

☐ 8. In the **AnimalsMvc - Microsoft Visual Studio** window, on the **File** menu, click **Exit**.

✓ **Results**: At the end of this exercise, you have built a simple MVC application in Visual Studio.