Use the Fetch API to GET Data Labs

Perform these labs on your own computer using Visual Studio 2022 or later to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Create Website Using ASP.NET Core

Open Visual Studio 2022 and click on the **Create a New Project** button as shown in Figure 1.

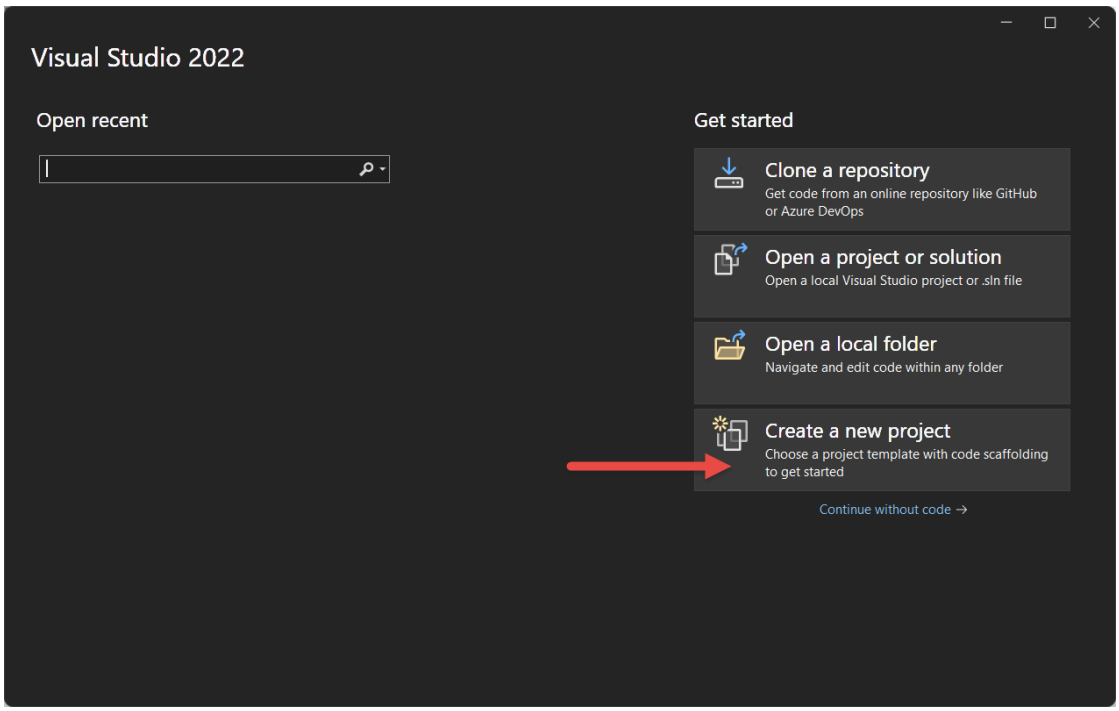

Figure 1: Create a new project in Visual Studio

Create a New Project Screen

In the search box at the top right (Figure 2), type in **MVC**.

In the list below the box locate **ASP.NET Core Web App (Model-View-Controller)** and click on it.

Click the **Next** button to continue to the next screen.

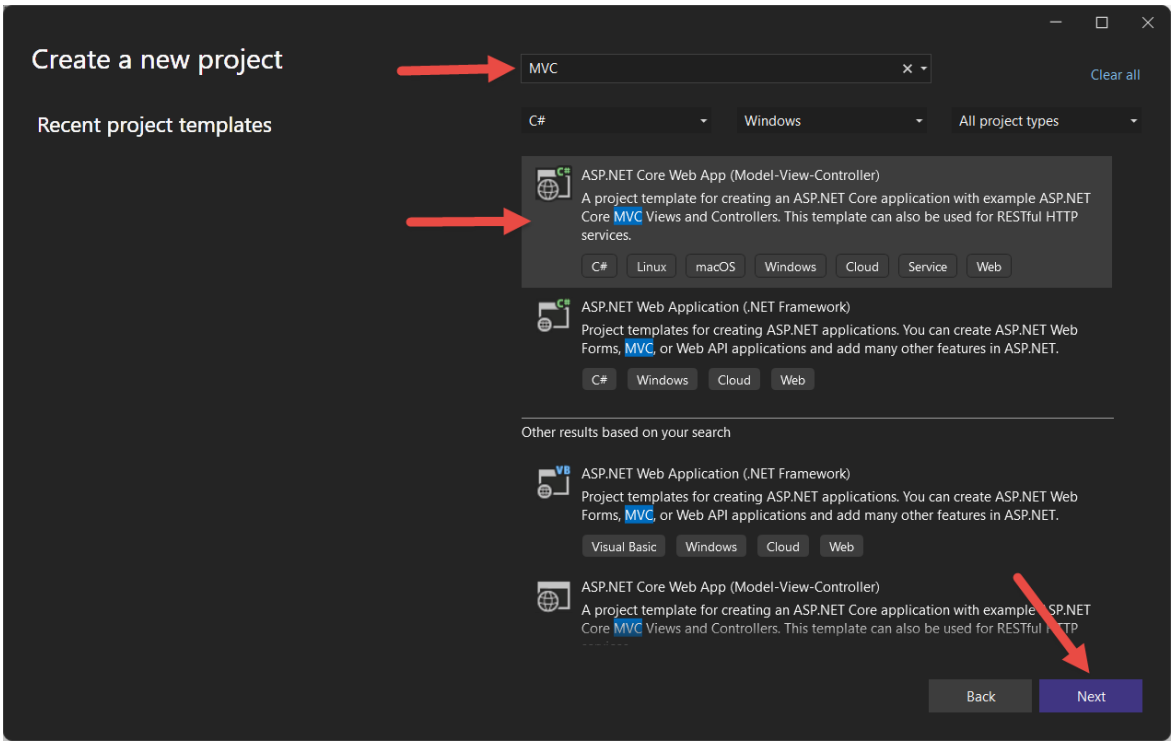

Figure 2: Create a new project screen

Configure Your New Project Screen

As shown in Figure 3 below, fill in the **Project Name** with **FetchAPISamples**.

Fill in the **Location** with the folder location where you typically place your projects.

Check the **Place solution and project in the same directory** option.

Click the **Next** button.

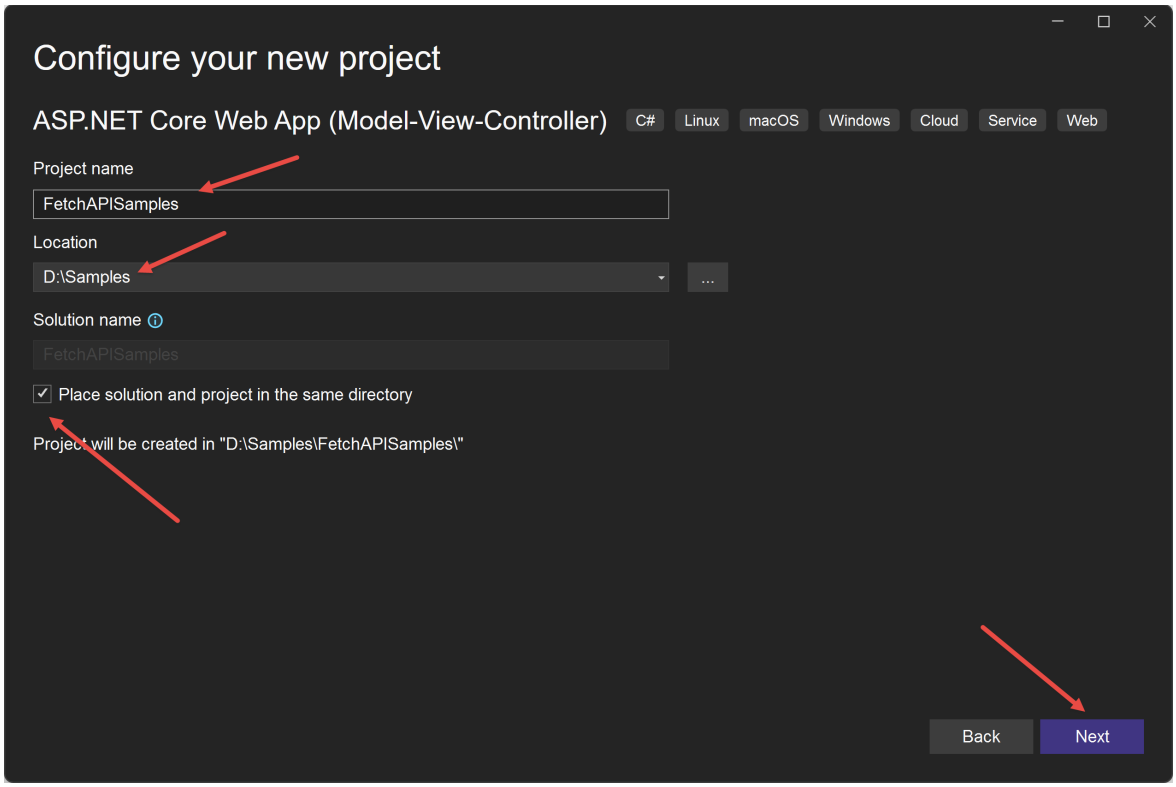

Figure 3: Configure your new project screen

Additional Information Screen

On the Additional Information Screen (shown in Figure 4), change the **Target Framework** to **.NET 6.0 (Long-term support)** or **.NET 8.0 (Long-term support)**.

Set the Authentication Type to **None**.

Uncheck **Configure for HTTPS**.

Uncheck **Do not use top-level statements**.

Click the **Create** button to build the new MVC project.

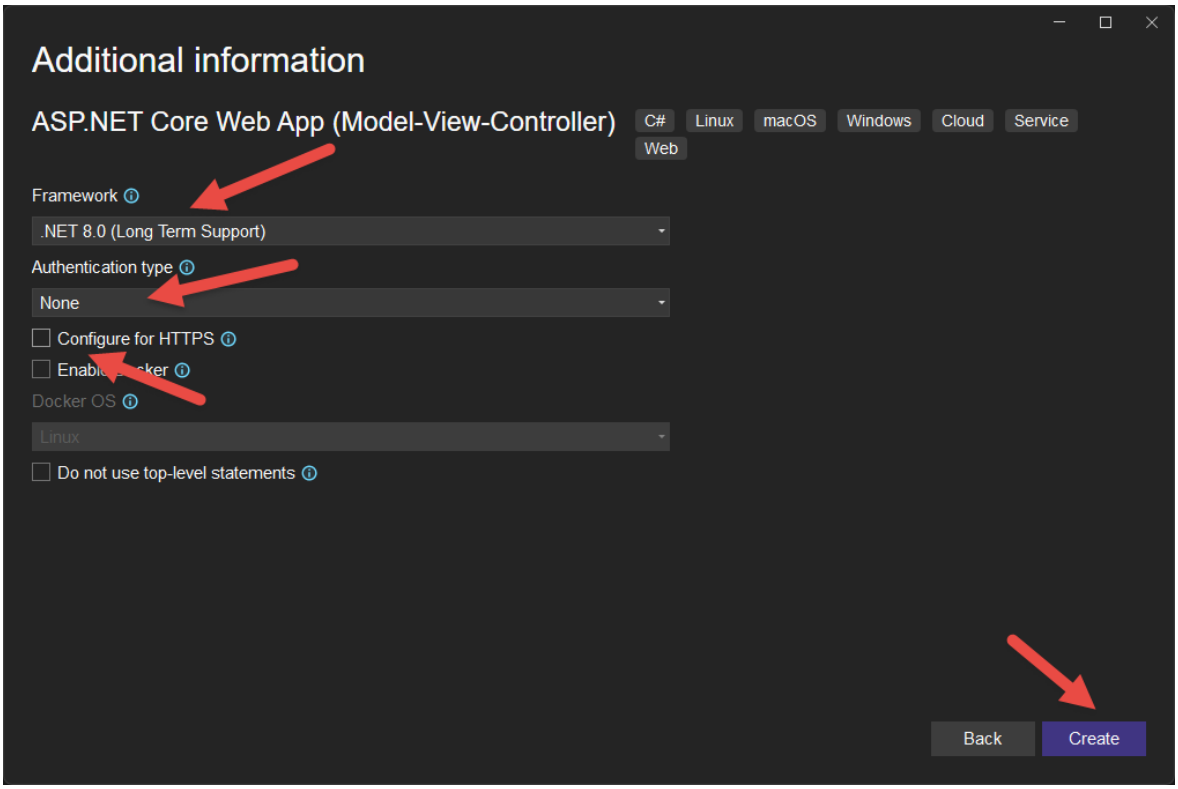

Figure 4: Additional information screen

Try It Out

Run the MVC application by selecting **Debug | Start Debugging** from the Visual Studio menu (or press F5) and you should see a home page that looks like Figure 5.

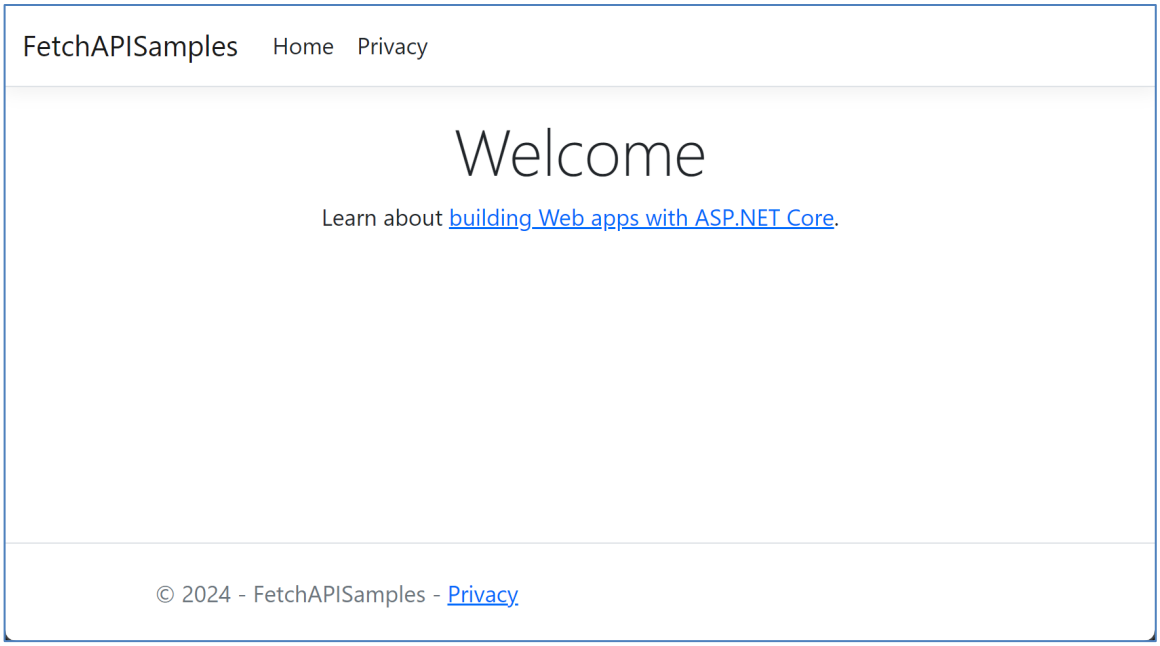

Figure 5: The default MVC application web page

Lab 2: View the Response Object

Open the **Views\Home\Index.cshtml** file and below the </h1> tag add the following code.

NOTE: Be sure to replace the xxxx with your port number from your Minimal Web API server.

```
<div>
  <button onclick="getAllData();">Get All</button>
</div>

<script>
  const SERVER_URI = "http://localhost:xxxx/api";

  function getAllData() {
    fetch(`${SERVER_URI}/people`)
      .then(response => console.log(response))
      .catch(error => console.error(error));
  }
</script>
```

Try It Out

Save all your changes.

Make sure your Web API node server is still running.

Run the MVC application.

Click on the **Get All** button. Bring up the Browser's Console Window and you should see the JSON response object.

Lab 3: Get body Using .json() Method

Open the **\Views\Home\Index.cshtml** file and make it look like the following. Be sure to substitute the localhost:**xxxx** with the port number of your Minimal Web API server you created in the last lesson. Also pay attention to whether you need to specify **http** or **https** for the call to your Minimal Web API server.

```
@{
  ViewData["Title"] = "Fetch API Samples";
}

<div>
  <button onclick="getAllData();">Get All</button>
</div>

@section Scripts
{
  <script>
    const SERVER_URI = "https://localhost:xxxx/api";

    function getAllData() {
      fetch(`${SERVER_URI}/people`)
        .then(response => response.json())
        .then(response => console.log(response))
        .catch(error => console.error(error));
    }
  </script>
}
```

Try It Out

Make sure your Minimal Web API Server is still running.

Run the MVC application.

Click on the **Get All** button. Bring up the Browser's Console Window and you should see a JSON array in the **data** property of the response object.

Lab 4: Get a Single JSON Object Using the Fetch API

Open the **Views\Home\Index.cshtml** file and just above the you created in the last lab, add the following.

```
<div>
  <label for="personId">Person ID</label>
  <input type="text" id="personId" value="1" />
</div>
```

Add a new button within the where the Get All button is located.

```
<div>
  <button onclick="getAllData();">Get All</button>
  <button onclick="getRow();">Get a Row</button>
</div>
```

Just below the getAllData() function, add a new function.

```
function getRow() {
  let id = document.getElementById("personId").value;
  fetch(`${SERVER_URI}/people/${id}`)
    .then(response => response.json())
    .then(response => console.log(response))
    .catch(error => console.error(error));
}
```

Try It Out

Save all your changes.

Run the MVC application.

Click on the **Get a Row** button. Bring up the Brower's Console Window and you should see a single JSON object in the **data** property of the response object.

Lab 5: Not Found

Enter the number 111 into the Product ID input field.

Click on the **Get a Row** button.

Bring up the Brower's Console Window and you should see a JSON response object with the **status** property set to 404, the **data** property is null, and the **message** property shows "Can't Find Person with id='111'.".

Lab 6: Handling Errors in the catch() Method

Open the **Views\Home\Index.cshtml** file and modify the port number from 5000 to 50.

```
http://localhost:50/api/
```

Try It Out

Save your change and run the project. Open your browser tools (F12) to get to the console window and you should see an error message that looks like the following if you are using the Chrome browser.

```
Failed to load resource: net::ERR_CONNECTION_REFUSED
```

In the error message label on the index page, you should see something that looks like the following message.

```
TypeError: Failed to fetch
```

Because the port number does not exist, a network error is detected by the Fetch API. Since the fetch() function is unable to reach the Web API server the .catch() method is called.