# Calling Web API & CORS Lab

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

## Lab 1: Create MVC Website

Startup another instance of Visual Studio 2022 and select **Create New Project** as shown in Figure 1.

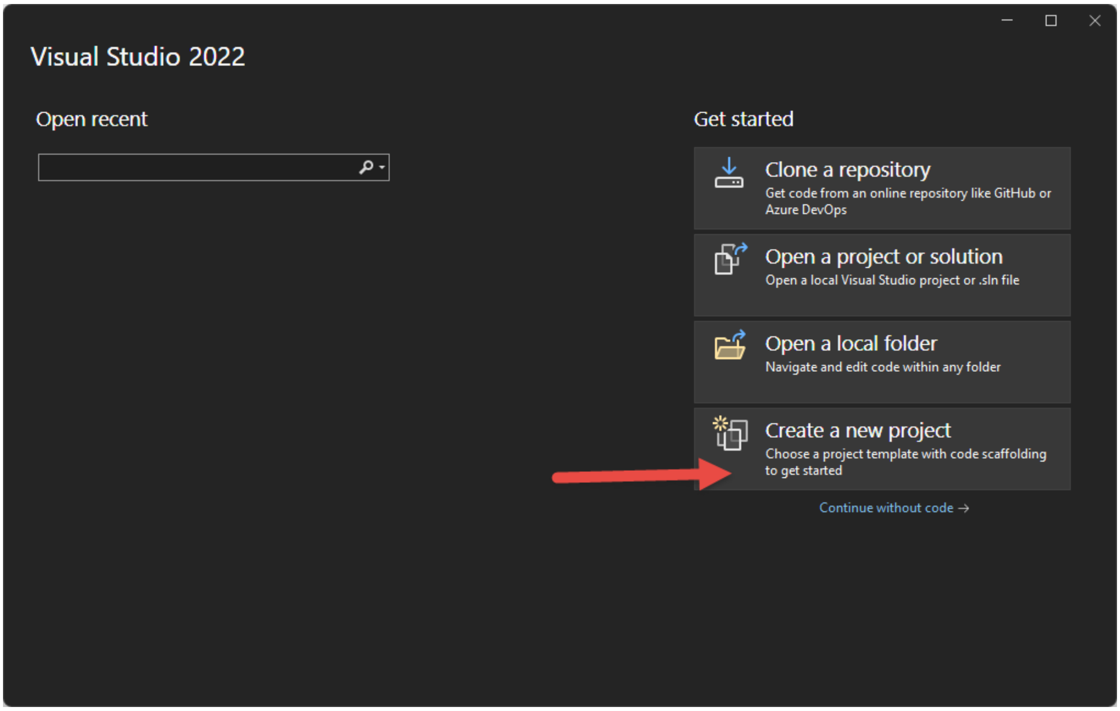
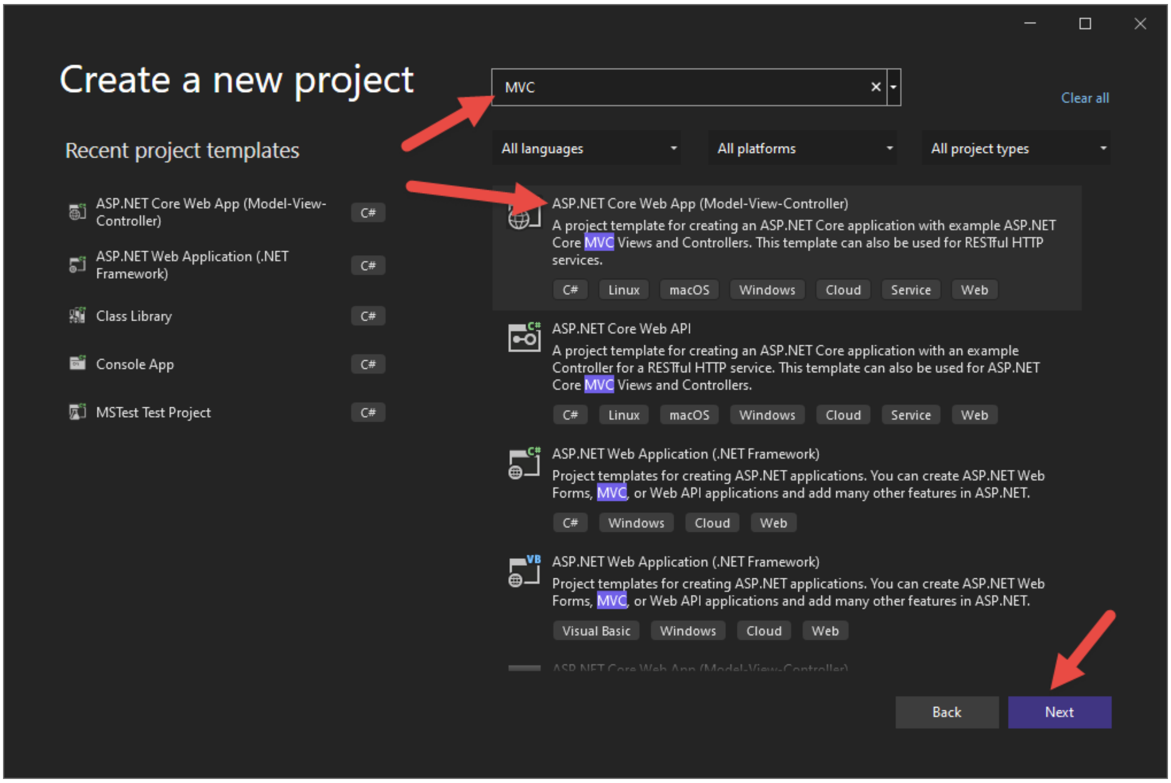

Visual Studio 2022
Open recent
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

Figure 1: Select what you want to do in Visual Studio startup screen

### Create a New Project Screen

Locate the project template **ASP.NET Core Web App (Model-View-Controller)** and select that one as shown in Figure 2.

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



ASP.NET Core Web App (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

C# Linux macOS Windows Cloud Service Web

ASP.NET Core Web API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

C# Linux macOS Windows Cloud Service Web

ASP.NET Web Application (.NET Framework)

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

C# Windows Cloud Web

ASP.NET Web Application (.NET Framework)

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Visual Basic Windows Cloud Web

ASP.NET Core Web App (Model-View-Controller)

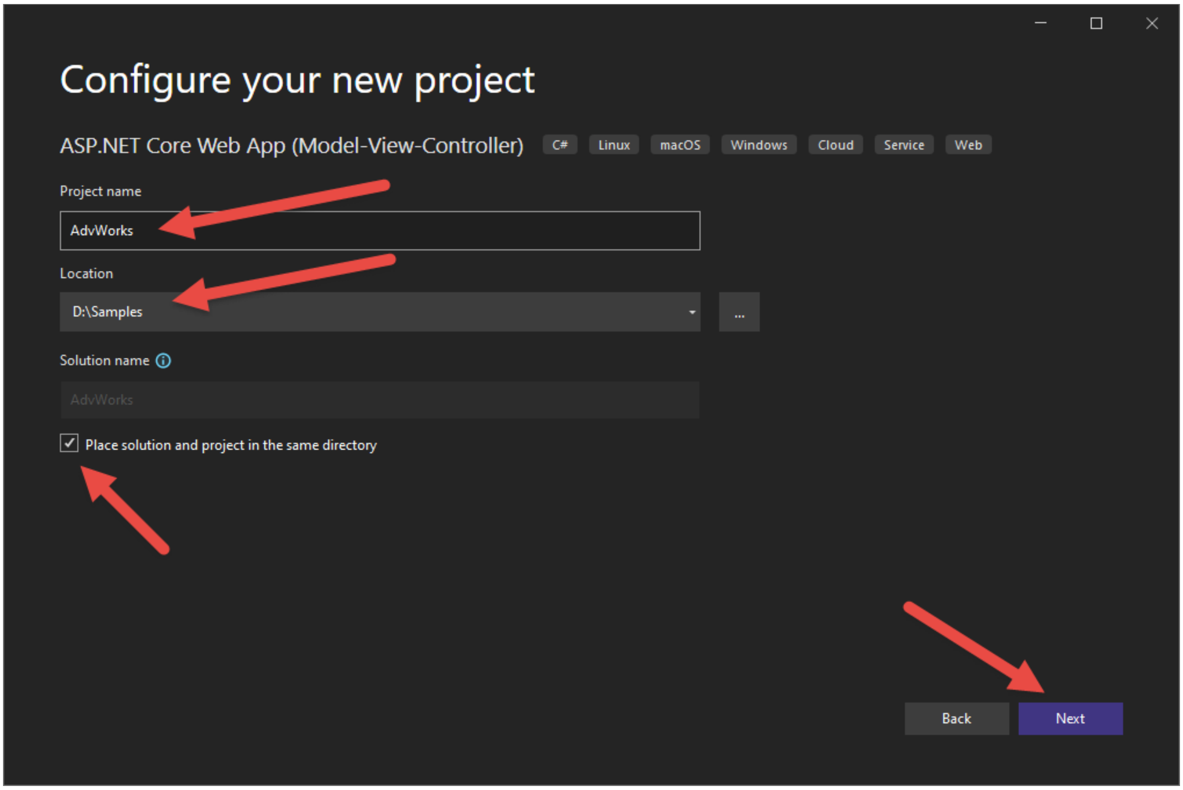
Figure 2: Select the ASP.NET Core Web App (Model-View-Controller) Project.

## Configure Your New Project Screen

Set the **Project Name** with **AdvWorks**

Set the **Location** to where you want the project to reside.

**Check** the Place solution and project in the same directory check box as shown in Figure 3.



Configure your new project

ASP.NET Core Web App (Model-View-Controller)

Project name: AdvWorks

Location: D:\Samples

Solution name: AdvWorks

☑ Place solution and project in the same directory

Figure 3: Configure your new project

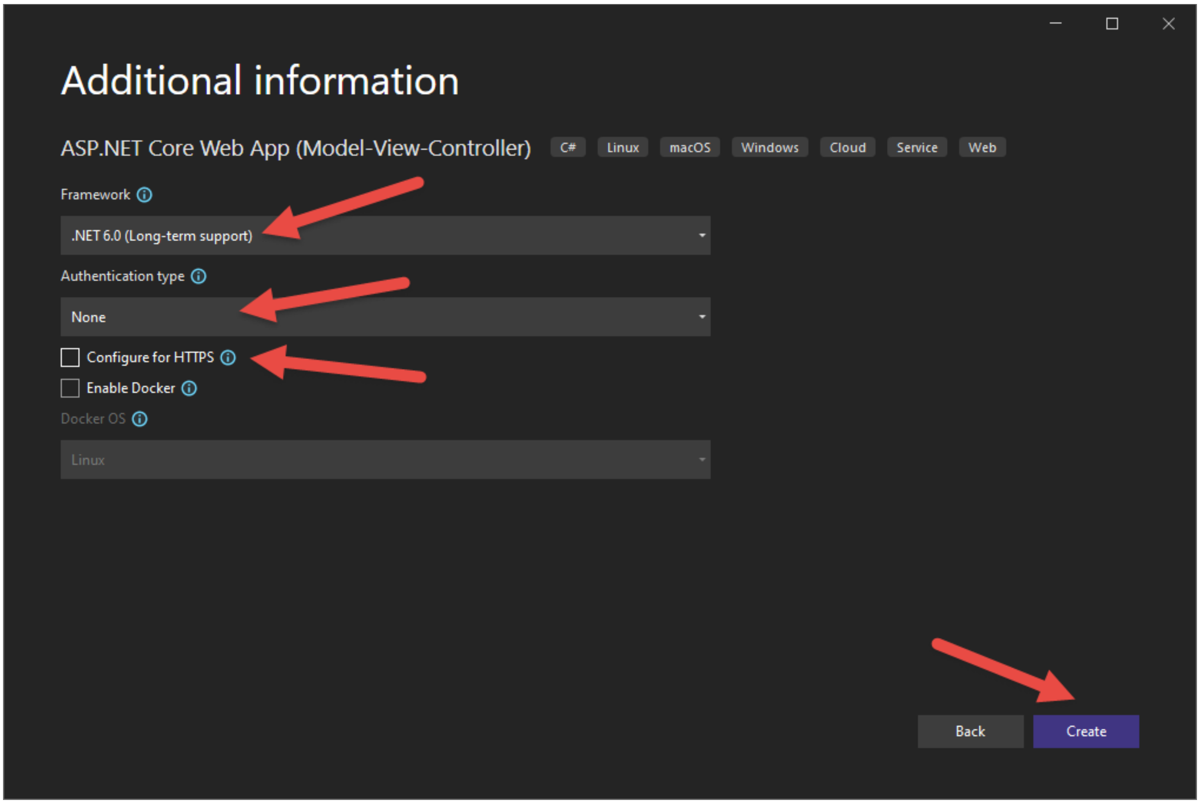
## Additional Information Screen

**Choose** .NET 6.0 (Long-term support)

**Choose** Authentication Type = None

**Uncheck** Configure for HTTPS

Click the **Create** button



Additional information

ASP.NET Core Web App (Model-View-Controller)

Framework: .NET 6.0 (Long-term support)

Authentication type: None

□ Configure for HTTPS

□ Enable Docker

Docker OS: Linux

Figure 4: Set the project information

## Try it Out

Select Debug | Start Debugging (F5) from the VS menu to build the MVC project and launch a browser.

# Lab 2: Call Web API from MVC Website via JavaScript

Open the **\Views\Home\Index.cshtml** file and replace all the code with the following:

```
@{
  ViewData["Title"] = "Call Web API";
}

<div class="row text-center">
  <div class="col">
    <h1>Get a Customer via Web API Call</h1>
  </div>
</div>

<div class="row text-center">
  <div class="col">
    <button onclick="getCustomer();">Get a
Customer</button>
  </div>
</div>

<div class="row text-center">
  <div class="col">
    <textarea id="Customer" rows="10"
cols="100"></textarea>
  </div>
</div>

@section Scripts
{
  <script>
    // The following URL is the location of the API call
in
    // a separate Web API project
    const URL =
"http://localhost:nnnn/api/Customer/306";

    function getCustomer() {
      fetch(URL)
        .then(response => response.json())
        .then(data => {
          $("#Customer").val(JSON.stringify(data));
          console.log(data);
        })
        .catch(error => {
          console.error(error);
          alert("ERROR: Check the Console Window");
        });
    }
  </script>
}
```

**NOTE**: Change the PORT number to match your AdvWorksAPI project.

## Try it Out

Run the Web API application.

Run the MVC application.

Click on the Get a Customer button.

You will get an error message alert.

Open the **Console window** on the **browser** to see the **CORS** error message.

# Lab 3: Modify the Web API Project to Support CORS

CORS is needed for these samples since they run on different ports.

Go to the **AdvWorksAPI** project and stop it running.

Open the **Program.cs** file and after the code where you configured SeriLog add the following code.

```
// Add CORS
builder.Services.AddCors(options =>
{
  options.AddPolicy("AdvWorksAPICorsPolicy",
    builder =>
    {
      //builder.AllowAnyOrigin();
      builder.WithOrigins("http://localhost:nnnn",
"http://www.example.com");
      //builder.WithOrigins("http://localhost:nnnn",
"http://www.example.com").AllowAnyMethod();
      //builder.WithOrigins("http://localhost:nnnn",
"http://www.example.com").WithMethods("GET", "POST",
"PUT");
    });
});
```

**NOTE**: Replace the **PORT** number with the value from the **AdvWorks MVC Website** Project. You can go to the **AdvWorks\Properties** folder and open the **launchSettings.json** file to get the port number.

Scroll down further in the Program.cs file and just before the call to app.UseAuthorization() add the following code.

```
// Enable CORS Middleware
app.UseCors("AdvWorksAPICorsPolicy");
```

# Try it Out

Run the **Web API** application.

Run the **MVC** application (if it is not already running).

Click on the **Get a Customer** button on the home page.

Click on the **Get a Customer** button on the Customer page.

You should now see the customer data appear.