Create Web API Server 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 Web API Project Using Visual Studio 2022

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

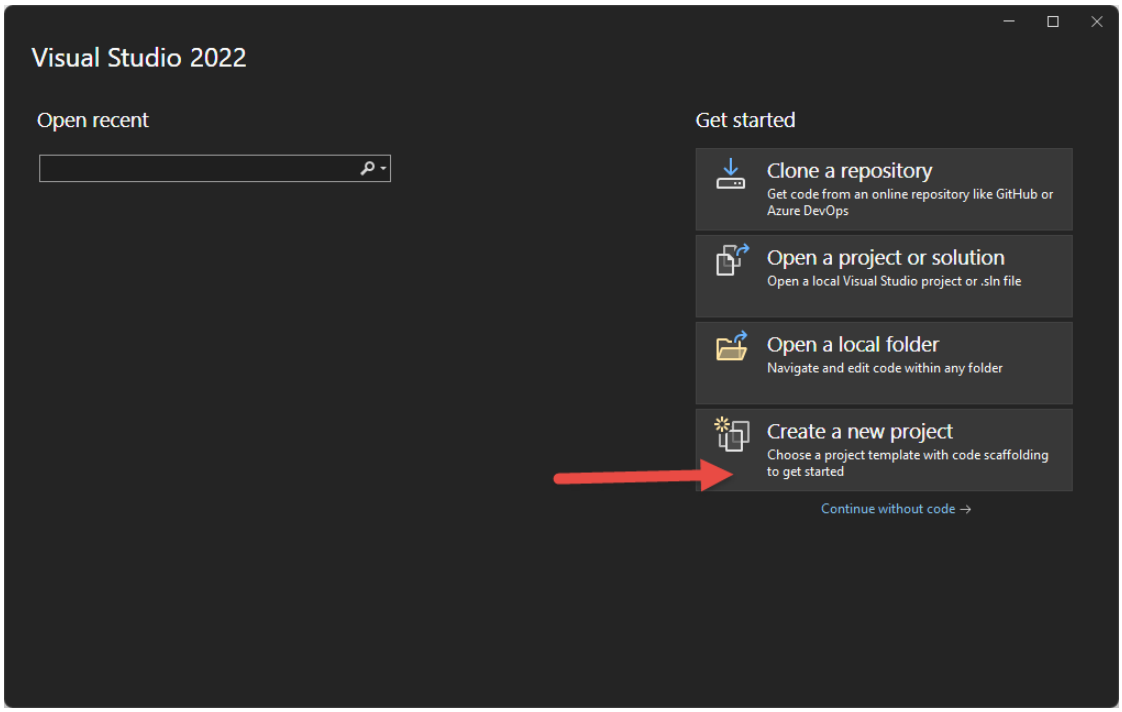

Figure 1: Select what you want to do in Visual Studio startup screen

Create a New Project Screen

Locate the project template **ASP.NET Core Web API** and select that one as shown in Figure 2.

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

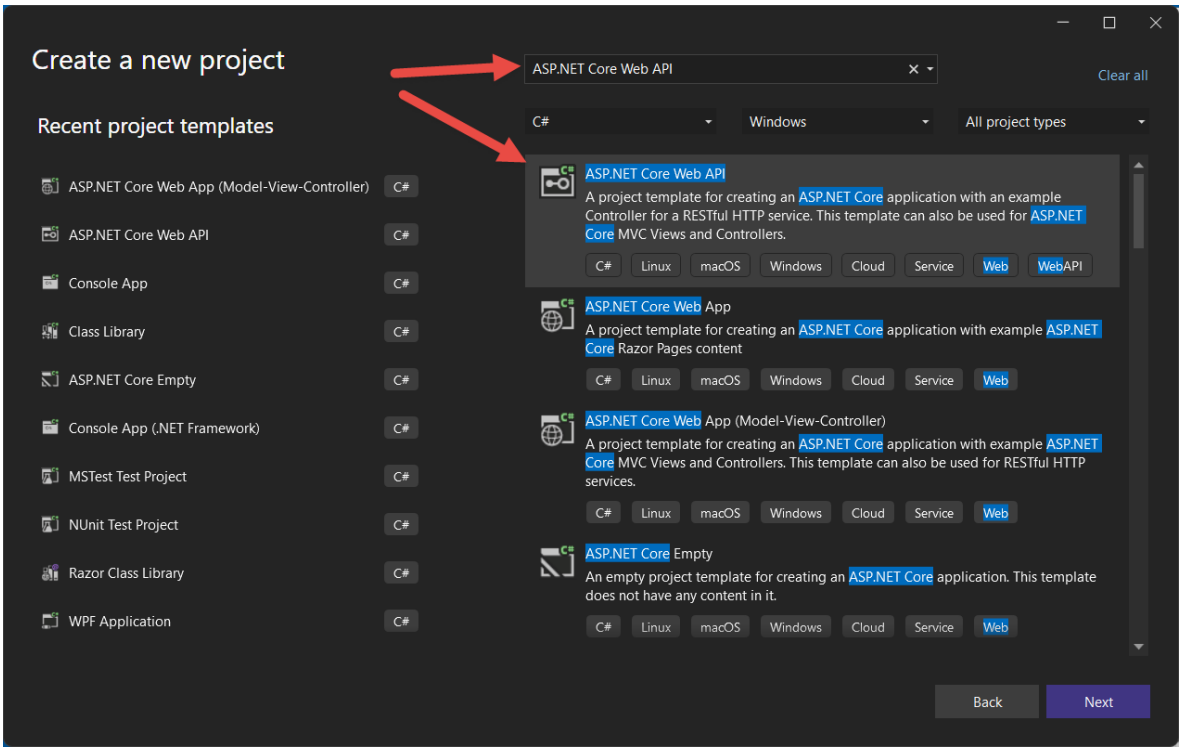

Figure 2: Select the ASP.NET Core Web API Project.

Configure Your New Project Screen

Set the **Project name** to **FetchSamples.WebAPI**.

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.

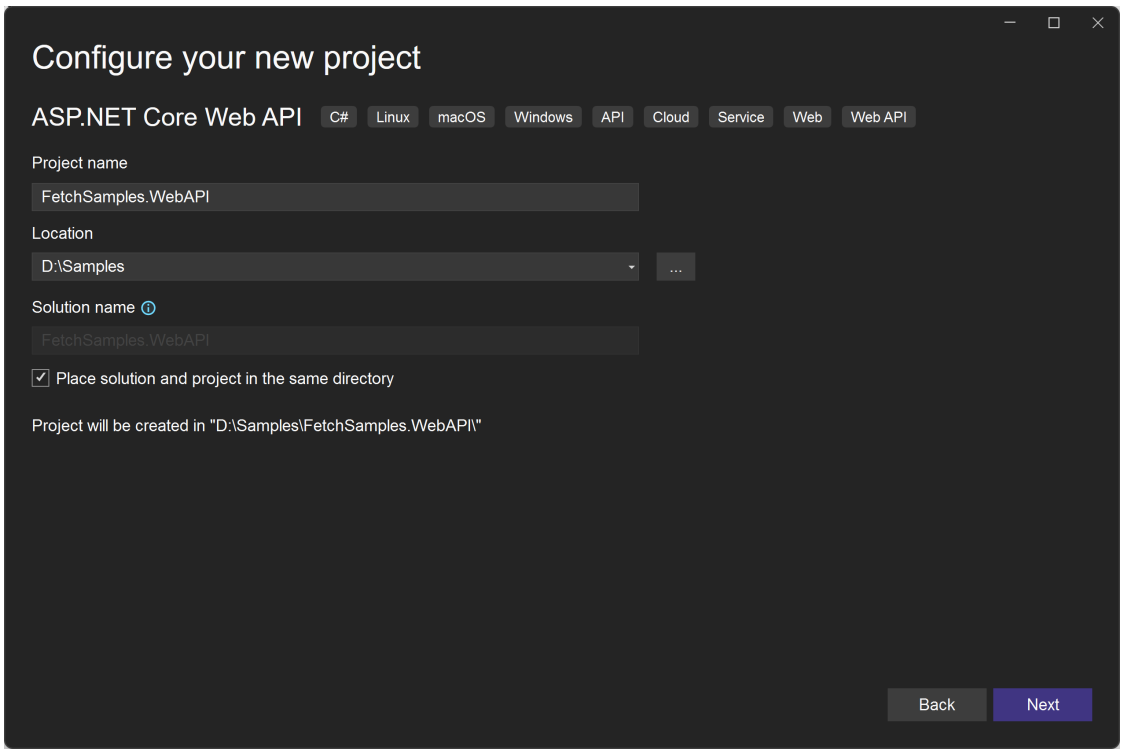

Figure 3: Configure your new project

Additional Information Screen

Choose .NET 6.0 (Long-term support) or .NET 8 (Long-term support)

Choose Authentication Type = None

Uncheck Configure for HTTPS

Uncheck the "Use controllers (uncheck to use minimal APIs)".

Check Enable OpenAPI support as shown in Figure 4.

Click the **Create** button to create the new project.

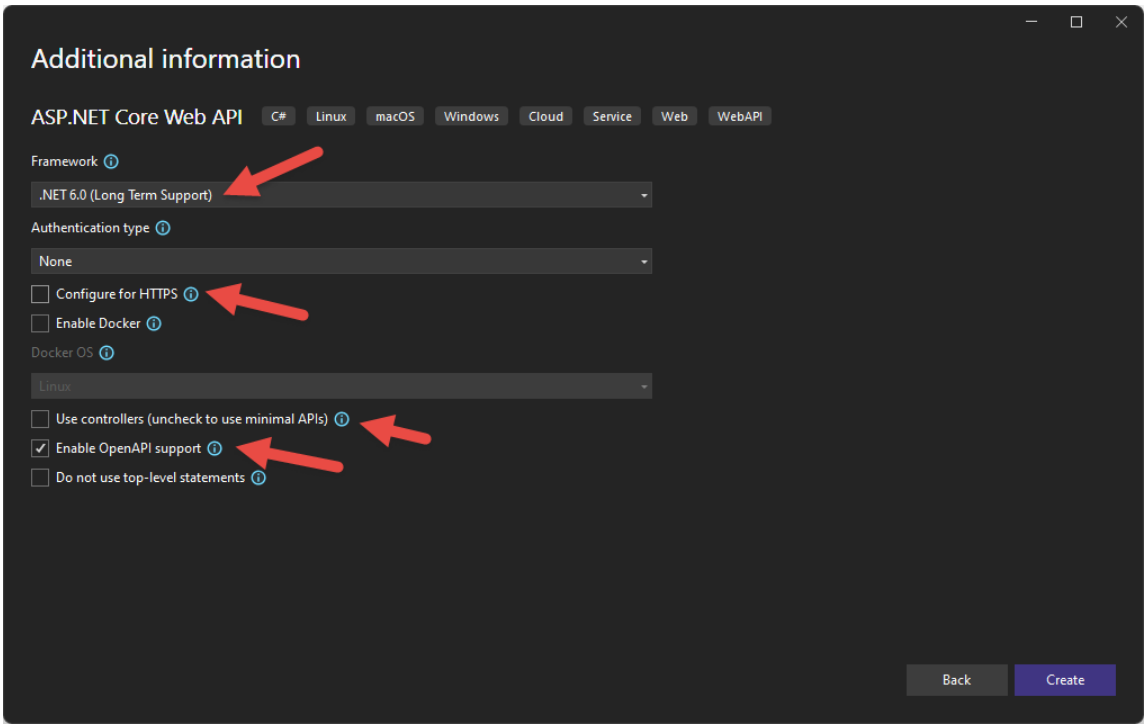

Figure 4: Additional information for your project.

Try it Out

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

> **NOTE**: If you get a dialog box that asks if you should trust the IIS Express certificate, select **Yes**. In the Security Warning dialog that appears next, select **Yes**.

When the browser appears, it will look like Figure 5.

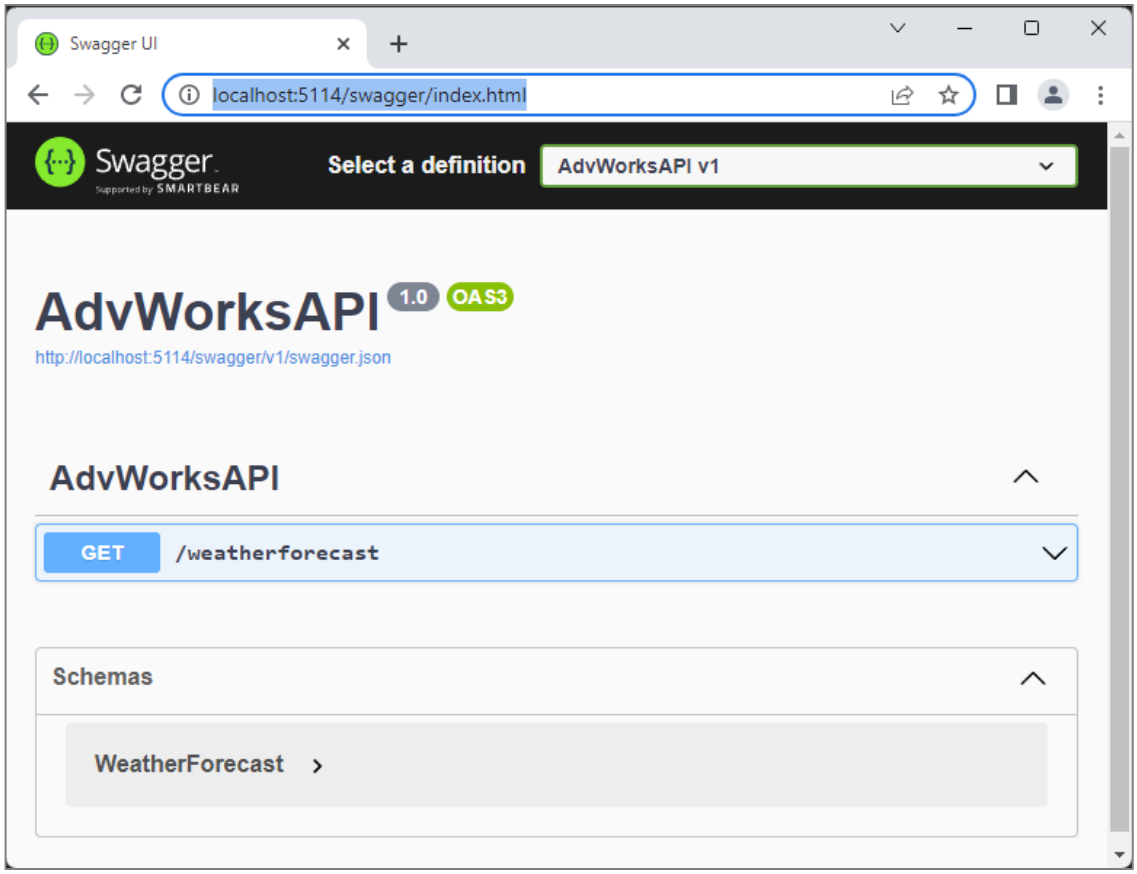

Figure 5: The Swagger Open API page is displayed

Click on the **GET /WeatherForecast** button to display some weather information.

Lab 2: Add a JSON File to Supply Data

Create a new folder named **Assets**.

Create a new file in the **Assets** folder named **people.json**. Put the following data into this new file.

```
[
  {
    "personId": 1,
    "firstName": "Paul",
    "lastName": "Shaefer",
    "emailAddress": "Pauls@netinc.com",
    "startDate": "1991-03-01"
  },
  {
    "personId": 2,
    "firstName": "Michael",
    "lastName": "Kawoski",
    "emailAddress": "Michaelk@netinc.com",
    "startDate": "1999-08-22"
  },
  {
    "personId": 3,
    "firstName": "Sara",
    "lastName": "Winchell",
    "emailAddress": "Saraw@netinc.com",
    "startDate": "2001-02-16"
  },
  {
    "personId": 4,
    "firstName": "John",
    "lastName": "Kroon",
    "emailAddress": "Johnk@netinc.com",
    "startDate": "2002-03-29",
    "salary": 90000
  },
  {
    "personId": 5,
    "firstName": "Tim",
    "lastName": "Nicker",
    "emailAddress": "Timn@netinc.com",
    "startDate": "2008-01-01"
  },
  {
    "personId": 6,
    "firstName": "Russ",
    "lastName": "Martlog",
    "emailAddress": "Russm@netinc.com",
    "startDate": "2009-02-15"
  },
  {
    "personId": 7,
    "firstName": "James",
```

```
    "lastName": "Birdy",
    "emailAddress": "Jamesb@netinc.com",
    "startDate": "2015-05-02"
  },
  {
    "personId": 8,
    "firstName": "Trey",
    "lastName": "Chen",
    "emailAddress": "Treyc@netinc.com",
    "startDate": "2005-06-30"
  },
  {
    "personId": 9,
    "firstName": "Jim",
    "lastName": "Jones",
    "emailAddress": "Jimj@netinc.com",
    "startDate": "2003-10-01"
  },
  {
    "personId": 10,
    "firstName": "John",
    "lastName": "Pittsburgh",
    "emailAddress": "Johnp@netinc.com",
    "startDate": "2007-04-16"
  },
  {
    "personId": 11,
    "firstName": "Jeanne",
    "lastName": "Russell",
    "emailAddress": "Jeanner@netinc.com",
    "startDate": "2004-09-11"
  },
  {
    "personId": 12,
    "firstName": "David",
    "lastName": "Lafeet",
    "emailAddress": "Davidl@netinc.com",
    "startDate": "2011-11-11"
  },
  {
    "personId": 13,
    "firstName": "Khanh",
    "lastName": "Voon",
    "emailAddress": "Khanhv@netinc.com",
    "startDate": "2006-07-05"
  },
  {
```

```
    "personId": 14,
    "firstName": "Jim",
    "lastName": "Russell",
    "emailAddress": "Jimr@netinc.com",
    "startDate": "2012-08-17"
  },
  {
    "personId": 15,
    "firstName": "David",
    "lastName": "Tarkas",
    "emailAddress": "Davidt@netinc.com",
    "startDate": "1999-03-16"
  },
  {
    "personId": 16,
    "firstName": "Craig",
    "lastName": "Showman",
    "emailAddress": "Craigs@netinc.com",
    "startDate": "2001-10-15"
  },
  {
    "personId": 17,
    "firstName": "Brooks",
    "lastName": "Anderson",
    "emailAddress": "Brooksa@netinc.com",
    "startDate": "2003-07-15"
  },
  {
    "personId": 18,
    "firstName": "Mark",
    "lastName": "Parks",
    "emailAddress": "Markp@netinc.com",
    "startDate": "2013-01-15"
  },
  {
    "personId": 19,
    "firstName": "John",
    "lastName": "Smith",
    "emailAddress": "JohnSmith@netinc.com",
    "startDate": "2002-04-01"
  }
]
```

Lab 3: Create a Person Entity Class

Right mouse-click on the project and add a new folder named **EntityClasses**.

Right mouse-click on the **EntityClasses** folder and add a new class named **Person**. Replace the entire contents of this new file with the following code.

```
namespace FetchSamples.WebAPI;

public partial class Person
{
  public int PersonId { get; set; } = 0;
  public string FirstName { get; set; } = string.Empty;
  public string LastName { get; set; } = string.Empty;
  public string? EmailAddress { get; set; } =
string.Empty;
  public DateTime? StartDate { get; set; } =
DateTime.Now;

  #region ToString Override
  public override string ToString()
  {
    return $"{LastName}, {FirstName}";
  }
  #endregion
}
```

Lab 4: Add a IRepository Inteface

Right mouse-click on the project and add a new folder named **Interfaces**.

Right mouse-click on the **Interfaces** folder and add a new class named **IRepository**. Replace the entire contents of this new file with the following code.

```
namespace FetchSamples.WebAPI;

public interface IRepository<T> where T : class
{
  List<T> Get();
  T? Get(int id);
  T? Insert(T entity);
  T? Update(T entity);
  bool Delete(T entity);
}
```

Lab 5: Create a Repository Object to Read JSON File

Right mouse-click on the project and add a new folder named **RepositoryClasses**.

Right mouse-click on the **RepositoryClasses** folder and add a new class named **PersonRepository**. Replace the entire contents of this new file with the following code.

```
using System.Text.Json;

namespace FetchSamples.WebAPI;

/// <summary>
/// This class creates fake data for the Person table.
/// </summary>
public partial class PersonRepository :
IRepository<Person>
{
  #region GetPeopleFromFile Method
  private string GetPeopleFromFile()
  {
    string ret = string.Empty;

    string fileName = Directory.GetCurrentDirectory();
    fileName += @"\Assets\people.json";

    if (File.Exists(fileName)) {
      ret = File.ReadAllText(fileName);
    }

    return ret;
  }
  #endregion

  #region SavePeopleToFile Method
  private void SavePeopleToFile(List<Person> people)
  {
    string fileName = Directory.GetCurrentDirectory();
    fileName += @"\Assets\people.json";

    JsonSerializerOptions options = new() {
      PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
      WriteIndented = true
    };
    string tmp = JsonSerializer.Serialize(people,
options);
    if (!string.IsNullOrEmpty(tmp)) {
      if (File.Exists(fileName)) {
        File.WriteAllText(fileName, tmp);
      }
    }
  }
  #endregion

  #region Get Method
```

```
  /// <summary>
  /// Get all Person objects
  /// </summary>
  /// <returns>A list of People objects</returns>
  public List<Person> Get()
  {
    List<Person> ret = new();
    string people = string.Empty;

    people = GetPeopleFromFile();

    if (!string.IsNullOrEmpty(people)) {
      JsonSerializerOptions options = new() {
        PropertyNamingPolicy =
JsonNamingPolicy.CamelCase
      };
      var tmp =
JsonSerializer.Deserialize<List<Person>>(people,
options);
      if (tmp != null) {
        ret = tmp;
      }
    }

    return ret;
  }
  #endregion

  #region Get(id) Method
  /// <summary>
  /// Get a single Person object
  /// </summary>
  /// <param name="id">The value to locate</param>
  /// <returns>A valid Person object, or null if not
found</returns>
  public Person? Get(int id)
  {
    return Get().Where(row => row.PersonId ==
id).FirstOrDefault();
  }
  #endregion

  #region Insert Method
  /// <summary>
  /// Insert a new Person object
  /// </summary>
  /// <param name="entity">The data to insert</param>
```

```
  /// <returns>The inserted Person object</returns>
  public Person? Insert(Person entity)
  {
    List<Person> list;

    list = Get();

    // Increment the Primary Key field
    entity.PersonId = list.Max(row => row.PersonId) + 1;

    // Add to collection
    list.Add(entity);

    // Insert into data store
    SavePeopleToFile(list);

    return entity;
  }
  #endregion

  #region Update Method
  /// <summary>
  /// Update existing Person object
  /// </summary>
  /// <param name="entity">The data to update</param>
  /// <returns>The updated Person object</returns>
  public Person? Update(Person entity)
  {
    List<Person> list;

    list = Get();

    // Look up the data by the specified id
    Person? current = list.FirstOrDefault(row =>
row.PersonId == entity.PersonId);

    if (current != null) {
      // Update Current Data
      current.FirstName = entity.FirstName;
      current.LastName = entity.LastName;
      current.EmailAddress = entity.EmailAddress;
      current.StartDate = entity.StartDate;

      // Update File
      SavePeopleToFile(list);
    }
```

```
    return current;
  }
  #endregion

  #region Delete Method
  /// <summary>
  /// Delete a Person object
  /// </summary>
  /// <param name="entity">The data to insert</param>
  /// <returns>True if delete, false if not
found</returns>
  public bool Delete(Person entity)
  {
    bool ret = false;
    List<Person> list;

    list = Get();

    // Look up the data by the specified id
    Person? current = list.FirstOrDefault(row =>
row.PersonId == entity.PersonId);

    if (current != null) {
      // Remove data from collection
      list.Remove(current);

      // Delete data from data store
      SavePeopleToFile(list);

      ret = true;
    }

    return ret;
  }
  #endregion
}
```

Lab 6: Add CORS

Open the **Program.cs** file and at the top of the file make it look like the following.

```
using FetchSamples.WebAPI;

var builder = WebApplication.CreateBuilder(args);

// Add CORS
builder.Services.AddCors(options =>
{
  options.AddPolicy("CorsPolicy",
    builder =>
    {

builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod
();
    });
});

// Add Authorization Service
builder.Services.AddAuthorization();
```

Scroll down and just before the call to **app.Run()**, add the following.

```
// Enable Authorization
app.UseAuthorization();

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

Lab 7: Create a Response Class

Right mouse-click on the **EntityClasses** folder and add a new class named **Response**. Replace the entire contents of this new file with the following code.

```
namespace FetchSamples.WebAPI;

public class Response
{
  public int Status { get; set; }
  public string StatusText { get; set; } = string.Empty;
  public string Message { get; set; } = string.Empty;
  public object? Data { get; set; }
}
```

Lab 8: Create the Web API Endpoints

Open the **Program.cs** file and **REMOVE** all traces of the "Weather" API.

Add the following endpoints just before the **app.Run()** call.

```
app.MapGet("/api/people", () =>
{
  IResult ret;
  List<Person> list;
  PersonRepository repo = new();
  list = repo.Get();

  if (list.Count > 0) {
    ret = Results.Ok(new Response() {
      Status = 200,
      StatusText = "OK",
      Message = "All People Retrieved.",
      Data = list
    });
  }
  else {
    ret = Results.NotFound(new Response() {
      Status = 404,
      StatusText = "NotFound",
      Message = "No People Available.",
      Data = null
    });
  }

  return ret;
});

app.MapGet("/api/people/{id}", (int id) =>
{
  IResult ret;
  Person? entity = new();
  PersonRepository repo = new();
  entity = repo.Get(id);

  if (entity != null) {
    ret = Results.Ok(new Response() {
      Status = 200,
      StatusText = "OK",
      Message = "Person Retrieved.",
      Data = entity
    });
  }
  else {
    ret = Results.NotFound(new Response() {
      Status = 404,
      StatusText = "NotFound",
      Message = $"Can't Find Person with id='{id}'.",
```

```
        Data = null
      });
    }

    return ret;
});
```

Try It Out

Run the application and you should see the following web page appear.

Expand the **GET /people** and click on the **Try it out** button. Click the **Execute** button and you should see a list of people from the JSON file appear.

Expand the **GET /people/:id** and click on the **Try it out** button. Enter the number 1 into the input field. Click the **Execute** button and you should see a single person JSON object appear.