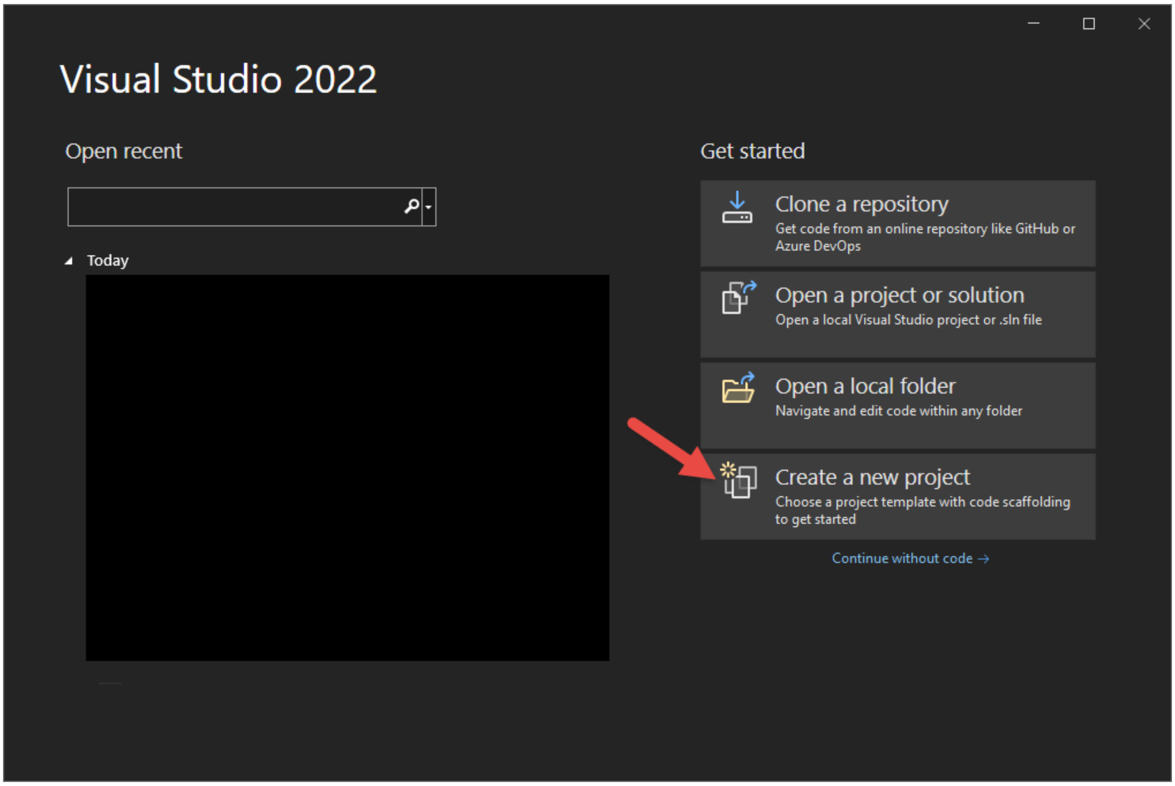
MVC Entity Framework Data Lab

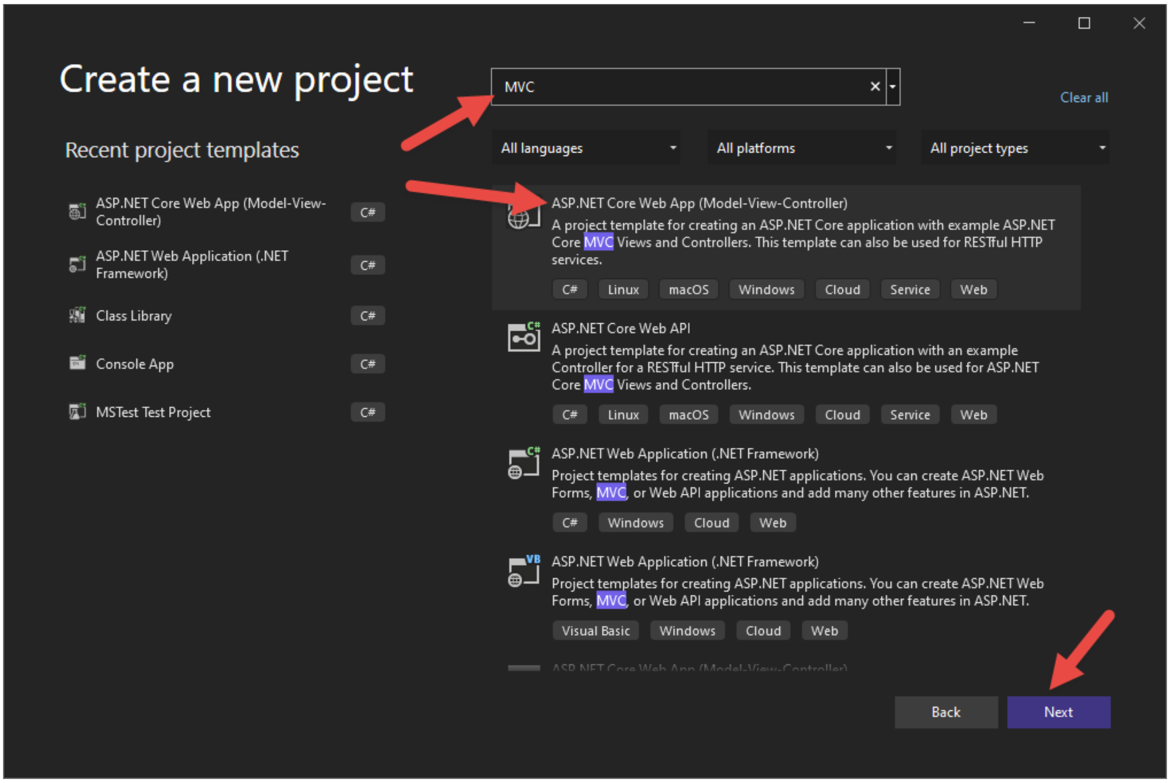
Lab 1: Create Project

Open Visual Studio 2022 and select **Create a New Project**


Visual Studio 2022
Open recent
Today
[redacted]
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 →

In the search box at the top right, type in **MVC**

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



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

ASP.NET Web Application (.NET Framework) C#

Class Library C#

Console App C#

MSTest Test Project C#

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

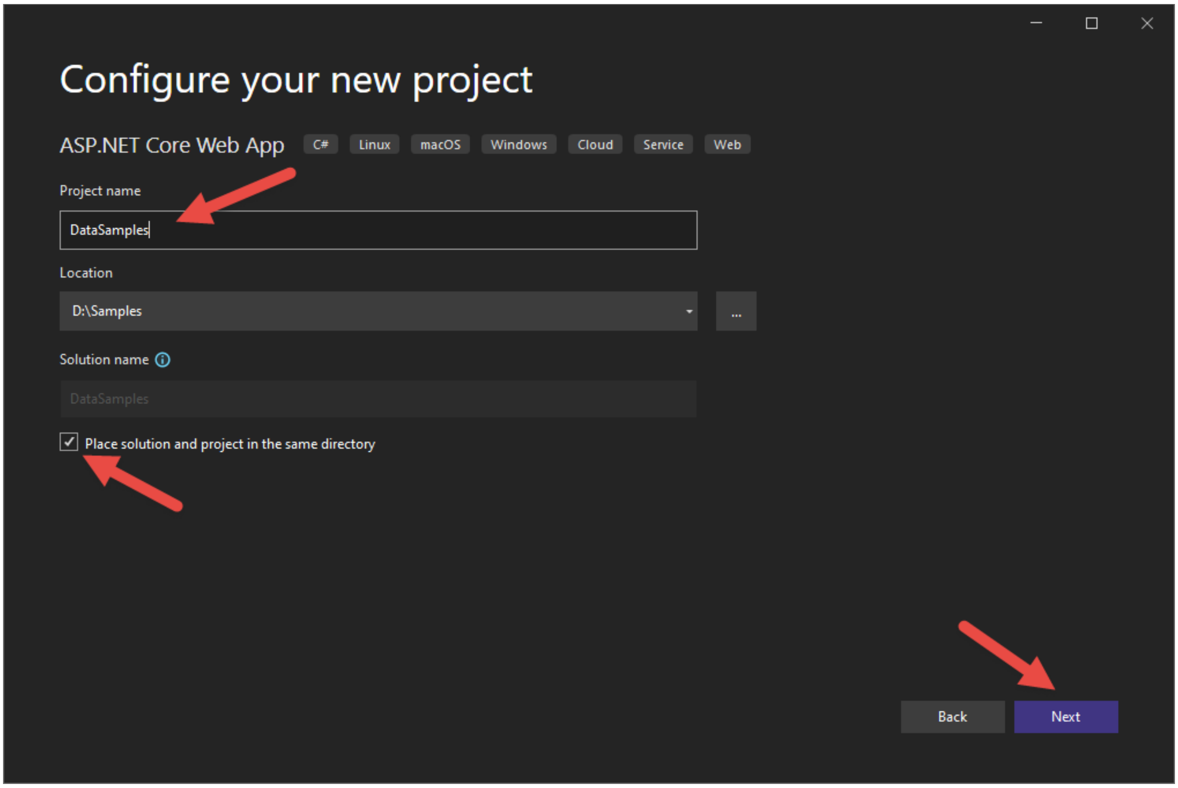
ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web

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

Click the **Next** button

Fill in the **Project Name** with **DataSamples**

Fill in the **Location** with your project folder



Configure your new project

ASP.NET Core Web App

Project name: DataSamples

Location: D:\Samples

Solution name: DataSamples

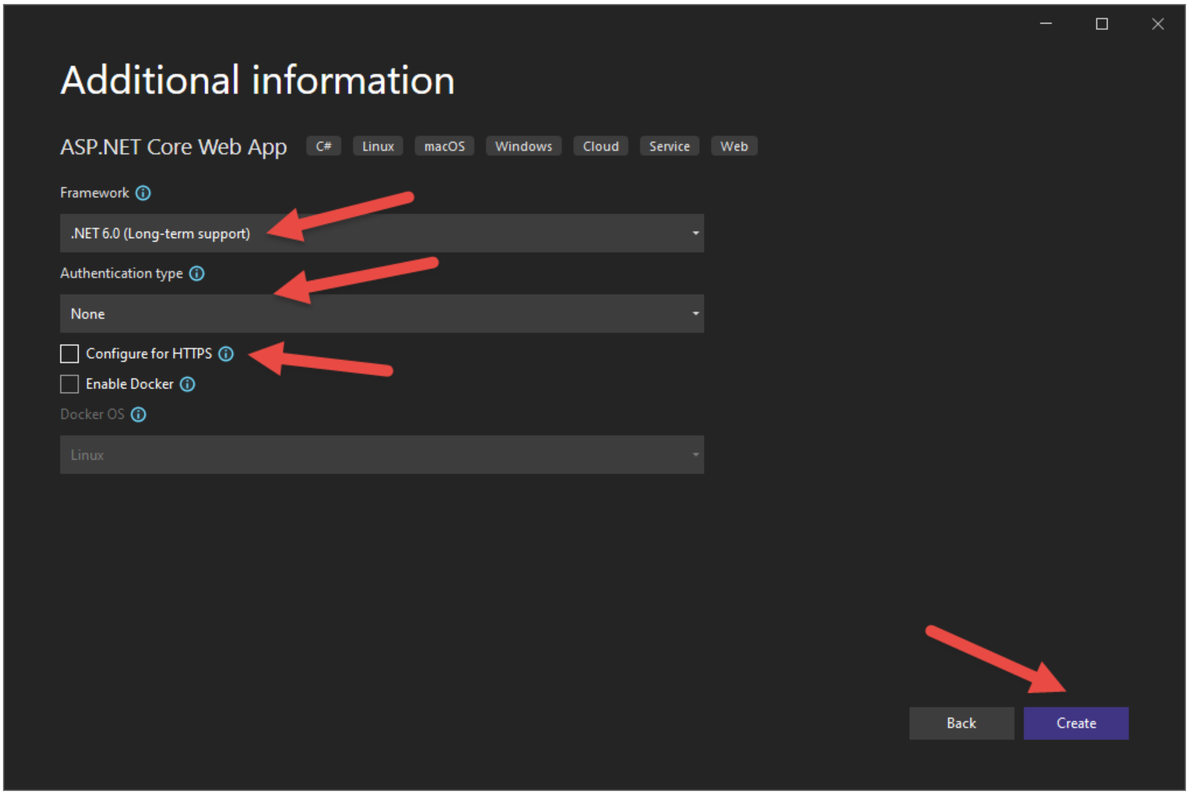
☑ Place solution and project in the same directory

Click the **Next** button

Change the Target Framework to **.NET 6.0 (Long-term support)**

Set the Authentication Type to **None**

Uncheck **Configure for HTTPS**



Additional information

ASP.NET Core Web App

Framework: .NET 6.0 (Long-term support)

Authentication type: None

□ Configure for HTTPS

□ Enable Docker

Docker OS: Linux

Click the **Create** button

Lab 2: Add Connection String

Open the **appsettings.Development.json** file and add a new property

```
"ConnectionStrings": {
  "DefaultConnection":
"Server=Localhost;Database=AdventureWorksLT;Integrated Security=Yes"
}
```

Lab 3: Add Entity Framework

Right mouse-click on the project and select **Manage NuGet Packages…**

Click on the Browse tab and type in

```
Microsoft.EntityFrameworkCore.SqlServer
```

Install the package

Lab 4: Create Product Class

Right mouse-click on the DataSamples project and select **Properties**.

Set the **Nullable** setting to **Disable**.

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

Right mouse-click on the \EntityClasses folder and add a new class named **Product**

```
using System.ComponentModel.DataAnnotations.Schema;

namespace DataSamples.EntityClasses
{
  [Table("Product", Schema = "SalesLT")]
  public partial class Product
  {
    public Product()
    {
      SellStartDate = DateTime.Now;
    }

    public int? ProductID { get; set; }
    public string Name { get; set; }
    public string ProductNumber { get; set; }
    public string Color { get; set; }
    public decimal StandardCost { get; set; }
    public decimal ListPrice { get; set; }
    public string Size { get; set; }
    public decimal? Weight { get; set; }
    public DateTime SellStartDate { get; set; }
    public DateTime? SellEndDate { get; set; }
    public DateTime? DiscontinuedDate { get; set; }

    public override string ToString()
    {
      return $"{Name} ({ProductID})";
    }
  }
}
```

Lab 5: Create DbContext Class

Right mouse-click on the \Models folder and add a new class named **AdvWorksLTDbContext**

```
using Microsoft.EntityFrameworkCore;
using DataSamples.EntityClasses;

namespace DataSamples.Models
{
  public partial class AdvWorksLTDbContext : DbContext
  {
    public AdvWorksLTDbContext
(DbContextOptions<AdvWorksLTDbContext> options) : base(options)
    {
    }

    public virtual DbSet<Product> Products { get; set; }

    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
      base.OnModelCreating(modelBuilder);
    }
  }
}
```

Lab 6: Create Repository Pattern

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

Right mouse-click on the \RepositoryClasses folder and create a new class named **IProductRepository**

Change the class to an interface

```
using DataSamples.EntityClasses;

namespace DataSamples.RepositoryClasses
{
  public interface IProductRepository
  {
    List<Product> Get();
  }
}
```

Right mouse-click on the \RepositoryClasses folder and create a new class named **ProductRepository**

```
using DataSamples.EntityClasses;
using DataSamples.Models;

namespace DataSamples.RepositoryClasses
{
  public class ProductRepository : IProductRepository
  {
    public ProductRepository(AdvWorksLTDbContext context)
    {
      _DbContext = context;
    }
    private AdvWorksLTDbContext _DbContext;

    public List<Product> Get()
    {
      return _DbContext.Products.ToList();
    }
  }
}
```

Lab 7: Create View Model

Right mouse-click on the DataSamples project and create a new folder named **ViewModelClasses**.

Right mouse-click on the \ViewModelClasses folder and create a new class named **ProductViewModel**

```
using DataSamples.EntityClasses;
using DataSamples.RepositoryClasses;

namespace DataSamples.ViewModelClasses
{
  public class ProductViewModel
  {
    #region Constructors
    public ProductViewModel()
    {
    }

    public ProductViewModel(IProductRepository repo)
    {
      Repository = repo;
    }
    #endregion

    #region Properties
    public IProductRepository Repository { get; set; }
    public List<Product> Products { get; set; }
    #endregion

    #region LoadProducts Method
    public virtual void LoadProducts()
    {
      Products = Repository.Get().OrderBy(p => p.Name).ToList();
    }
    #endregion
  }
}
```

Lab 8: Modify Program Class

Open the **Program.cs** file and add two using statements:

```
using Microsoft.EntityFrameworkCore;
using DataSamples.Models;
using DataSamples.RepositoryClasses;
```

Add the following code just after the WebApplication.CreateBuilder(args); line:

```
// Read in the connection string from the appSettings.json file
string connString =
builder.Configuration.GetConnectionString("DefaultConnection");

// Setup the AdvWorks DB Context
builder.Services.AddDbContext<AdvWorksLTDbContext>(
  options => options.UseSqlServer(connString));
```

Add a new scoped version of the IProductRepository just after the builder.Services.AddControllersWithViews(); call

```
// Create a scoped version of Product Repository
builder.Services.AddScoped<IProductRepository, ProductRepository>();
```

Lab 9: Modify HomeController

Open the **HomeController.cs** file and add the appropriate using statements.

```
using DataSamples.Models;
using DataSamples.RepositoryClasses;
using DataSamples.ViewModelClasses;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
```

Add a new private field

```
private readonly IProductRepository _repo;
```

Modify the constructor

```
public HomeController(ILogger<HomeController> logger,
  IProductRepository repo)
{
  _logger = logger;
  _repo = repo;
}
```

Add a **Sample01**() method

```
[HttpGet]
public IActionResult Sample01()
{
  ProductViewModel vm = new(_repo);

  vm.LoadProducts();

  return View(vm);
}
```

Lab 10: Add HTML

Right mouse-click on the **\Views\Home** folder and create a new view named **Sample01.cshtml** and add the following code

```
@using DataSamples.EntityClasses
@model DataSamples.ViewModelClasses.ProductViewModel

@{
  ViewData["Title"] = "Sample 1";
}

<h2>Sample 1 - Unordered List</h2>

<ul>
  @foreach (Product item in Model.Products) {
    <li>@item.Name</li>
  }
</ul>
```

Open the **Index.cshtml** file and replace all the code with the following

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

<h1>Data Samples</h1>

<div class="list-group">
  <a asp-action="Sample01" asp-controller="Home" class="list-group-
item">
    Sample 1 - Unordered List
  </a>
</div>
```

Try it Out

Run the application and display a list of products.