# ASP.NET Core Web API Guide

## 📌 What is ASP.NET Core Web API?

ASP.NET Core Web API is a framework for building RESTful APIs in .NET Core that:  
- Enables communication between clients and servers over HTTP.  
- Supports CRUD operations (Create, Read, Update, Delete).  
- Uses JSON format for data exchange.  
- Works on Windows, Linux, and macOS.

## 📌 When to Use ASP.NET Core Web API?

Use ASP.NET Core Web API when:  
✅ You need a backend API for a mobile, web, or desktop app.  
✅ You want to create a microservices-based architecture.  
✅ You need data exchange between frontend and backend.  
✅ You want to integrate third-party services via API.  
✅ You need a scalable and lightweight API.

## 📌 Folder Structure of .NET Core 8.0 Web API

When you create a .NET Core 8.0 Web API project, you get the following structure:

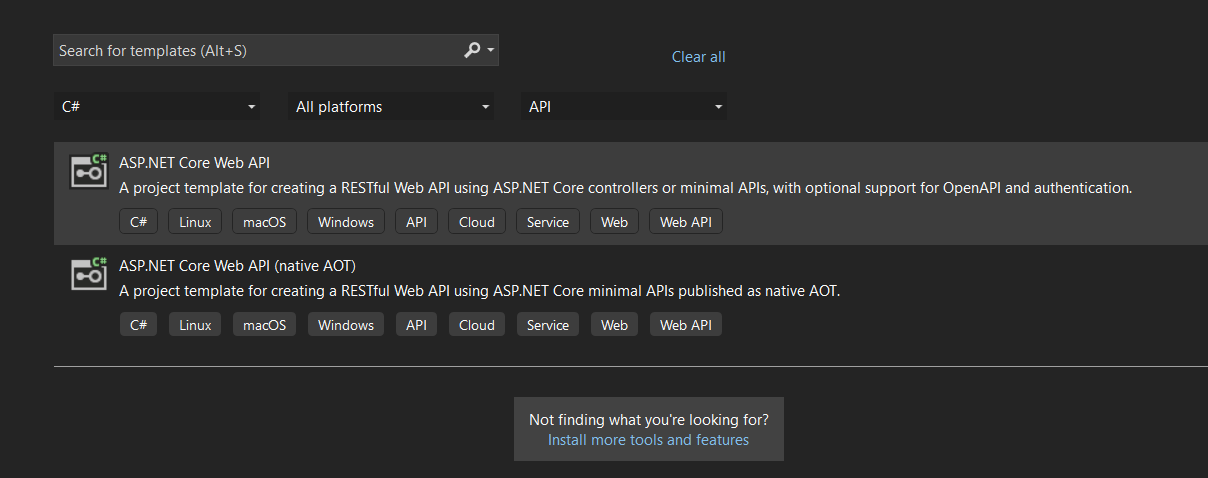
MyWebAPI/  
│-- Controllers/ # API controllers (handles HTTP requests)  
│ ├── WeatherController.cs  
│-- Models/ # Data models (DTOs, Entities)  
│ ├── Product.cs  
│-- Services/ # Business logic (optional)  
│ ├── ProductService.cs  
│-- Data/ # Database context (EF Core)  
│ ├── AppDbContext.cs  
│-- Program.cs # Entry point (configures services, middleware)  
│-- appsettings.json # Configuration settings (DB, logging, etc.)  
│-- Properties/  
│ ├── launchSettings.json # Debugging and environment settings

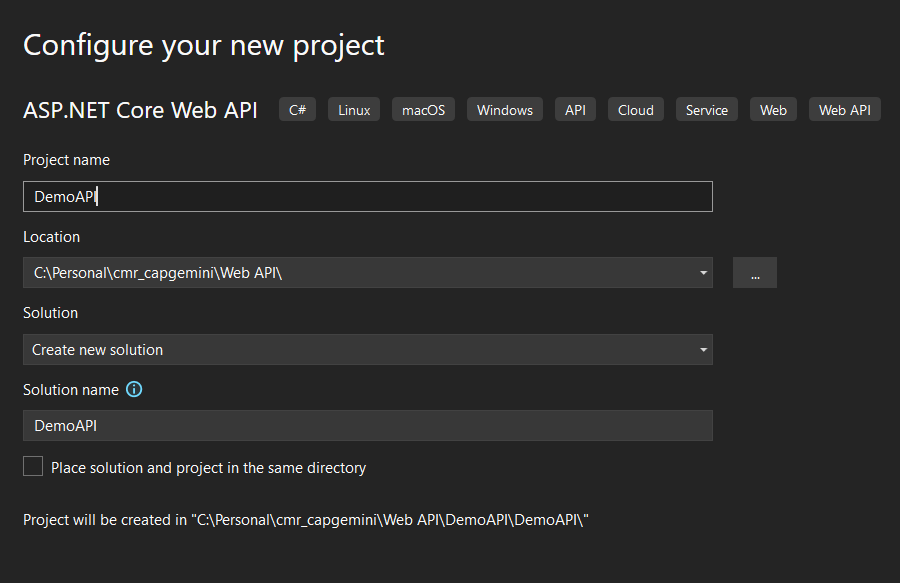
## 📌 Step-by-Step Guide to Creating a Simple ASP.NET Core 8.0 Web API

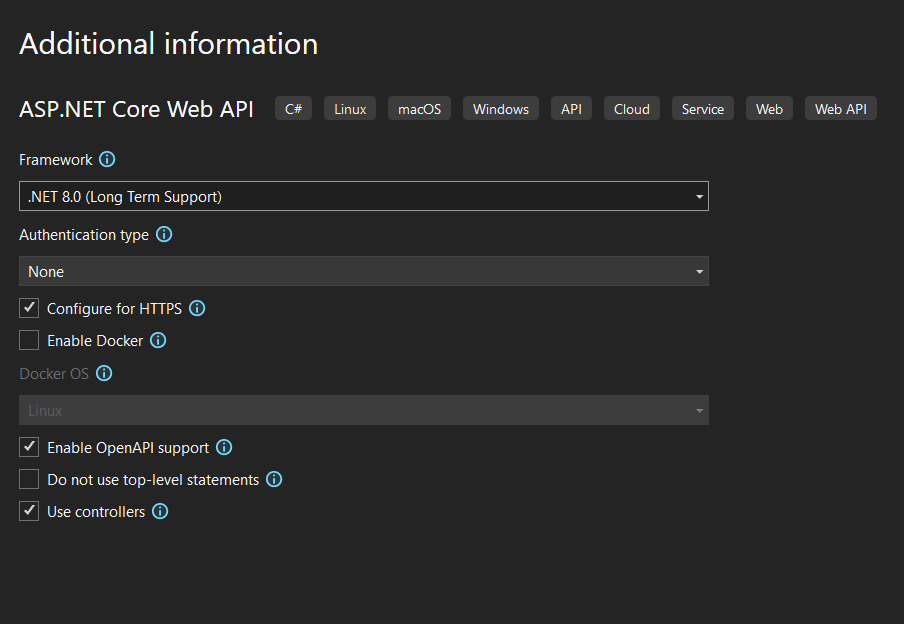
### 📝 Step 1: Create a New Web API Project

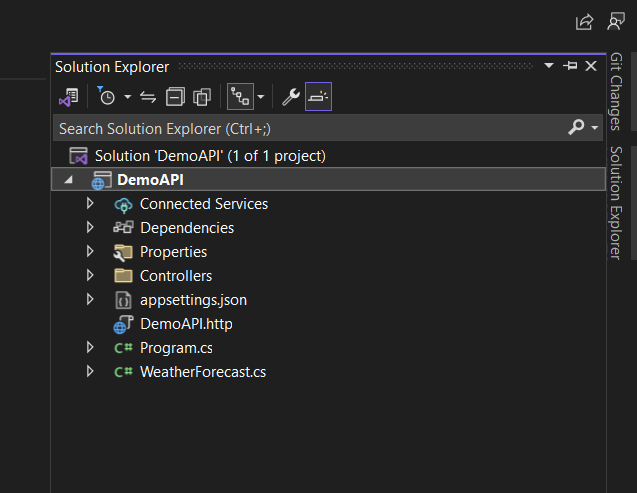
Run the following command in CLI (Command Line Interface):

dotnet new webapi -n MyWebAPI  
cd MyWebAPI  
dotnet run









### 📝 Step 2: Create the Model (Product.cs)

Location: Models/Product.cs

namespace MyWebAPI.Models  
{  
 public class Product  
 {  
 public int Id { get; set; }  
 public string Name { get; set; }  
 public decimal Price { get; set; }  
 }  
}

### 📝 Step 3: Create the Controller (ProductController.cs)

Location: Controllers/ProductController.cs

using Microsoft.AspNetCore.Mvc;  
using MyWebAPI.Models;  
using System.Collections.Generic;  
using System.Linq;  
  
namespace MyWebAPI.Controllers  
{  
 [Route("api/products")]  
 [ApiController]  
 public class ProductController : ControllerBase  
 {  
 private static List<Product> products = new List<Product>  
 {  
 new Product { Id = 1, Name = "Laptop", Price = 1000 },  
 new Product { Id = 2, Name = "Phone", Price = 500 }  
 };  
  
 [HttpGet]  
 public ActionResult<IEnumerable<Product>> GetProducts()  
 {  
 return Ok(products);  
 }  
  
 [HttpGet("{id}")]  
 public ActionResult<Product> GetProduct(int id)  
 {  
 var product = products.FirstOrDefault(p => p.Id == id);  
 if (product == null)  
 return NotFound("Product not found");  
 return Ok(product);  
 }  
  
 [HttpPost]  
 public ActionResult<Product> AddProduct([FromBody] Product product)  
 {  
 product.Id = products.Count + 1;  
 products.Add(product);  
 return CreatedAtAction(nameof(GetProduct), new { id = product.Id }, product);  
 }  
  
 [HttpPut("{id}")]  
 public ActionResult UpdateProduct(int id, [FromBody] Product updatedProduct)  
 {  
 var product = products.FirstOrDefault(p => p.Id == id);  
 if (product == null)  
 return NotFound("Product not found");  
  
 product.Name = updatedProduct.Name;  
 product.Price = updatedProduct.Price;  
 return NoContent();  
 }  
  
 [HttpDelete("{id}")]  
 public ActionResult DeleteProduct(int id)  
 {  
 var product = products.FirstOrDefault(p => p.Id == id);  
 if (product == null)  
 return NotFound("Product not found");  
  
 products.Remove(product);  
 return NoContent();  
 }  
 }  
}

### 📝 Step 4: Configure Middleware in Program.cs

Location: Program.cs

var builder = WebApplication.CreateBuilder(args);  
  
builder.Services.AddControllers();  
builder.Services.AddEndpointsApiExplorer();  
builder.Services.AddSwaggerGen();  
  
var app = builder.Build();  
  
app.UseHttpsRedirection();  
app.UseAuthorization();  
app.MapControllers();  
  
app.UseSwagger();  
app.UseSwaggerUI();  
  
app.Run();

### 📝 Step 5: Run the API

Run the following command:

dotnet run

## 📌 Summary

✅ ASP.NET Core Web API is used for RESTful services.  
✅ Folder structure includes Controllers, Models, Services, Data, and Configuration files.  
✅ Implemented CRUD operations using Controllers.  
✅ Swagger UI helps in API testing.

## 🚀 Next Steps

Would you like:  
- Integrate with Entity Framework Core for Database Storage?  
- Secure API using JWT Authentication?  
- Deploy to Azure or Docker?  
Let me know how you'd like to extend this further! 🚀