How to Create an ASP.NET Core Web API with Swagger Documentation

**Introduction:**

In this document, we will guide you through the steps to create an ASP.NET Core Web API and configure Swagger for API documentation. Swagger is a popular tool that helps generate interactive API documentation, making it easier for developers to explore and understand your API endpoints.

**Prerequisites:**

Before you begin, make sure you have the following prerequisites installed on your system:

.NET Core SDK (version 3.1 or higher)

Visual Studio Code or Visual Studio (latest version)

**Step 1:** Create a new ASP.NET Core Web API project

Open your preferred development environment (Ex: Visual Studio).

Click on files menu > New > New Project > select ASP.Net Core Web API

New project will be created

**Step 2:** Define the API Controller and Actions

1. Open the project in your preferred code editor.
2. In the project folder, locate the "Controllers" directory.
3. Create a new class file, for example, "MyController.cs," and define it as an API controller by inheriting from ControllerBase:

using Microsoft.AspNetCore.Mvc;

namespace MyWebApi.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class MyController : ControllerBase

{

// Actions will be defined here

}

}

Define an HTTP Get action method that accepts two parameters:

[HttpGet(Name = "Example")]

public async Task<IActionResult> Get(string parameter1, string parameter2){

// Execute your desired functionality using the provided parameters

// Return an appropriate response

}

**Step 3**: Configure Routing and Dependencies

1. In the project folder, locate the "Startup.cs" file.
2. Open the file and locate the ConfigureServices method.
3. Add the following code to configure routing for your API:

services.AddControllers();

1. Save the file

**Step 4:** Test the Web API

1. Build your project to ensure there are no build errors.
2. Run the application using Visual Studio or the command line

dotnet run

1. Open a web browser or a tool like Postman to test the API.
2. Verify that the API executes the desired functionality and returns an appropriate response.

Conclusion:

By following the steps outlined in this document, you can create an ASP.NET Core Web API that accepts two parameters and executes the specified functionality. This allows you to build robust and flexible APIs to serve various client applications.