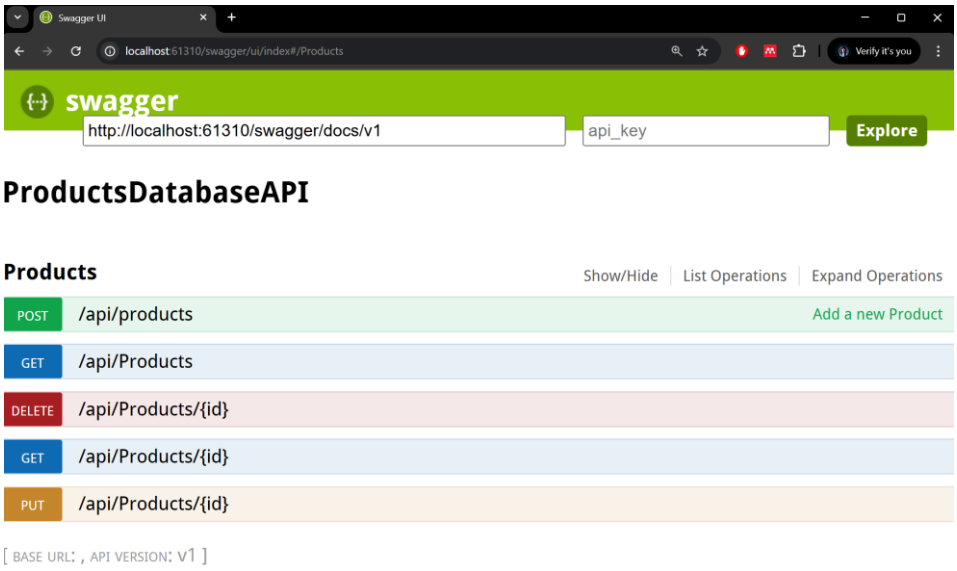
Documenting ASP.NET Web API with Swagger

Clear, comprehensive documentation is essential for the success of any API. Without it, developers can quickly become frustrated, leading to poor adoption and increased support overhead. To ensure your API is easy to understand and use, we'll demonstrate how to document an ASP.NET Web API using Swagger.



Swagger is a robust framework for designing, consuming, and visualizing RESTful APIs. It keeps your documentation, client libraries, and server code in perfect sync — automatically reflecting changes to methods, parameters, and models. This eliminates manual updates and ensures consistency across your development lifecycle.

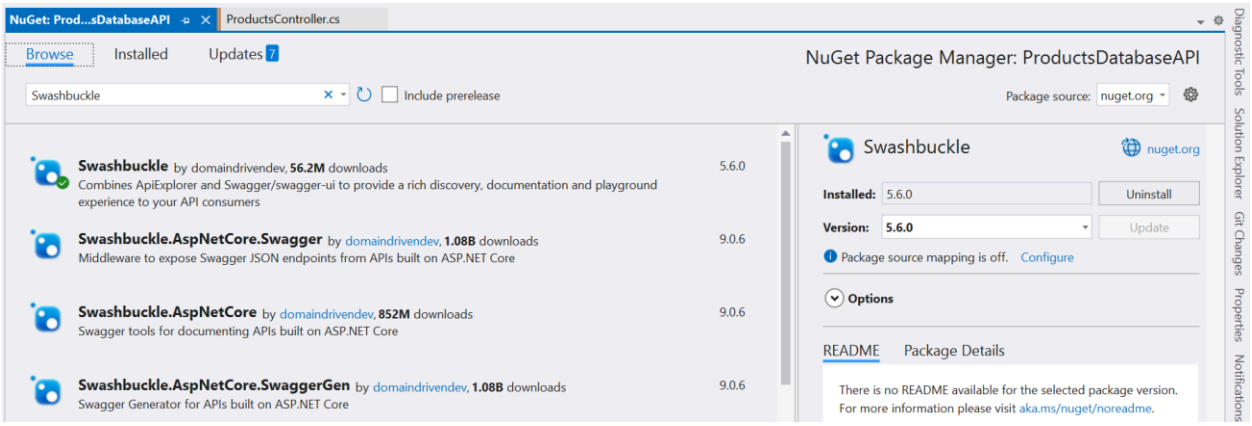
For ASP.NET Web API projects, **Swashbuckle** is the go-to Swagger implementation. It's lightweight, easy to integrate, and doesn't require any external dependencies. With just a few configuration steps, you can generate interactive, self-updating API docs that empower developers to explore and test endpoints directly from the browser.

1. **Adding Swagger to an ASP.NET Web API Project**

   Open the **NuGet Package Manager Console** (Tools->NuGet Package Manager) and run:

```
PM> Install-Package Swashbuckle
```

   alternatively use the visual studio **nuget manager** to install swashbuckle.



2. Enable XML Documentation

XML comments help provide detailed explanations for your API endpoints. To enable XML documentation:

1. Right-click your **project** in Solution Explorer → Select **Properties**.
2. Go to the **Build** tab.
3. Scroll down to the Output section, check the **XML documentation** filebox, and set the path to **bin\[YourProjectName].XML**.

This generates an XML file containing comments for your API methods and models.

3. Configure Swagger to Use XML Comments

Go to the `SwaggerConfig.cs` file (in your project **App_Start** folder) and modify it to include XML comments in the Swagger documentation. Search for `c.IncludeXmlComments(GetXmlCommentsPath())` and uncomment. Add the last method shown in the code:

```
public class SwaggerConfig
{
    public static void Register()
    {
        Swashbuckle.Bootstrapper.Init(GlobalConfiguration.Configuration);
        SwaggerSpecConfig.Customize(c =>
        {
            c.IncludeXmlComments(GetXmlCommentsPath()); //uncomment this line….
        });
    }

    protected static string GetXmlCommentsPath()
    {
        return System.String.Format(@"{0}\bin\YourProjectName.XML",
System.AppDomain.CurrentDomain.BaseDirectory);
    }
}
```

4. Add Annotations to API Methods and Models

Use XML comments (write ///) to describe your methods, parameters, and responses. Here's an example for a POST method in the `ProductController`:

```
/// <summary>
/// Add a new Product
/// </summary>
/// <param name="product">Product Model</param>
/// <remarks>Insert a new product into the system</remarks>
/// <response code="400">Bad request</response>
/// <response code="500">Internal Server Error</response>
/// <returns></returns>
// POST: api/Products
[Route("api/products")]
[ResponseType(typeof(Product))]
public IHttpActionResult Post([FromBody]Product product)
{
    //implementation goes here

    return BadRequest();
}
```

Swagger will automatically include these comments in the interactive documentation.