codewithmukesh

setup your auth in minutes!🚀

SIMPLIFIED AUTH IN .NET 8

mukesh murugan
@iammukeshm

codewithmukesh

Context

You are building a simple POC application on .NET 8, where you want to quickly attach a robust Auth System without worrying much.

These are the minimal packages you need to install on your shiny new .NET 8 Web API project.

```
<ItemGroup>
  <PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore"
  <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="8.0.1" />
  <PackageReference Include="Microsoft.EntityFrameworkCore" Version="8.0.1" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.InMemory" Version="8.
  <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
</ItemGroup>
```

.NET 8 introduces a new set of Identity API endpoints and support for token-based authentication.

Let’s see how to quickly set this up!

mukesh murugan
@iammukeshm

codewithmukesh

~8 Lines of Changes!

These are the only changes needed to setup a complete auth system in your .NET 8 application with Bearer Token Support.

Program.cs

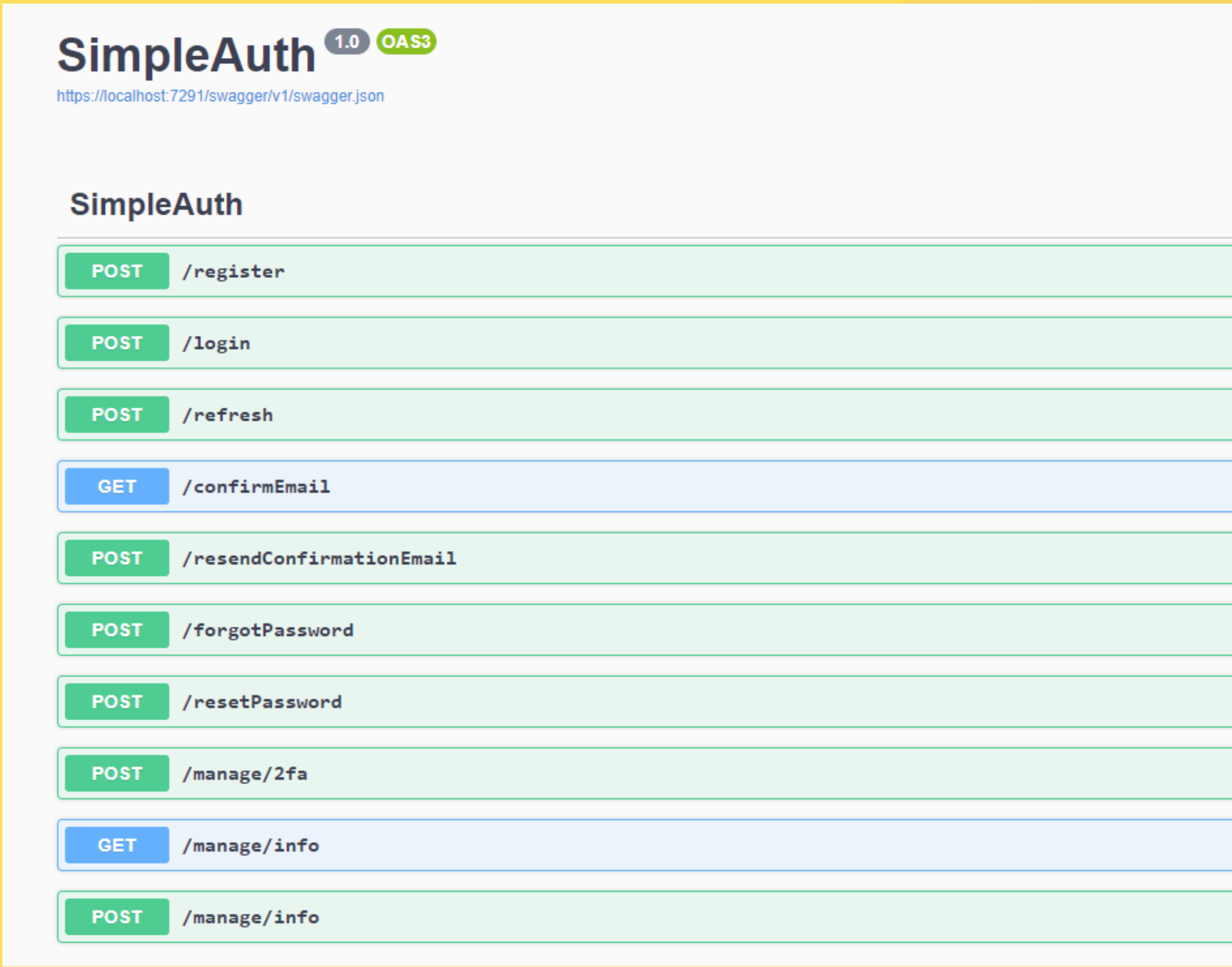
```
//configure database - in memory for demo purposes
builder.Services
    .AddDbContext<IdentityDbContext>(options => options.UseInMemoryDatabase("

//add auth scheme
builder.Services
    .AddAuthentication()
    .AddBearerToken(IdentityConstants.BearerScheme);

builder.Services
    .AddAuthorizationBuilder();

//add identity and store (ef)
builder.Services
    .AddIdentityCore<IdentityUser>()
    .AddEntityFrameworkStores<IdentityDbContext>()
    .AddApiEndpoints();

var app = builder.Build();

//add identity endpoints
app.MapIdentityApi<IdentityUser>();
```

mukesh murugan
@iammukeshm

codewithmukesh

New Auth Endpoints!

Once the changes are done, simply run your application, and open up Swagger UI. You can see the new Auth Endpoints.



mukesh murugan
@iammukeshm

codewithmukesh

User Registration

First, the registration endpoint, where we will have to pass email/pass to register a new user.

POST https://localhost:7291/register

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "email": "hello@codewithmukesh.com",
3   "password": "123Pa$$word!"
4 }
```

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize Text

mukesh murugan
@iammukeshm

codewithmukesh

Token Generation / Login

When you pass in **valid email/password** to the login endpoint, you will get an access token, expiry seconds, and refresh token.

POST https://localhost:7291/login

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
  "email": "hello@codewithmukesh.com",
  "password": "123Pa$$word!"
}
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "tokenType": "Bearer",
    "accessToken":
        "CfDJ8H0IjfCUJMpDqt6rgGRfeOyloNrcZ3LC5KElgDe2uREVj-3t9idJAzgDLk
        OPaQgy1u9gBt1nh-EybtpBcqB5Qo3IdtqEolG4RjuEhp1jD_KkPuD3nP4RSEQbr
        5Ssy094DBkdQ50CRHWBPtn8bDCFTOX90WEkE5ZHa4QPrHyD29qeXg4f15zt1KnC
        tjp19OGXWDSj7fT3zWbXowDdIqg6bIclL0p5yst1UUNxlg_P0vE0lSO1Exep2Ei
        CkRVjSNJ_ryHL7rIxrGnpD8RVdC3L_OuY30tmTNNfyyQA2t6mleJbQCstkR3wHH
        vYcyc5dHtclaTsXue7gRuJnP-hbXJdvXowiDdDpkQ-adQo4nzoCNqZ12t6XoAUI
        us8KFpNadNR6KdZwPYC_osFtaphqHY6ASgDCEg8gl-gSQ8yksT03oEwlZkN9yVx
    "expiresIn": 3600,
    "refreshToken":
        "CfDJ8H0IjfCUJMpDqt6rgGRfeOzGNUDPMQZufO6uwEItUdjTqum2lARrjNAprN
```

mukesh murugan
@iammukeshm

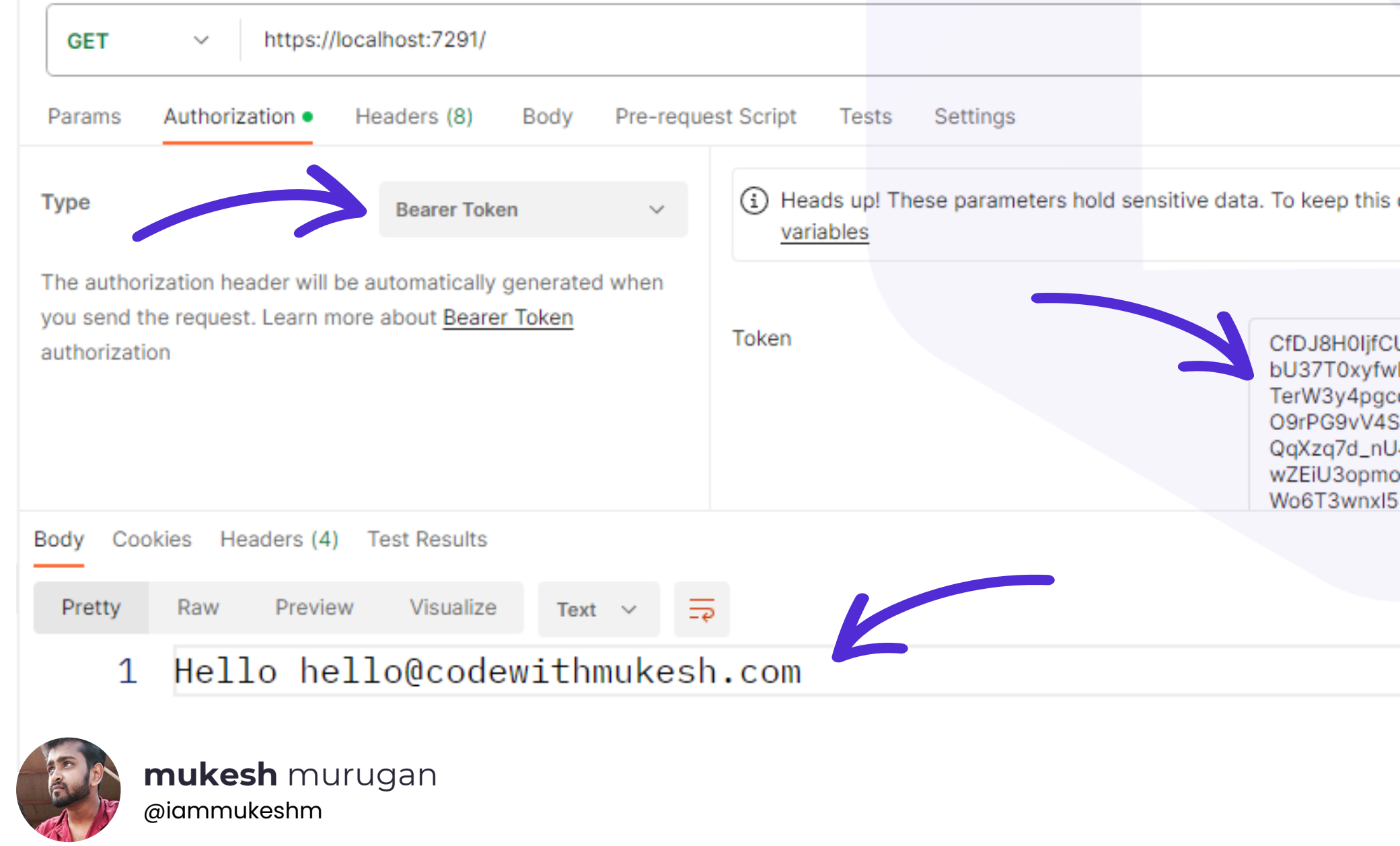
codewithmukesh

Testing

For testing this, I added a **secured Minimal API** that returns a simple Hello message, but with the email of the user. Note that you need to pass the token generated from /login endpoint to the Authorization header of the new ***secured*** endpoint.

C# Program.cs

```
app.MapGet("/", (ClaimsPrincipal claim) => $"Hello {claim.Identity?.Name}")
    .RequireAuthorization();
```



mukesh murugan
@iammukeshm

This is super helpful for small-scale applications. However, there are some limitations.

Limitations.

- The best use case is when your application is simple and doesn't need much customization in Identity. Anything larger than a simple application, things related to Identity might get tough with this approach.
- No control over endpoints.
- Cannot handle advanced Identity Workflows, like firing some background job post user registration, etc.
- However, there are chances that these limitations can be fixed in the later versions, maybe with some event-driven approach.

mukesh murugan
@iammukeshm

codewithmukesh

dotnet

Subscribe to my .NET Newsletter!

I will be starting a **.NET 8 Zero to Hero** Series soon via my Newsletter, for **FREE**.

Join the waitlist by Subscribing.

Link in the Description!

mukesh murugan
@iammukeshm

codewithmukesh

WAS THIS HELPFUL?

Share with a friend who needs it!

mukesh murugan
@iammukeshm

Follow me for more!