Microservices - JWT

Program.cs > Program > <top-level-statements-entry-point>

```cs
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = builder.Configuration["Jwt:Issuer"],
        ValidAudience = builder.Configuration["Jwt:Audience"],
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))
    };
});

builder.Services.AddAuthorization();

builder.Services.AddControllers();

var app = builder.Build();

app.UseHttpsRedirection();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllers();

app.Run();
```

TestController.cs > ...

```cs
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("api/[controller]")]
0 references
public class TestController : ControllerBase
{
    [HttpGet("hello")]
    0 references
    public IActionResult Hello() => Ok("Hello from test controller!");
}
```

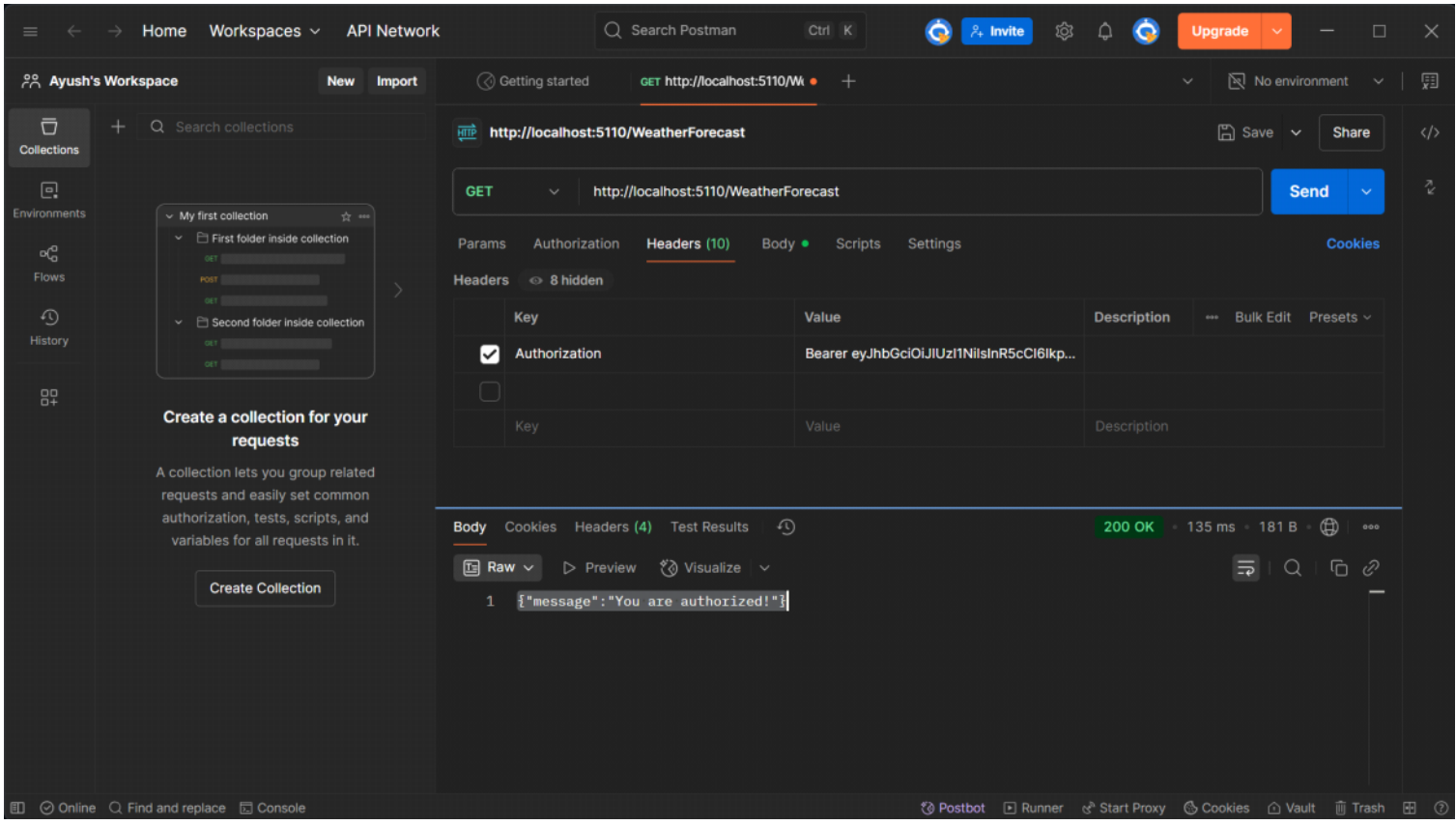
WeatherForecastController.cs > ...

```cs
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("[controller]")]
[Authorize]
0 references
public class WeatherForecastController : ControllerBase
{
    [HttpGet]
    0 references
    public IActionResult Get()
    {
        return Ok(new { Message = "You are authorized!" });
    }
}
```

AuthController.cs > AuthController > IsValidUser

```cs
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

[ApiController]
[Route("api/[controller]")]
1 reference
public class AuthController : ControllerBase
{
    1 reference
    private readonly IConfiguration _configuration;

    0 references
    public AuthController(IConfiguration configuration)
    {
        _configuration = configuration;
    }

    [HttpPost("login")]
    0 references
    public IActionResult Login([FromBody] LoginModel model)
    {
        if (IsValidUser(model))
        {
            var token = GenerateJwtToken(model.Username);
            return Ok(new { Token = token });
        }
        return Unauthorized();
    }

    1 reference
    private bool IsValidUser(LoginModel model)
    {
        return model.Username == "testuser" && model.Password == "password123";
    }

    1 reference
    private string GenerateJwtToken(string username)
    {
        var claims = new[]
        {
            new Claim(ClaimTypes.Name, username)
        };

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            issuer: _configuration["Jwt:Issuer"],
            audience: _configuration["Jwt:Audience"],
            claims: claims,
            expires: DateTime.UtcNow.AddMinutes(double.Parse(_configuration["Jwt:DurationInMinutes"])),
            signingCredentials: creds
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

```
PS C:\Users\Lenovo\MySecureApi> dotnet run
Using launch settings from C:\Users\Lenovo\MySecureApi\Properties\launchSettings.json...
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5110
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Lenovo\MySecureApi
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Users\Lenovo\MySecureApi
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
      Failed to determine the https port for redirect.
```



WebApi_Handson

Program.cs > ...

```
using System;
using System.Threading.Tasks;

0 references
class Program
{
    0 references
    static async Task Main(string[] args)
    {
        if (args.Length > 0 && args[0].Equals("Producer", StringComparison.OrdinalIgnoreCase))
        {
            await Producer.Main(args);
        }
        else if (args.Length > 0 && args[0].Equals("Consumer", StringComparison.OrdinalIgnoreCase))
        {
            Consumer.Main(args);
        }
        else
        {
            Console.WriteLine("Please specify 'Producer' or 'Consumer'");
        }
    }
}
```

Producer.cs > ...

```
using Confluent.Kafka;
using System;
using System.Threading.Tasks;

1 reference
class Producer
{
    1 reference
    public static async T
    {
        var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

        using var producer = new ProducerBuilder<Null, string>(config).Build();

        Console.WriteLine("Enter messages to send to Kafka (type 'exit' to quit):");

        while (true)
        {
            var message = Console.ReadLine();
            if (message?.ToLower() == "exit") break;

            await producer.ProduceAsync("chat-app", new Message<Null, string> { Value = message });
            Console.WriteLine($"Sent: {message}");
        }
    }
}
```

ProducerConfig.ProducerConfig() (+ 2 overloads)

Initialize a new empty ProducerConfig instance.

Consumer.cs > ...

```
using Confluent.Kafka;
using System;
using System.Threading;

1 reference
class Consumer
{
    1 reference
    public static void Main(string[] args)
    {
        var config = new ConsumerConfig
        {
            BootstrapServers = "localhost:9092",
            GroupId = "chat-consumer-group",
            AutoOffsetReset = AutoOffsetReset.Earliest
        };

        using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();
        consumer.Subscribe("chat-app");

        Console.WriteLine("Listening for messages...");
        while (true)
        {
            var cr = consumer.Consume(CancellationToken.None);
            Console.WriteLine($"Received: {cr.Message.Value}");
        }
    }
}
```

```
PS C:\Users\Lenovo\KafkaChatApp> dotnet run -- Producer
>>
Listening for messages...
Received: messages
Received: hello it is me
```

```
PS C:\Users\Lenovo\KafkaChatApp> dotnet build
Restore complete (0.5s)
  KafkaChatApp succeeded (0.2s) → bin\Debug\net9.0\KafkaChatApp.dll

Build succeeded in 1.0s
PS C:\Users\Lenovo\KafkaChatApp> dotnet run -- Consumer
>>
Listening for messages...
```