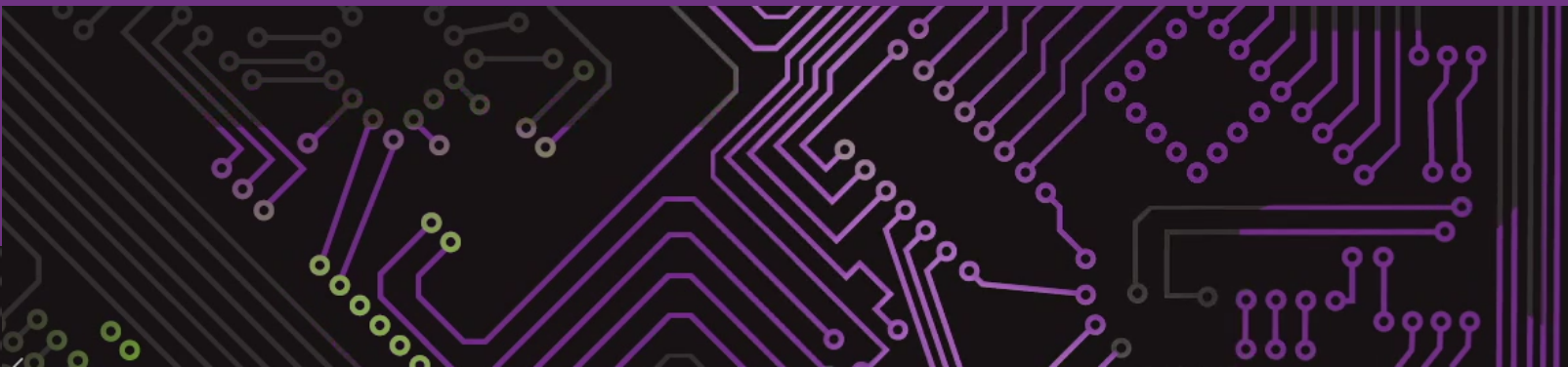
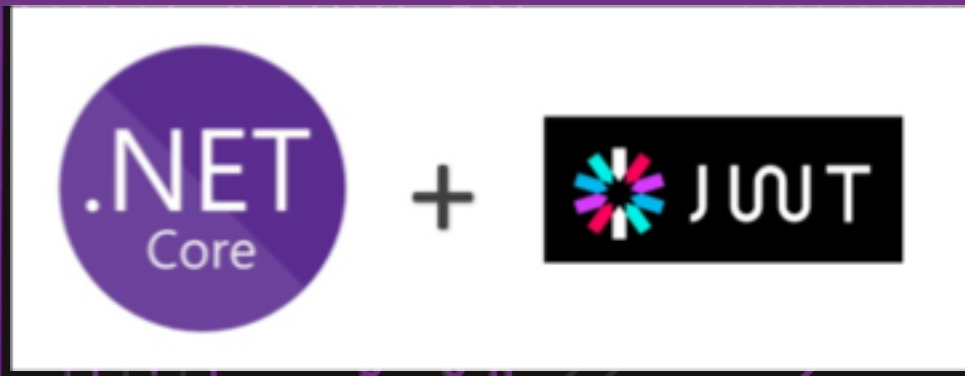
Secure your ASP.NET Core Web API with JWT(JSON Web Token)

Arthur





Overview

- What is JWT
- When should you use JWT
- How do JWT work
- JWT + ASP.NET Core(Web API)

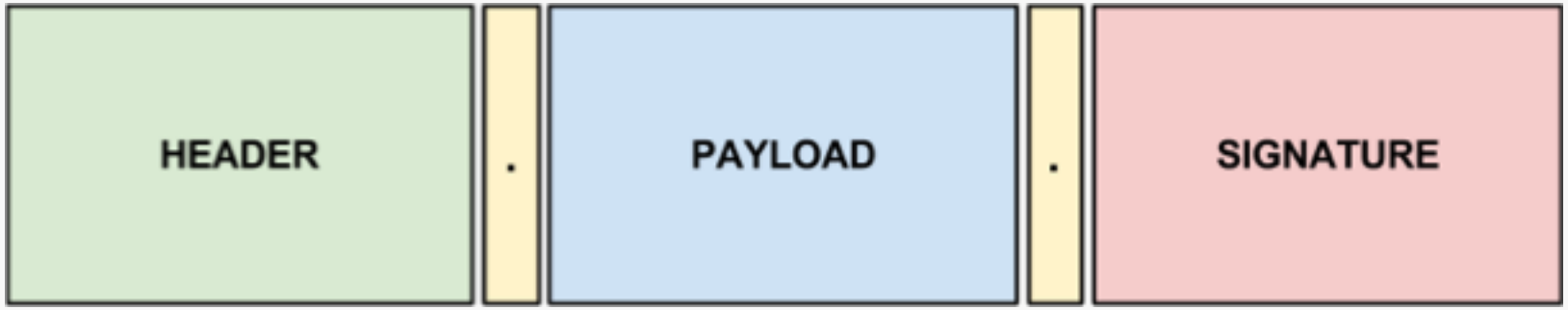
What's JSON Web Token?

- JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the **HMAC** algorithm) or a public/private key pair using **RSA** or **ECDSA**.

Source: JWT official website(https://jwt.io/introduction/)

JWT Structure

JWT consist of three parts separated by dots(.), which are:



Therefore, a JWT typically looks like the following, and it is base64 encoded.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

JWT Structure - cont.

- Header

The header typically consists of two parts: the type of the token, which is JWT, and the hashing algorithm being used, such as HMAC SHA256 or RSA.

For example:

```
{
 "alg": "HS256",
 "typ": "JWT"
}
```

JWT Structure - cont.

- Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims.

For example:

```
{
"sub": "1234567890",
"name": ”Arthur Xu",
"admin": true
}
```

JWT Structure - cont.

- Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

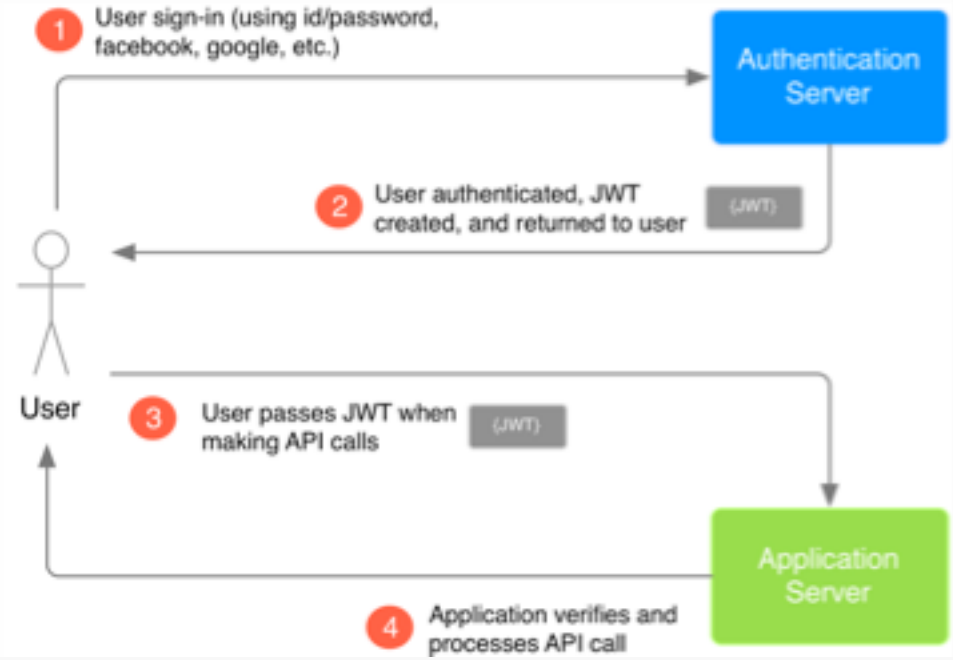
For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:

```
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)
```

When should you use JSON Web Tokens?

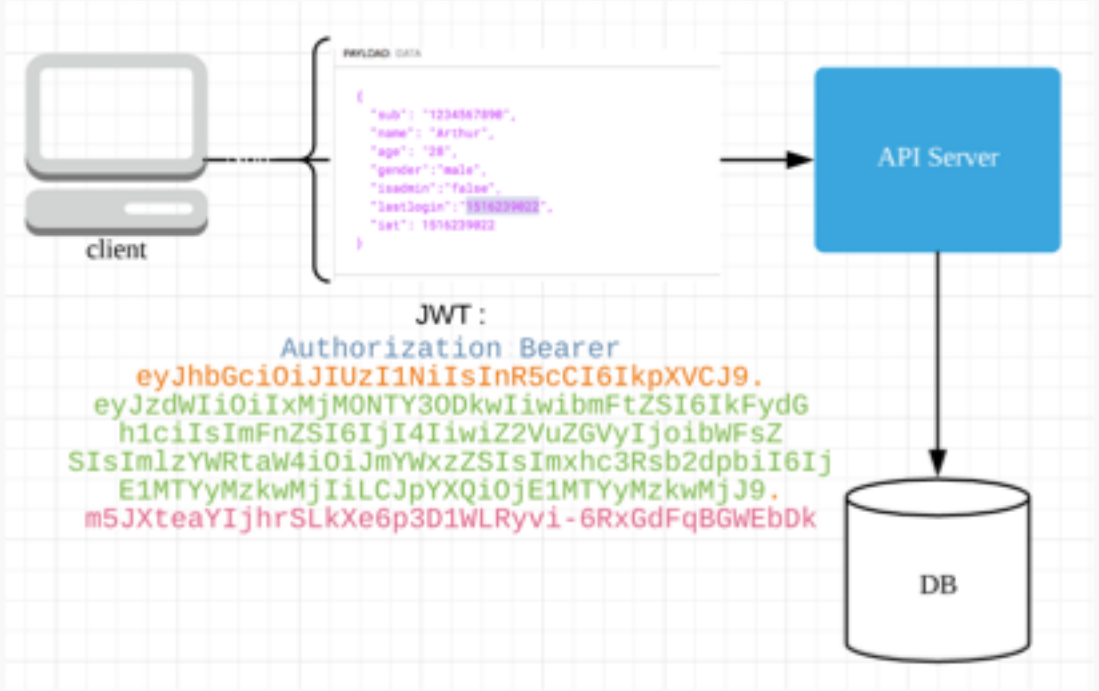
Authorization

Once user login, each subsequent request will include the JWT, allowing user to access services and resources



Information Exchange

a good way of securely transmitting information between parties, for instance, using public/private key pairs—you can be sure the senders are who they say they are.



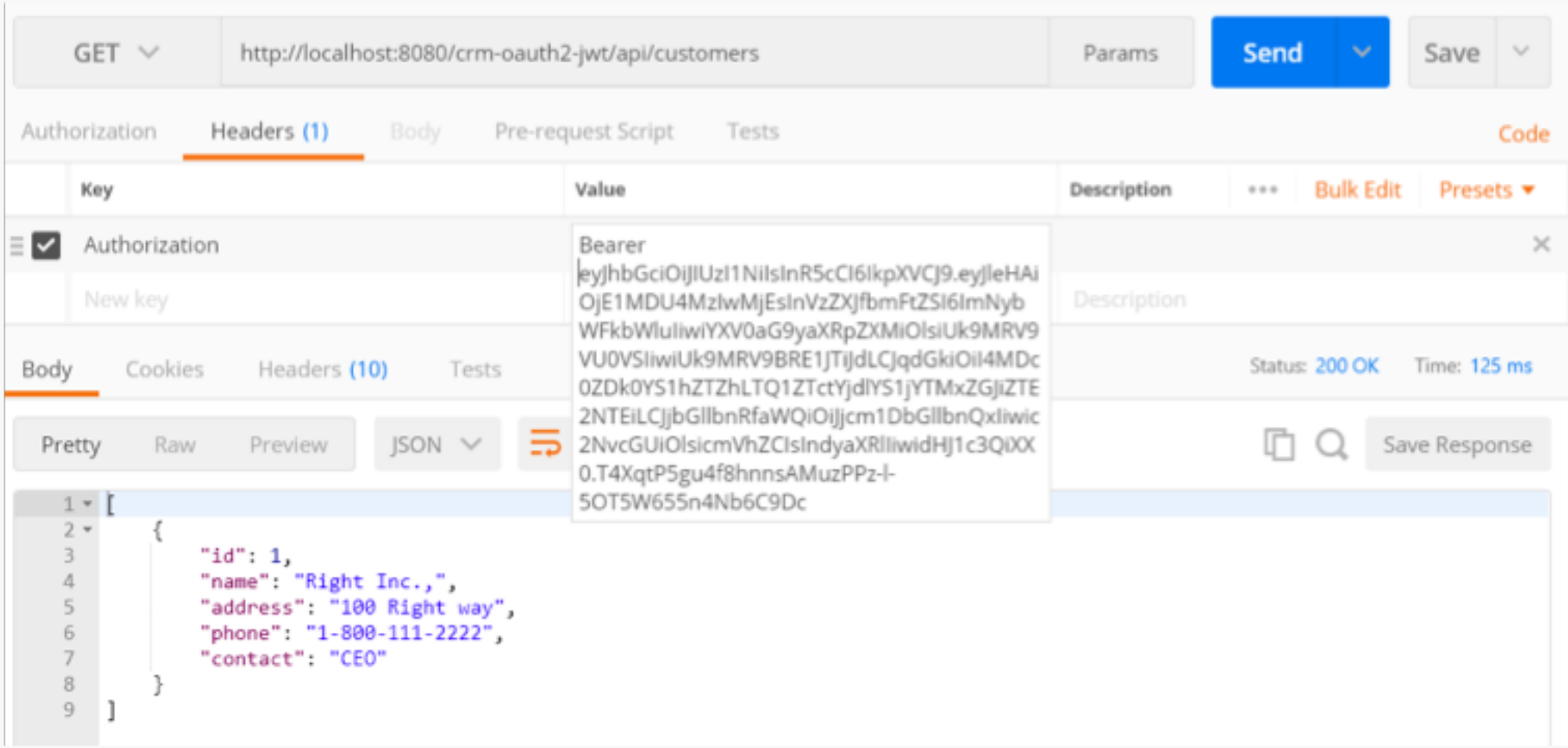
How do JWT work?

Bearer authentication (also called **token authentication**) is an HTTP authentication scheme that involves security tokens called bearer tokens. The name “Bearer authentication” can be understood as “give access to the bearer of this token.” The bearer token is a cryptic string, usually generated by the server in response to a login request. The client must send this token in the `Authorization` header when making requests to protected resources:

Authorization: Bearer <token>

Source: https://swagger.io/docs/specification/authentication/bearer-authentication/

JWT Request Example(Postman)



Demo: ASP.NET Core Web API with JWT Authentication

Token Configuration

- appsettings.json

```
"Authentication": {
  "JwtBearer": {
    "SecurityKey": "SampleCommerceApp_8BBA3490-23C8-4A99-A381-A142E0213335",
    "Issuer": "SampleCommerceApp",
    "Audience": "SampleCommerceApp"
  }
},
```

- Get Configuration in Startup

```
services.AddSingleton<TokenAuthConfiguration>(x =>
{
    var tokenConfig = new TokenAuthConfiguration()
    {
        SecurityKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(Configuration["Authentication:JwtBearer:SecurityKey"]
        Issuer = Configuration["Authentication:JwtBearer:Issuer"],
        Audience = Configuration["Authentication:JwtBearer:Audience"],
        Expiration = TimeSpan.FromDays(1)
    };
    tokenConfig.SigningCredentials = new SigningCredentials(tokenConfig.SecurityKey, SecurityAlgorithms.HmacSha256);
    return tokenConfig;
});
```

Generate Token in ASP.NET Core

```
1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
private string CreateAccessToken(IEnumerable<Claim> claims, TimeSpan? expiration = null)
{
    var now = DateTime.UtcNow;

    var jwtSecurityToken = new JwtSecurityToken(
        issuer: _configuration.Issuer,
        audience: _configuration.Audience,
        claims: claims,
        notBefore: now,
        expires: now.Add(expiration ?? _configuration.Expiration),
        signingCredentials: _configuration.SigningCredentials
    );

    return new JwtSecurityTokenHandler().WriteToken(jwtSecurityToken);
}

1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
private static List<Claim> CreateJwtClaims(ClaimsIdentity identity)
{
    var claims = identity.Claims.ToList();
    var nameIdClaim = claims.First(c => c.Type == ClaimTypes.NameIdentifier);

    // Specifically add the jti (random nonce), iat (issued timestamp), and sub (subject/user) claims.
    claims.AddRange(new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, nameIdClaim.Value),
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
        new Claim(JwtRegisteredClaimNames.Iat, DateTimeOffset.Now.ToUnixTimeSeconds().ToString(), ClaimValueTypes.Integer64)
    });

    return claims;
}
```

Authenticate Action

```
[HttpGet]
[ActionName("auth")]
0 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
public IActionResult Authenticate(string userId, string password)
{
    var user = _userService.SignIn(new Dto.UserDto() {
        UserId = userId,
        Password = password
    });

    if (user!= null)
    {
        var identity = new ClaimsIdentity();
        identity.AddClaim(new Claim(ClaimTypes.NameIdentifier, user.UserId.ToString()));
        identity.AddClaim(new Claim(ClaimTypes.Name, user.UserName));
        identity.AddClaim(new Claim(ClaimTypes.Email, user.Email));

        //return Token
        var accessToken = CreateAccessToken(CreateJwtClaims(identity));

        var result = new GenericAPIResponse()
        {
            Success = true,
            Result = new AuthenticateResultModel()
            {
                AccessToken = accessToken,
                UserId = user.UserId,
                ExpireInSeconds = (int)_configuration.Expiration.TotalSeconds,
            }
        };

        return Ok(result);
    }
    else
    {
        return Unauthorized();
    }
}
```

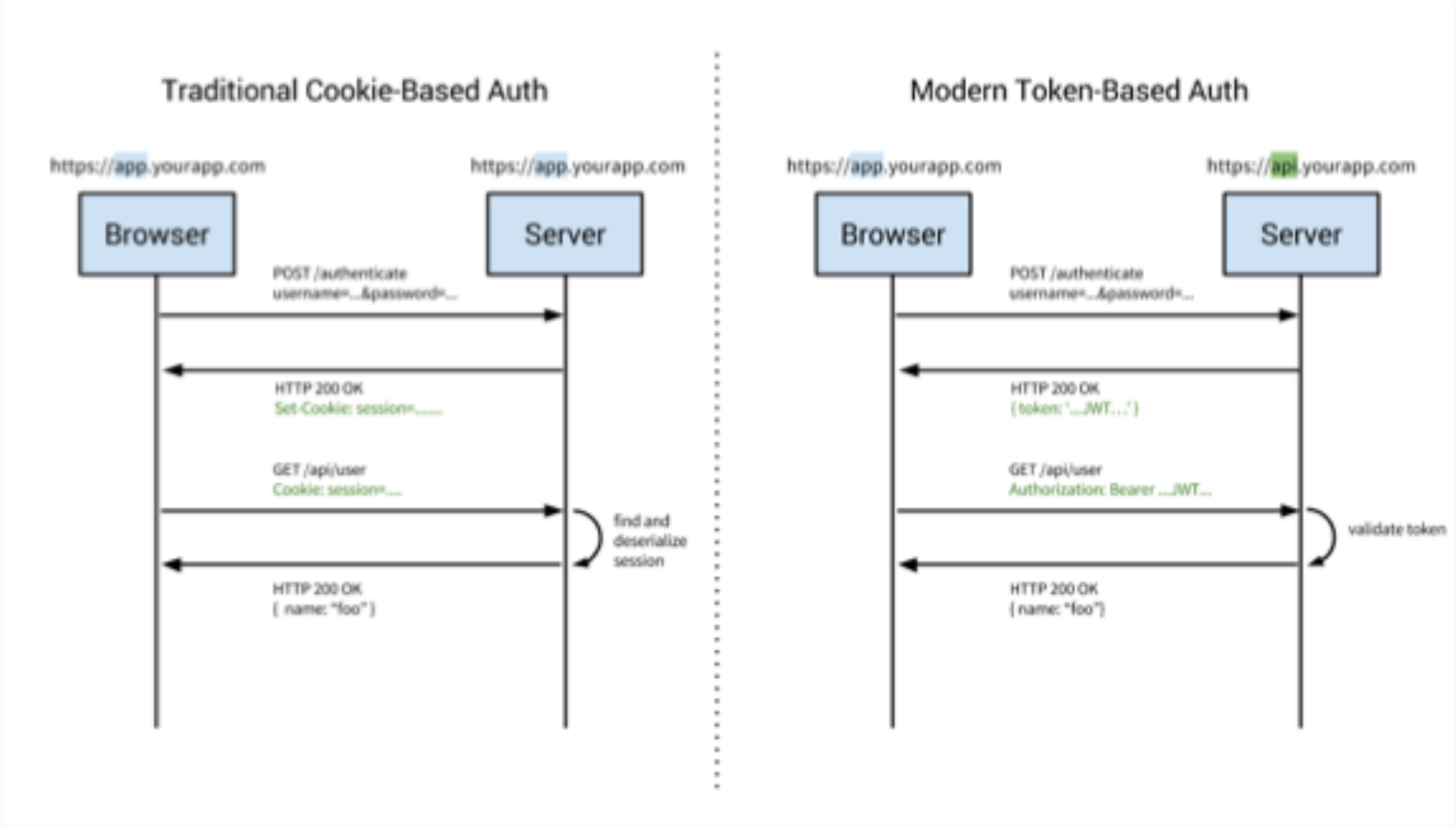
Token Validation In Product API

Use JWT middleware to enable JWT Authentication in ASP.NET Core API, just make sure the JWT configuration is same as Authentication API

```
1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
public static void Configure(IServiceCollection services, IConfiguration configuration)
{
        services.AddAuthentication(options => {
            options.DefaultAuthenticateScheme = "JwtBearer";
            options.DefaultChallengeScheme = "JwtBearer";
        }).AddJwtBearer("JwtBearer", options =>
        {
            options.Audience = configuration["Authentication:JwtBearer:Audience"];

            options.TokenValidationParameters = new TokenValidationParameters
            {
                // The signing key must match!
                ValidateIssuerSigningKey = true,
                IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(configuration["Authentication:JwtBearer:SecurityKe

                // Validate the JWT Issuer (iss) claim
                ValidateIssuer = true,
                ValidIssuer = configuration["Authentication:JwtBearer:Issuer"],

                // Validate the JWT Audience (aud) claim
                ValidateAudience = true,
                ValidAudience = configuration["Authentication:JwtBearer:Audience"],

                // Validate the token expiry
                ValidateLifetime = true,

                // If you want to allow a certain amount of clock drift, set that here
                ClockSkew = TimeSpan.Zero
            };
        });
}
```

Session vs Token

Doesn’t support mobile application



Stateless!

Sample Code

- GitHub
  - https://github.com/xuyae573/JWTNetCoreSample

- Tools:
  - Swagger UI
  - Visual Studio 2017
  - NET Core SDK 2.0

References

- Web storage vs. cookies

  https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage

- ASP. NET Core 2.0 Web Api JWT Authentication with Identity & MySQL

  https://medium.com/@ozgurgul/asp-net-core-2-0-webapi-jwt-authentication-with-identity-mysql-3698eeba6ff8

- Authorize with a specific scheme in ASP.NET Core

  https://docs.microsoft.com/en-us/aspnet/core/security/authorization/limitingidentitybyscheme?view=aspnetcore-2.1&tabs=aspnetcore2x

- Securing ASP.NET Core 2.0 Applications with JWTs

  https://auth0.com/blog/securing-asp-dot-net-core-2-applications-with-jwts/

Q & A

Thank You