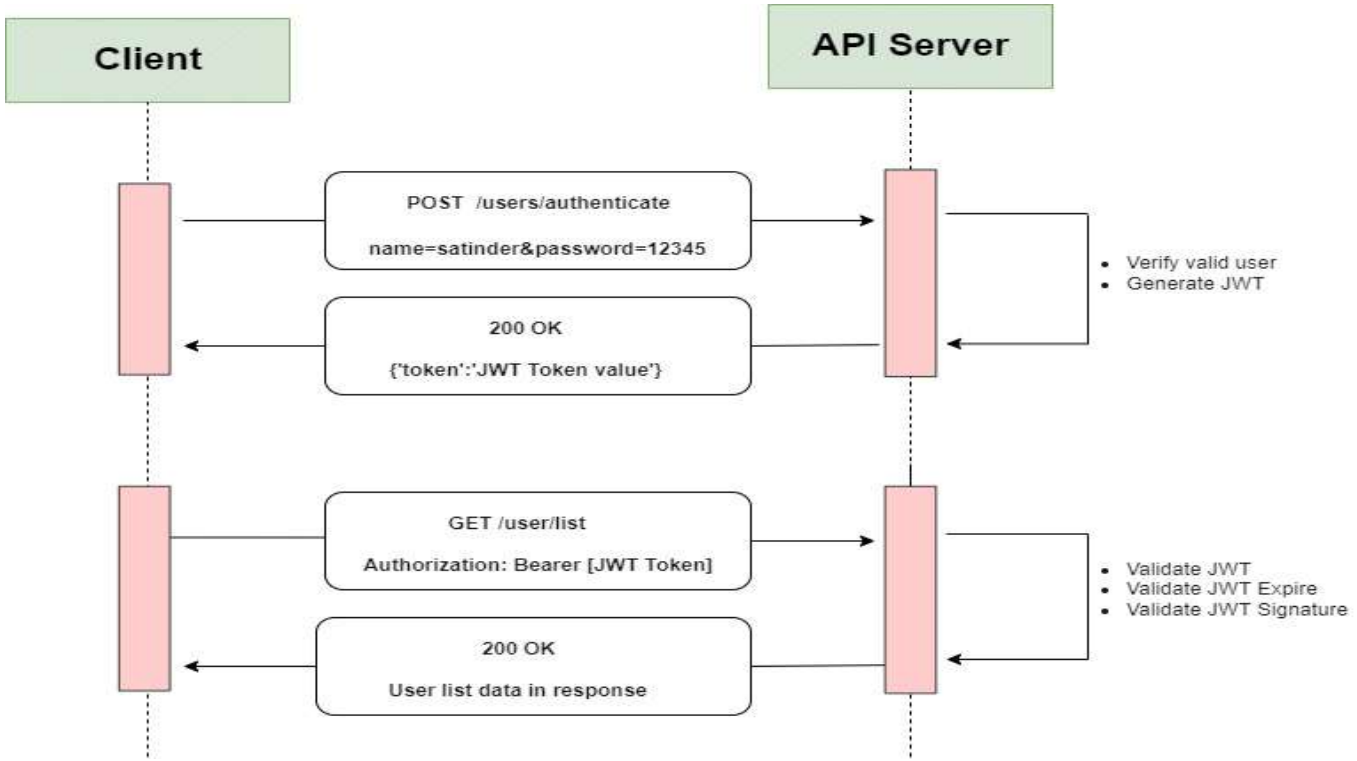
JWT

Json Web Tokens

JWT Authentication Workflow



Json Web Tokens

- JWT token is a string and has three parts separated by dot (.)
- a) Header b) Payload c) Signature
- Header & Payload are JSON objects
- Header contains algorithm & type of token which is jwt
- Payload contains claims (key/value pairs) + expiration date + aud/issuer etc.
- Signature is HASH value computed using Base64(Header) +"." + Base64(Payload). This information is passed to an algorithm with a secret key.
- Token structure is base64(header) + "." + base64(payload) + "." + hash

workflow using JWT

- Client sends a request to server for token
- Server generates a JWT (which contains a hash). Hash is generated using a secret key.
- Client receives the token and stores it somewhere locally.
- Client sends the token in future requests.
- Server gets the token from request header, computes Hash again by using a) Header from token b) payload from token c) secret key which server already has.
- If ("newly computed hash" = "hash came in token"), token is valid otherwise it is tempered or not valid.

JWT in ASP.NET Web API

(.Net Framework)

Creating JWT in ASP.NET Web API

- Add following nuget Package:

  System.IdentityModel.Tokens.Jwt

- Open Controller and add following namespaces:

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

Creating JWT in ASP.NET Web API

```
public object gettoken()
        {
            string key = "my_secret_key_12345"; //Secret key which will be used later during validation
            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));
            var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);
            var token = new JwtSecurityToken(
                            expires: DateTime.Now.AddDays(1),
                            signingCredentials: credentials);
            var jwt_token = new JwtSecurityTokenHandler().WriteToken(token);
            return new { data = jwt_token };
        }
```

```
public Object GetToken()
{
    string key = "my_secret_key_12345"; //Secret key which will be used later during validation
    var issuer = "http://mysite.com";  //normally this will be your site URL

    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));
    var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

    //Create a List of Claims, Keep claims name short
    var permClaims = new List<Claim>();
    permClaims.Add(new Claim("valid", "1"));
    permClaims.Add(new Claim("userid", "1"));
    permClaims.Add(new Claim("name", "bilal"));

    //Create Security Token object by giving required parameters
    var token = new JwtSecurityToken(issuer, //Issure
                    issuer,  //Audience
                    permClaims,
                    expires: DateTime.Now.AddDays(1),
                    signingCredentials: credentials);
    var jwt_token = new JwtSecurityTokenHandler().WriteToken(token);
    return new { data = jwt_token };
}
```

Validate JWT Token

Add the following nuget packages,

- Microsoft.Owin.Security.Jwt
- Microsoft.AspNet.WebApi.Owin
- Microsoft.Owin.Host.SystemWeb

Create Owin Statup class -> Right click on Web Project -> Add -> Owin Startup Class.

a) Add following namespaces

```
using Microsoft.Owin.Security.Jwt;
using Microsoft.Owin.Security;
using Microsoft.IdentityModel.Tokens;
using System.Text;
```

```
public void Configuration(IAppBuilder app)
        {
            app.UseJwtBearerAuthentication(
              new JwtBearerAuthenticationOptions
              {
                  AuthenticationMode = AuthenticationMode.Active,
                  TokenValidationParameters = new TokenValidationParameters()
                  {
                      ValidateIssuer = false,
                      ValidateAudience = false,
                      ValidateIssuerSigningKey = true,
                      IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("my_secret_key_12345"))
                  }
              });
        }
```

Test

```
[HttpPost]
public String GetName1() {
 if (User.Identity.IsAuthenticated) {
  var identity = User.Identity as ClaimsIdentity;
  if (identity != null) {
   IEnumerable < Claim > claims = identity.Claims;
  }
  return "Valid";
 } else {
  return "Invalid";
 }
}
```

JWT in ASP.NET Web API

(DotNet Core)

Creating JWT in ASP.NET Web API(validate method)

- Install package : -Microsoft.AspNetCore.Authentication.JwtBearer
- System.IdentityModel.Tokens.Jwt.

- In ConfigureServices

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateLifetime = true,
         ValidateAudience=false,
         ValidateIssuer=false,
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes
("welcome to my key"))
        };
    });
```

Creating JWT in ASP.NET Web API(validate method)

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters

        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = Configuration["Jwt:Issuer"],
            ValidAudience = Configuration["Jwt:Issuer"],
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetByt
es("welcome to my key”))
        };
    });
```

Creating JWT in ASP.NET Web API(validate method)

- Validate the server (ValidateIssuer = true) that generates the token.
- Validate the recipient of the token is authorized to receive (ValidateAudience = true)
- Check if the token is not expired and the signing key of the issuer is valid (ValidateLifetime = true)
- Validate signature of the token (ValidateIssuerSigningKey = true)
- Additionally, we specify the values for the issuer, audience, signing key. In this example, I have stored these values in appsettings.json file.

=> app.UseAuthentication() method in the Configure method of startup class

Generate JSON Web Token

```
    private string GenerateJSONWebToken(UserModel userInfo)
        {
            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.
GetBytes("welcome to my key"));
            var credentials = new SigningCredentials(securityKey, Sec
urityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(_config["Jwt:Issuer"],

              _config["Jwt:Issuer"],
              null,
              expires: DateTime.Now.AddMinutes(120),
              signingCredentials: credentials);

            return new JwtSecurityTokenHandler().WriteToken(token);

        }
```

Authorize Web Token

- [Authorize]
- [AllowAnonymous]

```
[HttpPost]
public String GetName1() {
 if (User.Identity.IsAuthenticated) {
  var identity = User.Identity as ClaimsIdentity;
  if (identity != null) {
   IEnumerable < Claim > claims = identity.Claims;
  }
  return "Valid";
 } else {
  return "Invalid";
 }
}
```

C# client

```
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
```

JavaSCript Client

```
Authorization: Bearer token
```