Web Application Programming Interface (API)

Tahaluf Training Center 2021

Chapter 08

1. Overview of Authentication
2. Create LOGIN using JWT Token
3. Overview of Authorization
4. Create Authorization

Overview of Authentication

Authentication happens in the host in Web API. The host is IIS, which uses HTTP methods for authentication. It allows to configure project for using any of the authentication modules built in to IIS, ASP.NET or write HTTP method to perform custom authentication.

HTTP Message Handlers for Authentication

Using the host for authentication used to put authentication logic into an HTTP Message Handler. In that case, the message handler sets the principal and examines the HTTP request.

When should use message handlers for authentication?

- ✓ An HTTP method sees all requests that go through the pipeline. A message handler sees only requests that are routed to Web API.

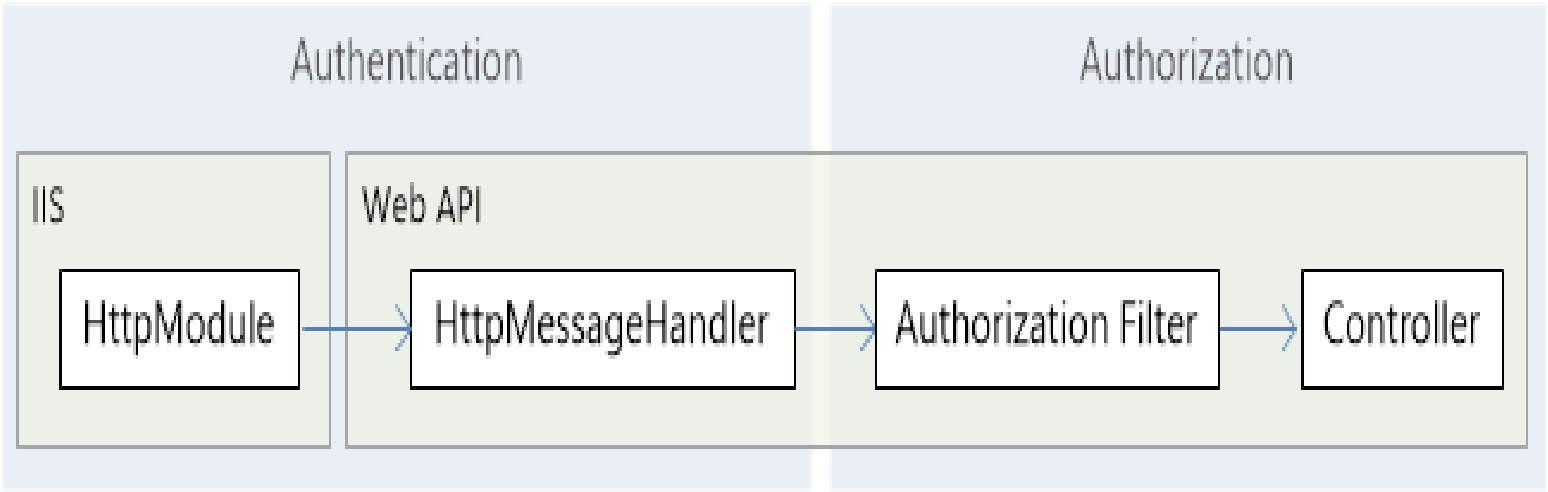
HTTP Message Handlers for Authentication

- ✓ Can set per route message handlers, which lets applying an authentication to a specific route.
- ✓ HTTP methods are specific to IIS. Message handlers are host agnostic, so they can be used with both self hosting and web hosting.

HTTP Message Handlers for Authentication

- ✓ HTTP methods run earlier in the pipeline. If authentication is handled in a message handler, the principal does not get set because the handler runs. The principal back to the previous principal when the response leaves the message handler.

Overview of Authentication

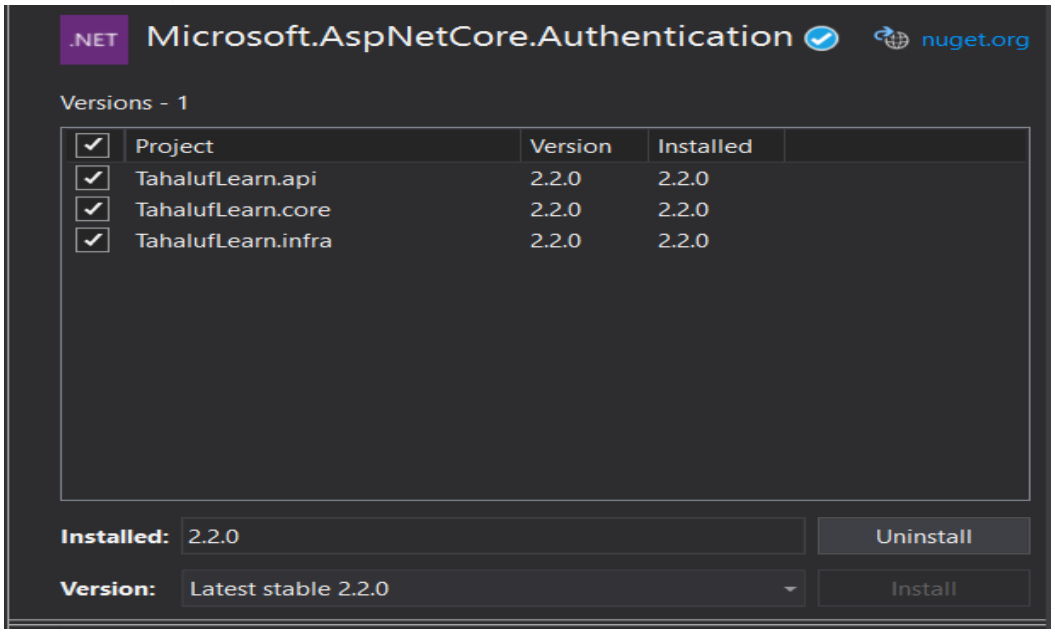


Create LOGIN using JWT Token

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Install the following libraries:

Microsoft.AspNetCore.Authentication

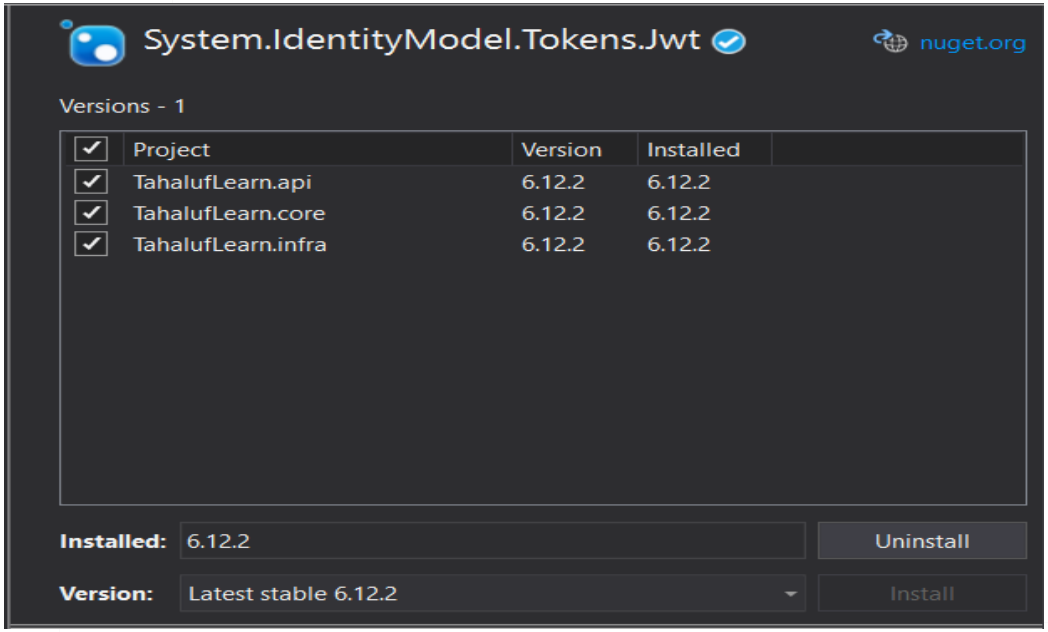


Create LOGIN using JWT Token

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Install the following libraries:

System.IdentityModel.Tokens.Jwt

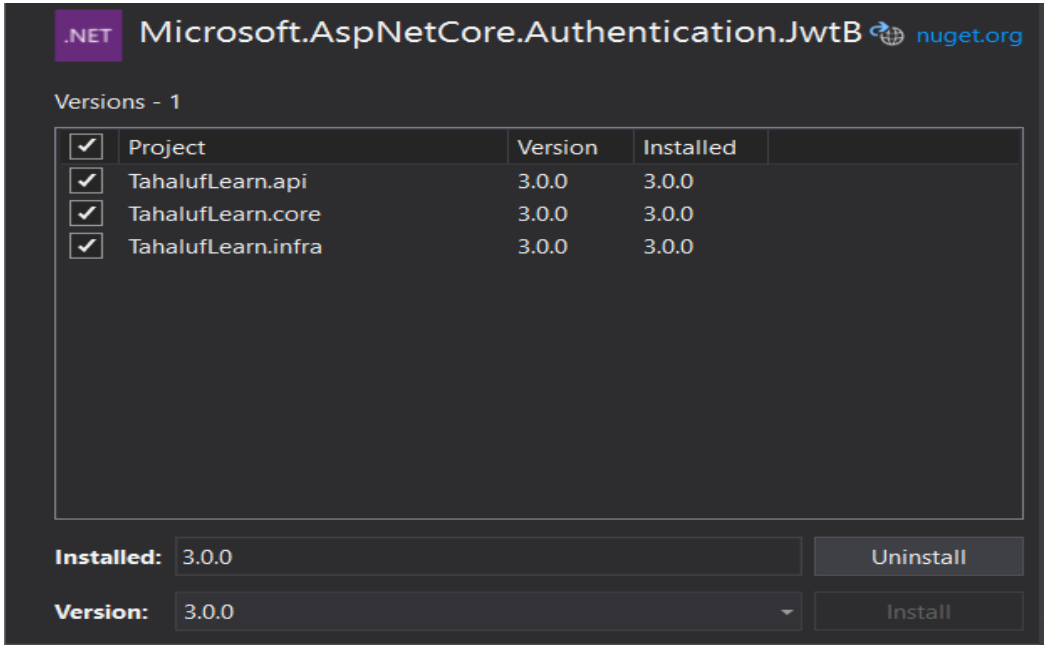


Create LOGIN using JWT Token

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Tools => NuGet Package Manager => Manage NuGet Packages for Solution => Install the following libraries:

Microsoft.AspNetCore.Authentication.JwtBearer



Create LOGIN using JWT Token

Create a Stored Procedure:

```
CREATE PROCEDURE [dbo].[LOGIN]
@UserName varchar(255),
@Password varchar(255)
as
select UserName, RoleName from Login
where UserName=@UserName and Password=@Password
```

Create JWTRepository in Tahaluf.LMS.Infra => Write the following code:

```
public class JwtRepository : IJwtRepository
{
        private readonly IDBContext dBContext;

        public JwtRepository(IDBContext _dBContext)
        {
            dBContext = _dBContext;
        }
```

Create LOGIN using JWT Token

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

```
public Login Auth(Login login)
{
var p = new DynamicParameters();
p.Add("@username", login.Username, dbType: DbType.String,
direction: ParameterDirection.Input);
p.Add("@password", login.Password, dbType: DbType.String,
direction: ParameterDirection.Input);
IEnumerable<Login> result =
dBContext.Connection.Query<Login>("createauth", p,
commandType: CommandType.StoredProcedure);
return result.FirstOrDefault();
}
```

Create IJwtRepository => in Tahaluf.LMS.Core => Write the following code:

```
public    interface IJwtRepository
{
     public Login Auth(Login login);
}
```

Startup.cs => Configure Service:

```
services.AddScoped<IJwtRepository, JwtRepository>();
```

Create JwtService => in Tahaluf.LMS.Infra => Write the following code:

```
public  class JwtService: IJwtService
{
        private readonly IJwtRepository jwtRepository;
        public JwtService(IJwtRepository _jwtRepository)
        {
            jwtRepository = _jwtRepository;
        }
```

Create LOGIN using JWT Token

```
public string Auth(Login login)
{

    var result = jwtRepository.auth(login);
    if (result == null)
    {
        return null;
    }
    else
    {
      var tokenHandler = new JwtSecurityTokenHandler();
      var tokenKey = Encoding.ASCII.GetBytes("[SECRET
      USED TO SIGN AND VERIFY JWT TOKENS, IT CAN BE ANY
      STRING]");
      var tokenDescriptor = new SecurityTokenDescriptor
```

Create LOGIN using JWT Token

```
{
Subject = new ClaimsIdentity(new Claim[]
{
        new Claim(ClaimTypes.Name, result.Username),
        new Claim(ClaimTypes.Role, result.Rolename),
}),
        Expires = DateTime.UtcNow.AddHours(1),
        SigningCredentials = new SigningCredentials(new
        SymmetricSecurityKey(tokenKey),
        SecurityAlgorithms.HmacSha256Signature)
};
        var token = tokenHandler.CreateToken(tokenDescriptor);
        return tokenHandler.WriteToken(token);
}
```

Create IJwtService => in Tahaluf.LMS.Core => Write the following code:

```
public interface IJwtService
{
        public string Auth(Login login);
}
```

Startup.cs => Configure Service:

```
services.AddScoped<IJwtService, JwtService >();
```

Startup.cs => Configure Service:

```
services.AddAuthentication(x =>
{
        x.DefaultAuthenticateScheme
        JwtBearerDefaults.AuthenticationScheme;
        x.DefaultChallengeScheme =
        JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
        y.RequireHttpsMetadata = false;
        y.SaveToken = true;
```

Create LOGIN using JWT Token

```
y.TokenValidationParameters = new
TokenValidationParameters
{
      ValidateIssuerSigningKey = true,
      IssuerSigningKey = new
      SymmetricSecurityKey(Encoding.ASCII.GetBytes
      ("[SECRET USED TO SIGN AND VERIFY JWT TOKENS, IT
      CAN BE ANY STRING]")),
      ValidateIssuer = false,
      ValidateAudience = false
};
});

 app.UseAuthentication();
```

Create JwtController:

```
[Route("api/[controller]")]
[ApiController]
public class JwtController : Controller
{
    private readonly IJwtService jwtService;
    public JwtController(IJwtService _jwtService)
    {
            jwtService = _jwtService;
    }
```

Create LOGIN using JWT Token

Create JwtController:

```
[HttpPost]
public IActionResult Authen([FromBody]Login login)
{
      var token = jwtService.Auth(login);
      if (token==null)
      {
           return Unauthorized();
      }
      else
      {
            return Ok(token);
      }
}
```

Overview of Authorization

Authorization allows an user to grant and restrict permissions on Website, data and functionality.

Authorization checks if a user is allowed to access to some functionality or perform an action. For example, having the permission to post data and get data is a part of authorization.

Create Authorization in the functions:

```
1. [Authorize("Admin")]
2. [Authorize("Teacher")]
```

References

[1]. https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/authentication-and-authorization-in-aspnet-web-api