Web Application Programming Interface (API)

Tahaluf Training Center 2021

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

Chapter 08

1 Authentication and Authorization in ASP.NET Web API

2 Create a JWT Token

Authentication and Authorization in ASP.NET Web API

Authentication is knowing the identity of the user. For example, Alice logs in with her username and password, and the server uses the password to authenticate Alice.

Authorization is deciding whether a user is allowed to perform an action. For example, Alice has permission to get a resource but not create a resource.

Authentication

Web API assumes that authentication happens in the host. For web-hosting, the host is IIS, which uses HTTP modules for authentication. You can configure your project to use any of the authentication modules built in to IIS or ASP.NET, or write your own HTTP module to perform custom authentication.

HTTP Message Handlers for Authentication

Instead of using the host for authentication, you can put authentication logic into an HTTP message handler. In that case, the message handler examines the HTTP request and sets the principal.

HTTP Message Handlers for Authentication

- An HTTP module sees all requests that go through the ASP.NET pipeline. A message handler only sees requests that are routed to Web API.
- You can set per-route message handlers, which lets you apply an authentication scheme to a specific route.
- HTTP modules are specific to IIS. Message handlers are host-agnostic, so they can be used with both web-hosting and self-hosting.

HTTP Message Handlers for Authentication

- HTTP modules participate in IIS logging, auditing, and so on.

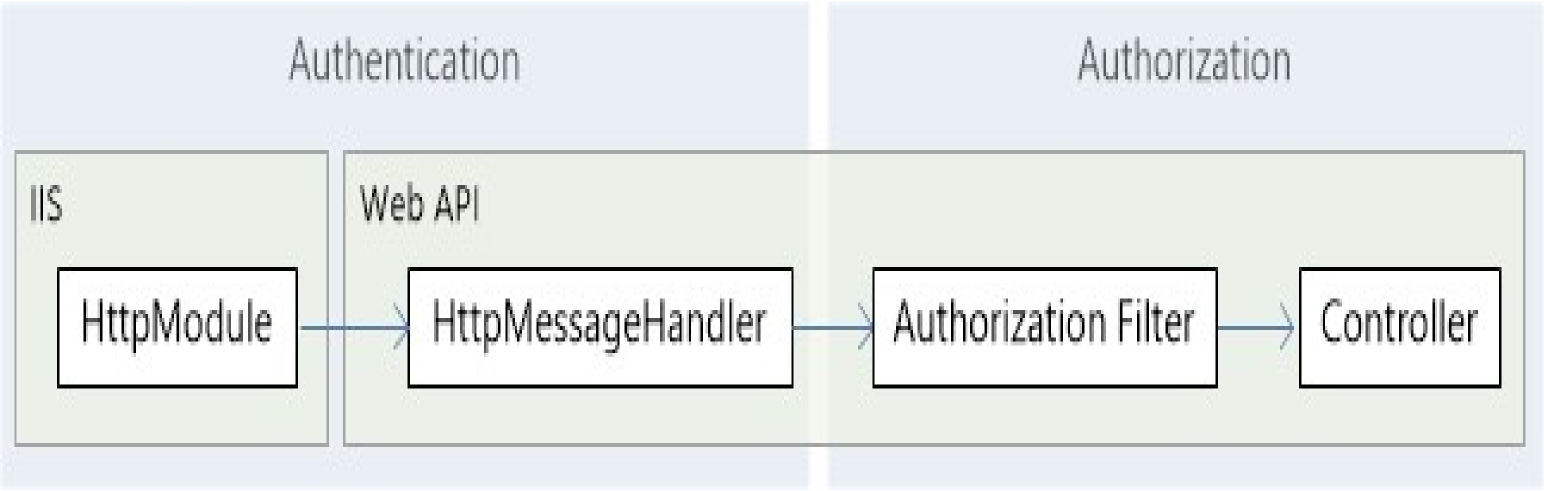
- HTTP modules run earlier in the pipeline. If you handle authentication in a message handler, the principal does not get set until the handler runs. Moreover, the principal reverts back to the previous principal when the response leaves the message handler.

Authorization

Authorization happens later in the pipeline, closer to the controller. That lets you make more granular choices when you grant access to resources.

- Authorization filters run before the controller action. If the request is not authorized, the filter returns an error response, and the action is not invoked.
- Within a controller action, you can get the current principal from the **ApiController.User** property. For example, you might filter a list of resources based on the user name, returning only those resources that belong to that user.

Authentication and Authorization in ASP.NET Web API



Chapter 08

1 Authentication and Authorization in ASP.NET Web API

2 Create a JWT Token

Create a JWT Token

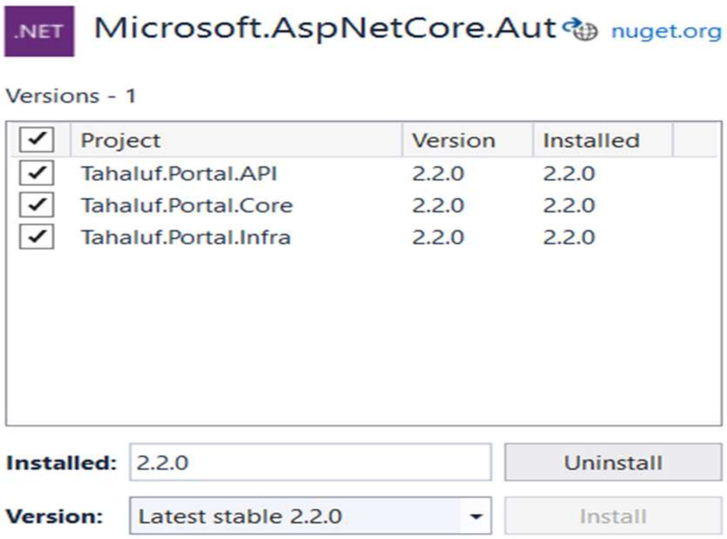
Create User Class:

```
public class User
{
public int UserId { get; set; }
public string UserName { get; set; }
public string Password { get; set; }
}
```

Create a JWT Token

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

- Microsoft.AspNetCore.Authentication



Microsoft.AspNetCore.Aut

Versions - 1

☑	Project	Version	Installed
☑	Tahaluf.Portal.API	2.2.0	2.2.0
☑	Tahaluf.Portal.Core	2.2.0	2.2.0
☑	Tahaluf.Portal.Infra	2.2.0	2.2.0

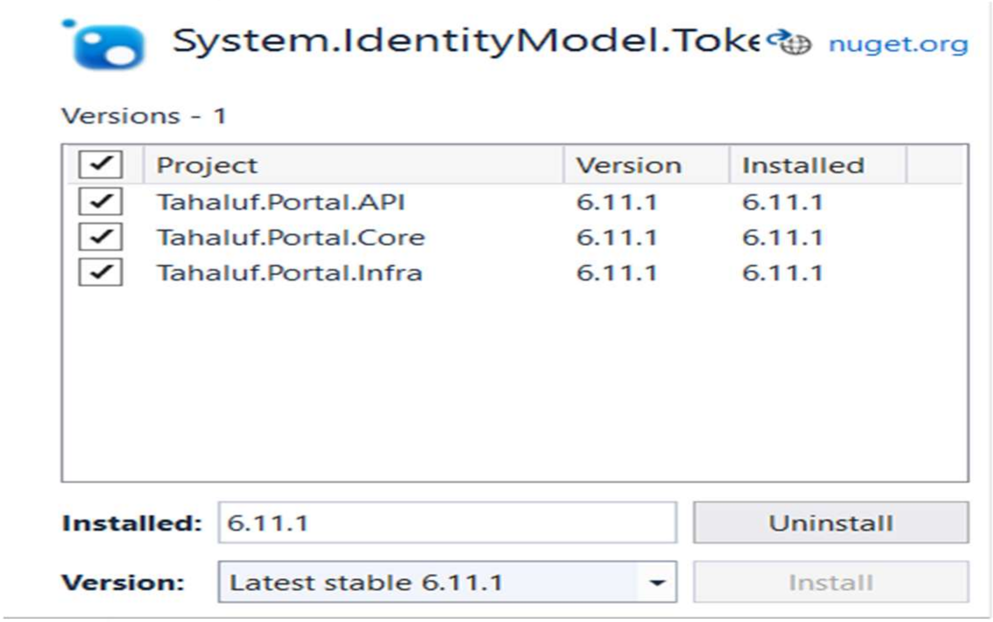
Installed: 2.2.0

Version: Latest stable 2.2.0

Create a JWT Token

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

- System.IdentityModel.Tokens.Jwt



Versions - 1

☑	Project	Version	Installed
☑	Tahaluf.Portal.API	6.11.1	6.11.1
☑	Tahaluf.Portal.Core	6.11.1	6.11.1
☑	Tahaluf.Portal.Infra	6.11.1	6.11.1

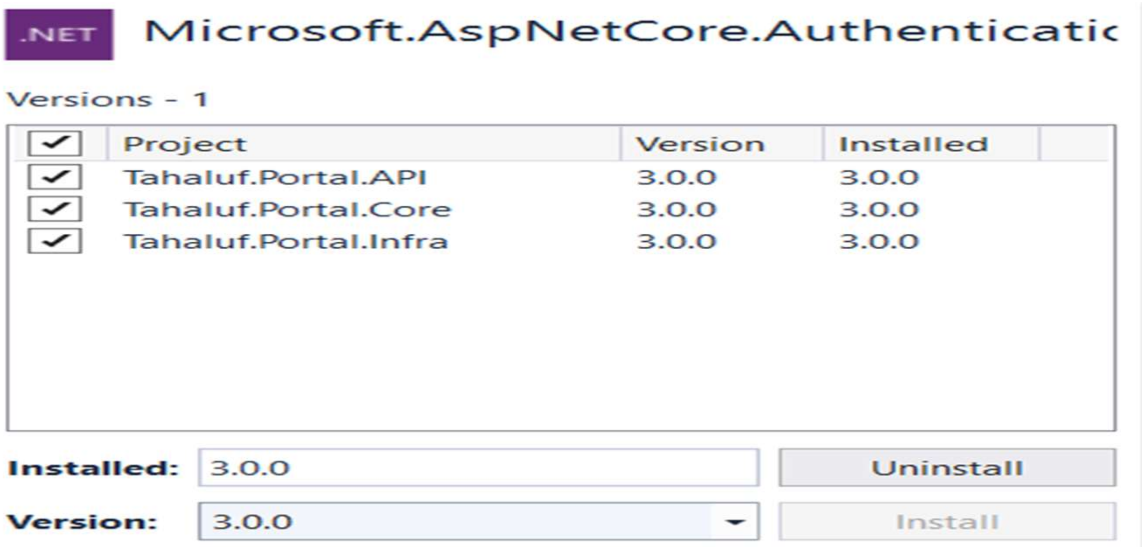
Installed: 6.11.1

Version: Latest stable 6.11.1

Create a JWT Token

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

- Microsoft.AspNetCore.Authentication.JwtBearer



.NET Microsoft.AspNetCore.Authenticatic

Versions - 1

☑	Project	Version	Installed
☑	Tahaluf.Portal.API	3.0.0	3.0.0
☑	Tahaluf.Portal.Core	3.0.0	3.0.0
☑	Tahaluf.Portal.Infra	3.0.0	3.0.0

Installed: 3.0.0

Version: 3.0.0

Create a JWT Token

In a Services in Tahaluf.Portal.Infra => Add JwtUserAuthService => Insert the following Code:

```
public class JwtUserAuthService : IJwtUserAuthService
{
public string Authenticate(User user)
{
var tokenHandler = new JwtSecurityTokenHandler();
var tokenKey = Encoding.ASCII.GetBytes("[SECRET USED TO
SIGN AND VERIFY JWT TOKENS, IT CAN BE ANY STRING]");
```

Create a JWT Token

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

Create a JWT Token

In a Services in Tahaluf.Portal.Core => Add IJwtUserAuthService => Insert the following Code:

```
public interface IJwtUserAuthService
{
string Authenticate(User user);
}
```

Create a JWT Token

In Startup:

```
services.AddScoped<IJwtUserAuthService, JwtUserAuthService>();
app.UseAuthentication();
services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
x.DefaultChallengeScheme =
JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
```

Create a JWT Token

```
y.RequireHttpsMetadata = false;
y.SaveToken = true;
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
};
});
```

Create a JWT Token

Add UserAuthController => Insert the following Code:

```
public class UserAuthController : Controller
{
private readonly IJwtUserAuthService JwtUserAuthService;
public UserAuthController(IJwtUserAuthService
jwtUserAuthService)
{
JwtUserAuthService = jwtUserAuthService;
}
```

Create a JWT Token

```
[HttpPost]
[Route("Authenticate")]
[ProducesResponseType(typeof(User), StatusCodes.Status200OK)]
[ProducesResponseType(typeof(User),
StatusCodes.Status400BadRequest)]
public IActionResult Authenticate([FromBody] User user)
{
var token = JwtUserAuthService.Authenticate(user);
if (token == null)
return Unauthorized();
else
return Ok(token);
}
```

References

[1]. https://www.educative.io/blog/crud-operations

[2]. https://www.codecademy.com/articles/what-is-crud

[3]. https://www.restapitutorial.com/lessons/httpmethods.html

[4]. https://www.w3schools.com/sql/sql_stored_procedures.asp

[5]. https://www.pragimtech.com/blog/blazor/rest-api-repository-pattern/#:~:text=sql%20server%20database-,What%20is%20Repository%20Pattern,is%20in%20the%20respective%20repository

[6]. https://code-maze.com/filtering-aspnet-core-webapi/

[7]. https://www.infoworld.com/article/3562271/how-to-use-data-transfer-objects-in-aspnet-core-31.html