ASP Net RESTFul Web API Core Security JWT

Topics

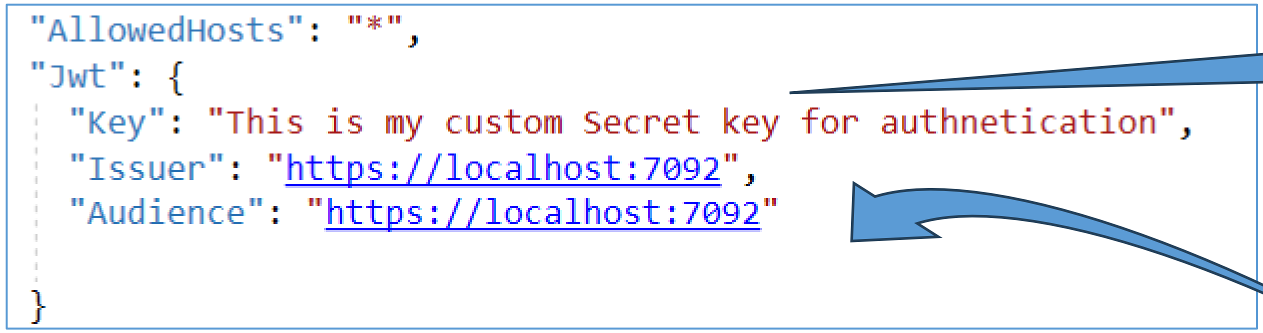
- **01 Installation**
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

Installation

- יש צורך להתקין את 3 החבילות הבאות. נא לשים לב לגרסאות שמתאימות לגרסת ה .NET שלכם:
- Microsoft.AspNetCore.Authentication.JwtBearer
- Microsoft.IdentityModel.Tokens
- System.IdentityModel.Tokens.Jwt

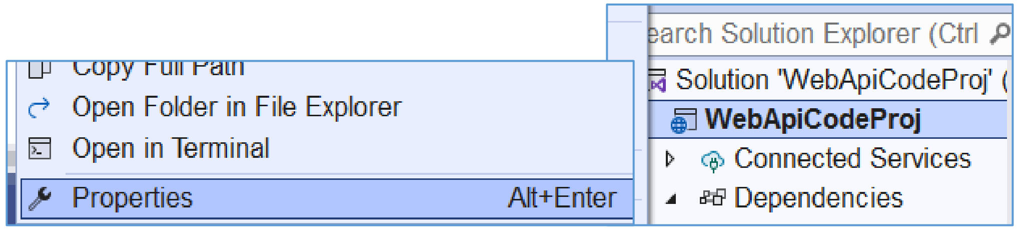
Set Up

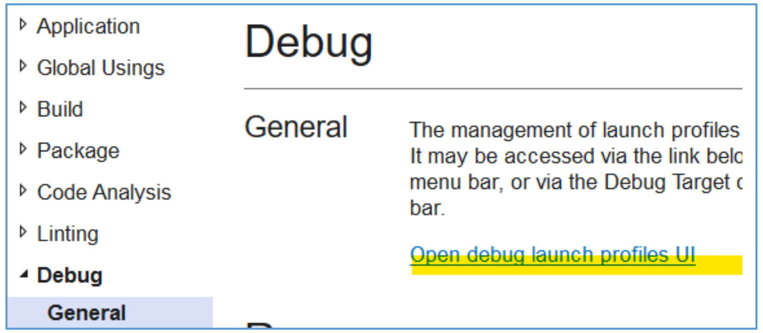
- בקובץ appsettings.json



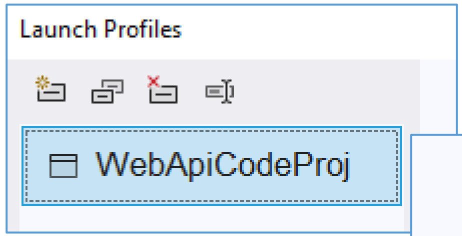
```
"AllowedHosts": "*",
"Jwt": {
  "Key": "This is my custom Secret key for authnetication",
  "Issuer": "https://localhost:7092",
  "Audience": "https://localhost:7092"

}
```

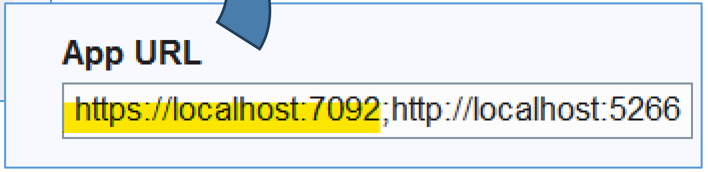
Just a random key. For example from https://randomkeygen.com/





The management of launch profiles
It may be accessed via the link belc
menu bar, or via the Debug Target c
bar.
Open debug launch profiles UI



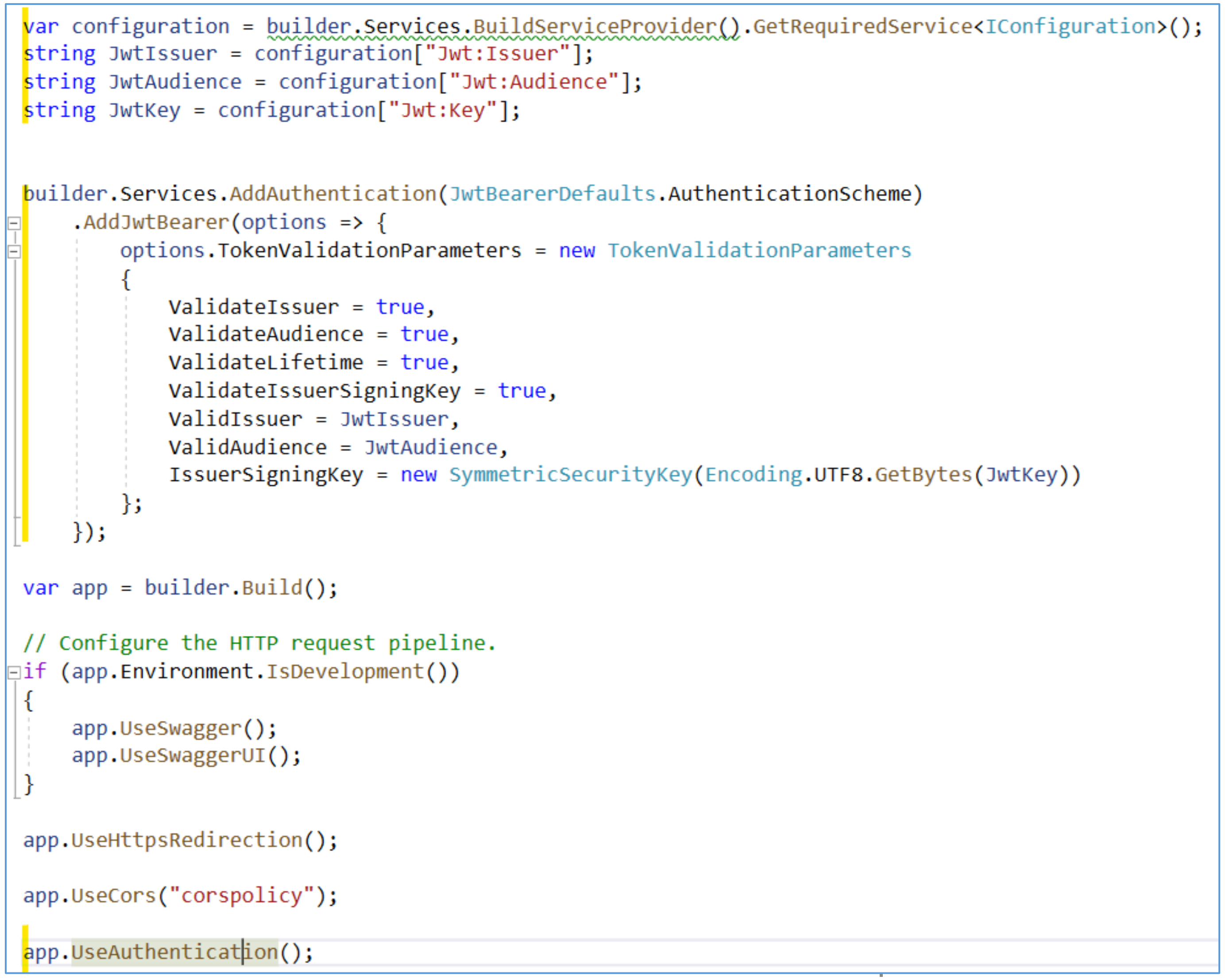


App URL

https://localhost:7092;http://localhost:5266

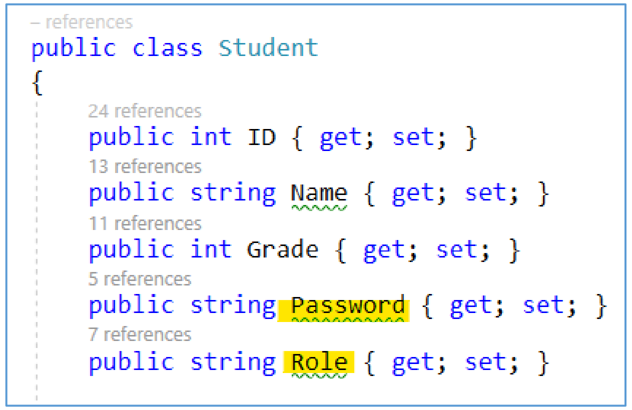
Set Up cont’

- בקובץ Program.cs

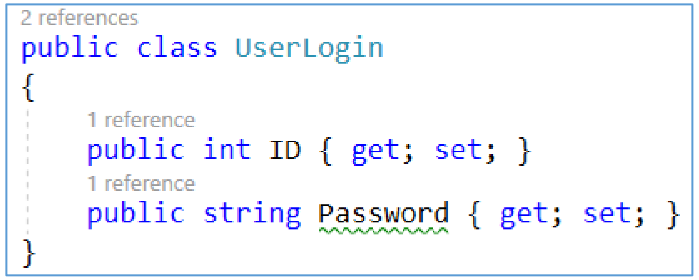


```
var configuration = builder.Services.BuildServiceProvider().GetRequiredService<IConfiguration>();
string JwtIssuer = configuration["Jwt:Issuer"];
string JwtAudience = configuration["Jwt:Audience"];
string JwtKey = configuration["Jwt:Key"];


builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options => {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = JwtIssuer,
            ValidAudience = JwtAudience,
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(JwtKey))
        };
    });

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseCors("corspolicy");

app.UseAuthentication();
```

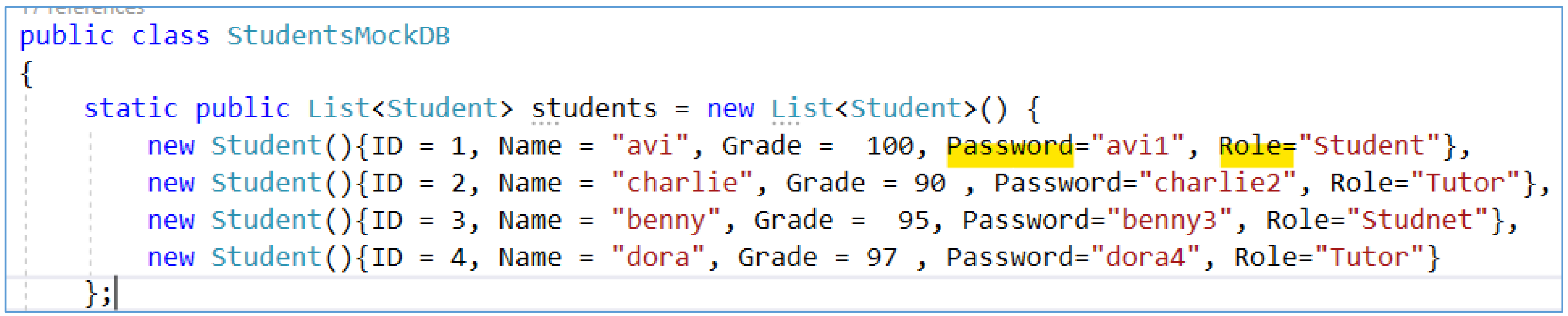
Model



```
– references
public class Student
{
    24 references
    public int ID { get; set; }
    13 references
    public string Name { get; set; }
    11 references
    public int Grade { get; set; }
    5 references
    public string Password { get; set; }
    7 references
    public string Role { get; set; }
```



```
2 references
public class UserLogin
{
    1 reference
    public int ID { get; set; }
    1 reference
    public string Password { get; set; }
}
```



```
public class StudentsMockDB
{
    static public List<Student> students = new List<Student>() {
        new Student(){ID = 1, Name = "avi", Grade =  100, Password="avi1", Role="Student"},
        new Student(){ID = 2, Name = "charlie", Grade = 90 , Password="charlie2", Role="Tutor"},
        new Student(){ID = 3, Name = "benny", Grade =  95, Password="benny3", Role="Studnet"},
        new Student(){ID = 4, Name = "dora", Grade = 97 , Password="dora4", Role="Tutor"}
    };
```

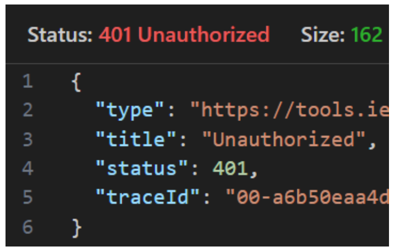
Login Controller – creating the JWT

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

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

    [AllowAnonymous]
    [HttpPost]
    0 references
    public IActionResult Login([FromBody] UserLogin user)
    {
        IActionResult response = Unauthorized();
        Student student = Authenticate(user);
        if (student != null)
        {
            var tokenStr = GenerateJSONWebToken(student);
            response = Ok(new { token = tokenStr });
        }
        return response;
    }
```

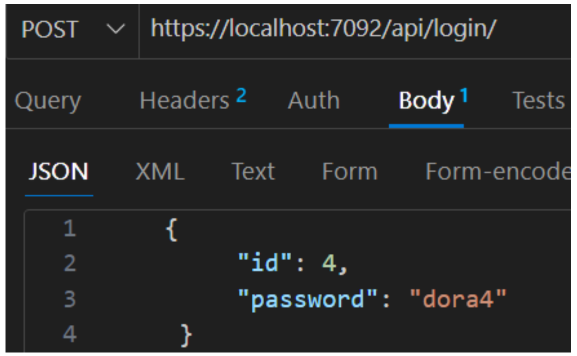
Dependency injection

If login failed

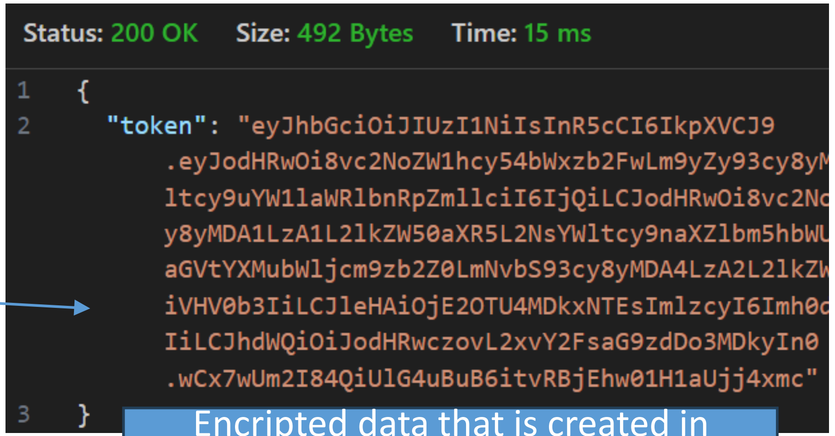
Next page


Status: 401 Unauthorized Size: 162

```
{
  "type": "https://tools.ie
  "title": "Unauthorized",
  "status": 401,
  "traceId": "00-a6b50eaa4d
}
```



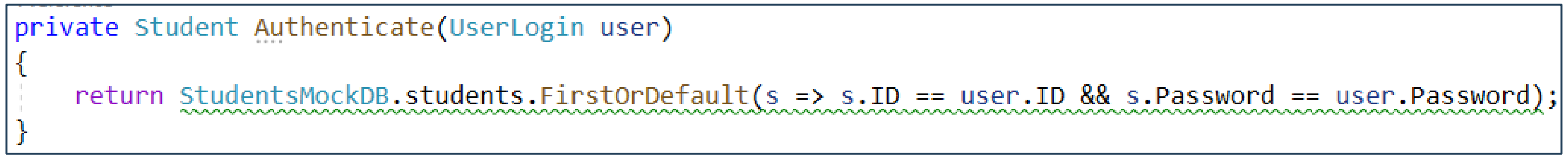
```
{
    "id": 4,
    "password": "dora4"
}
```


Status: 200 OK Size: 492 Bytes Time: 15 ms

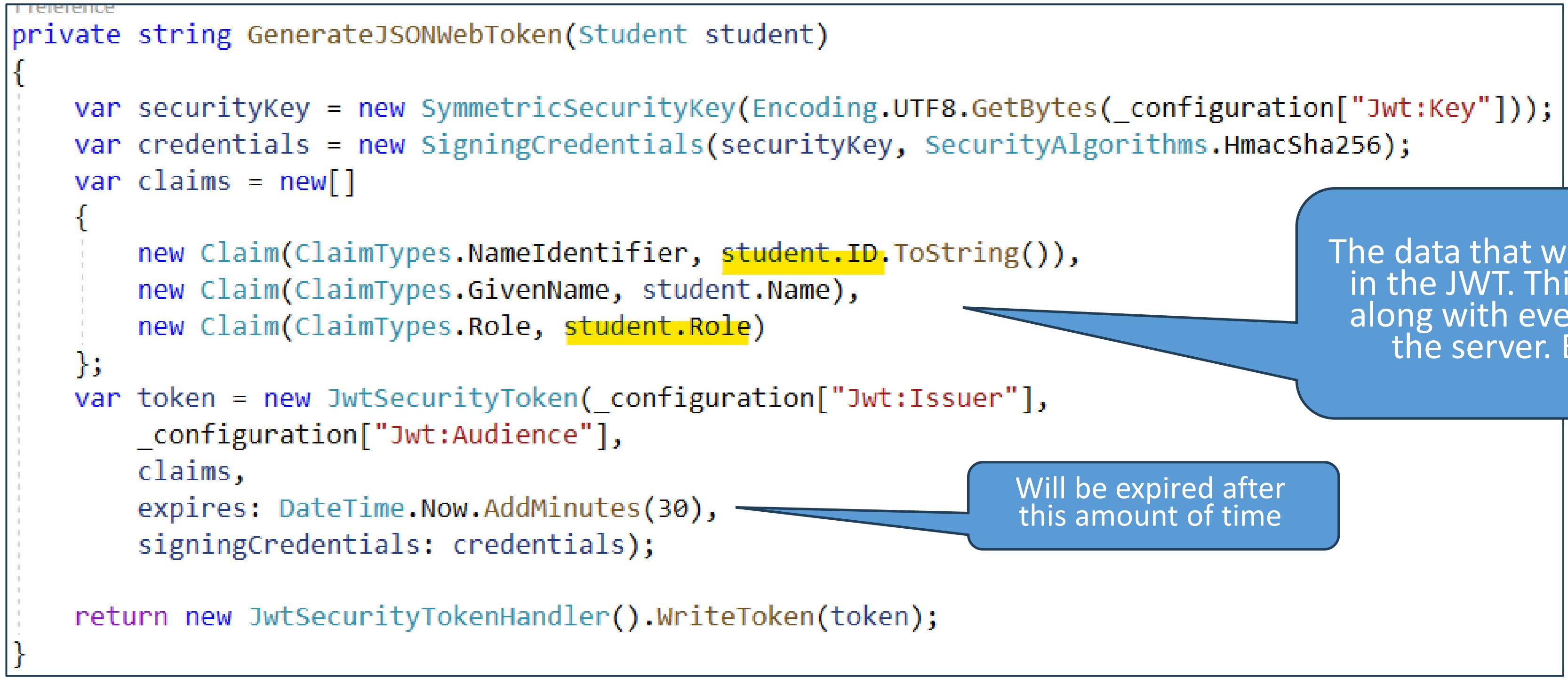
```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yM
      ltcy9uYW1laWRlbnRpZmllciI6IjQiLCJodHRwOi8vc2Nc
      y8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9naXZlbm5hbWU
      aGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW
      iVHV0b3IiLCJleHAiOjE2OTU4MDkxNTEsImlzcyI6Imh0d
      IiLCJhdWQiOiJodHRwczovL2xvY2FsaG9zdDo3MDkyIn0
      .wCx7wUm2I84QiUlG4uBuB6itvRBjEhw01H1aUjj4xmc"
}
```

Encripted data that is created in GenerateJSONWebToken function

Login Controller cont’



```
private Student Authenticate(UserLogin user)
{
    return StudentsMockDB.students.FirstOrDefault(s => s.ID == user.ID && s.Password == user.Password);
}
```

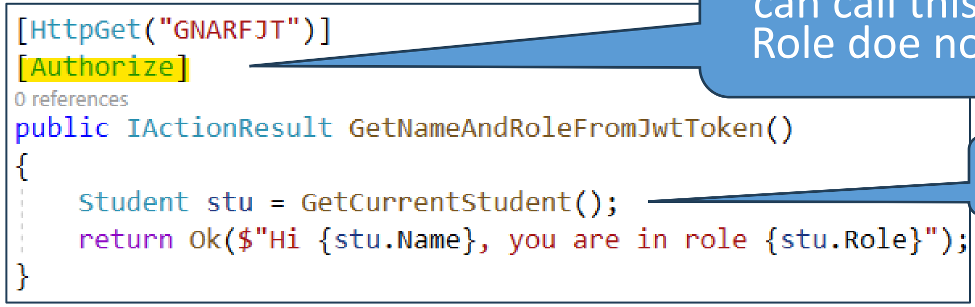


```
1 reference
private string GenerateJSONWebToken(Student student)
{
    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
    var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);
    var claims = new[]
    {
        new Claim(ClaimTypes.NameIdentifier, student.ID.ToString()),
        new Claim(ClaimTypes.GivenName, student.Name),
        new Claim(ClaimTypes.Role, student.Role)
    };
    var token = new JwtSecurityToken(_configuration["Jwt:Issuer"],
        _configuration["Jwt:Audience"],
        claims,
        expires: DateTime.Now.AddMinutes(30),
        signingCredentials: credentials);

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

The data that we need to hold in the JWT. This will be sent along with every request to the server. Encrypted.

Will be expired after this amount of time

Students Controller – using the JWT

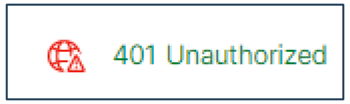


```
[HttpGet("GNARFJT")]
[Authorize]
0 references
public IActionResult GetNameAndRoleFromJwtToken()
{
    Student stu = GetCurrentStudent();
    return Ok($"Hi {stu.Name}, you are in role {stu.Role}");
}
```

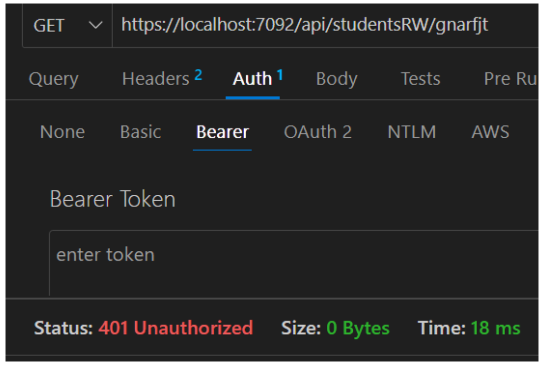
Only logged in students can call this method. Role doe not matter!

Next page...

- If not logged in

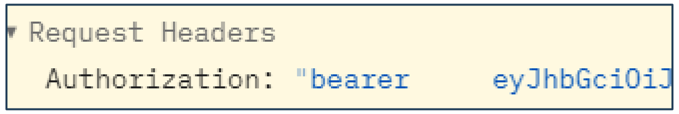


401 Unauthorized

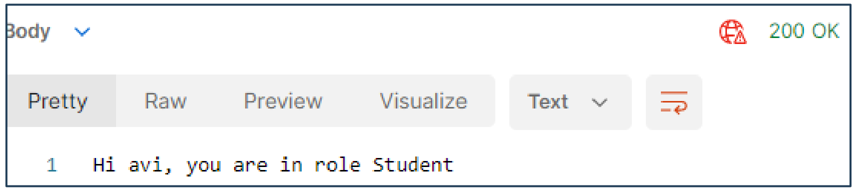


```
GET  https://localhost:7092/api/studentsRW/gnarfjt
Auth: Bearer
Bearer Token: enter token
Status: 401 Unauthorized   Size: 0 Bytes   Time: 18 ms
```

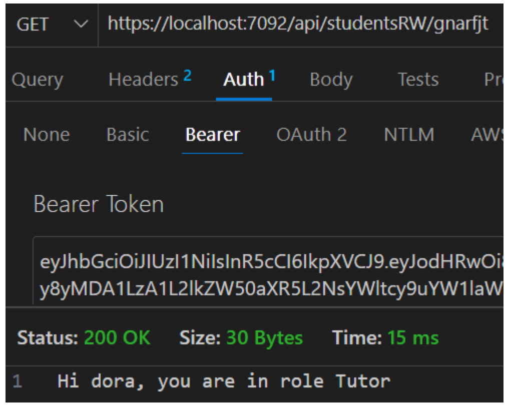
- Else send the token along with the request like this:



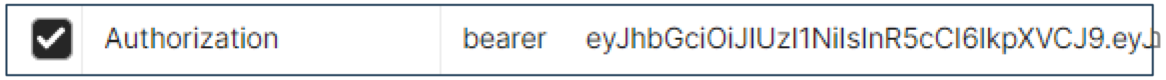
```
Request Headers
  Authorization: "bearer     eyJhbGciOiJ
```



```
200 OK
Hi avi, you are in role Student
```

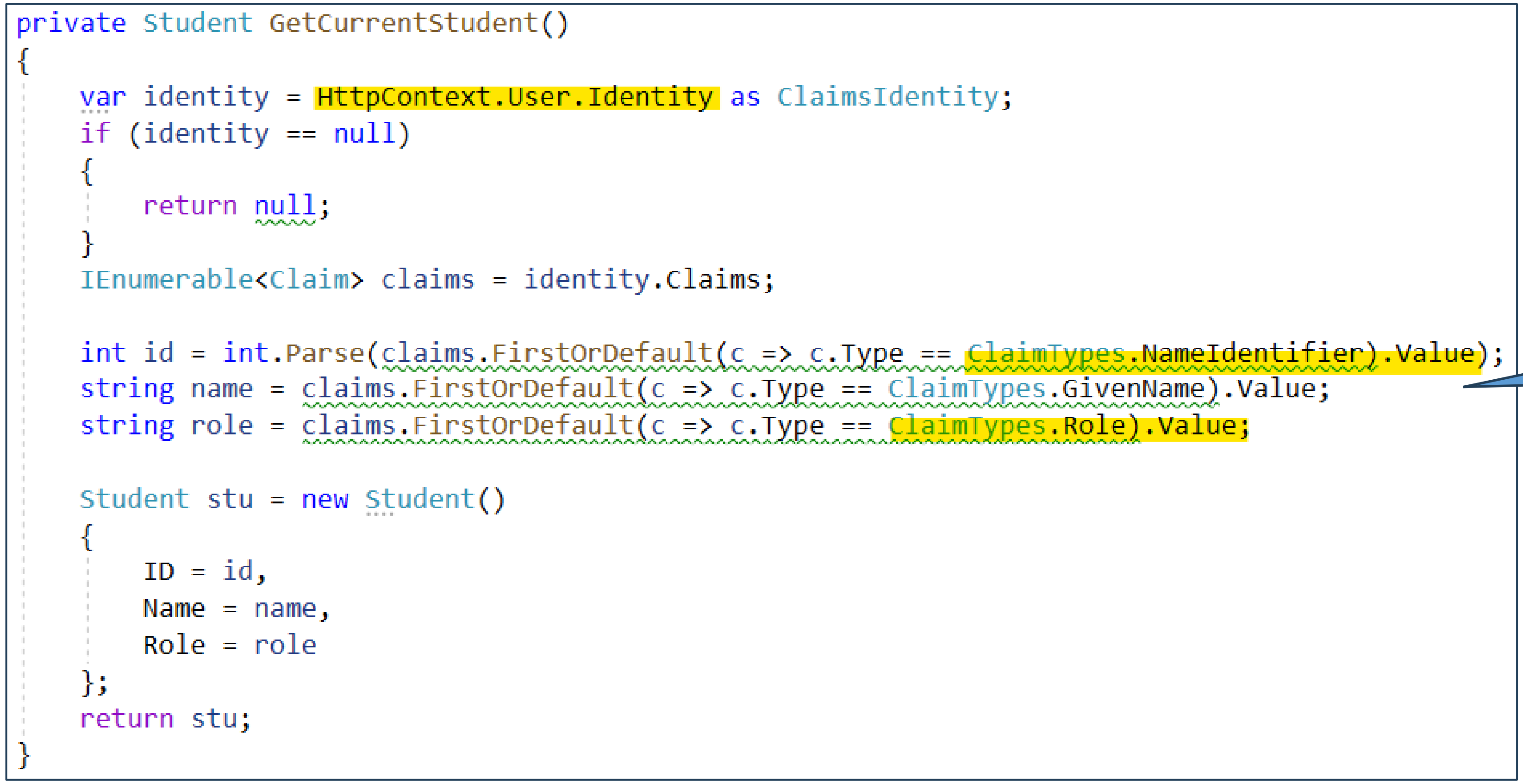


```
GET  https://localhost:7092/api/studentsRW/gnarfjt
Auth: Bearer
Bearer Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi
y8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laW
Status: 200 OK   Size: 30 Bytes   Time: 15 ms
Hi dora, you are in role Tutor
```



✓	Authorization	bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ

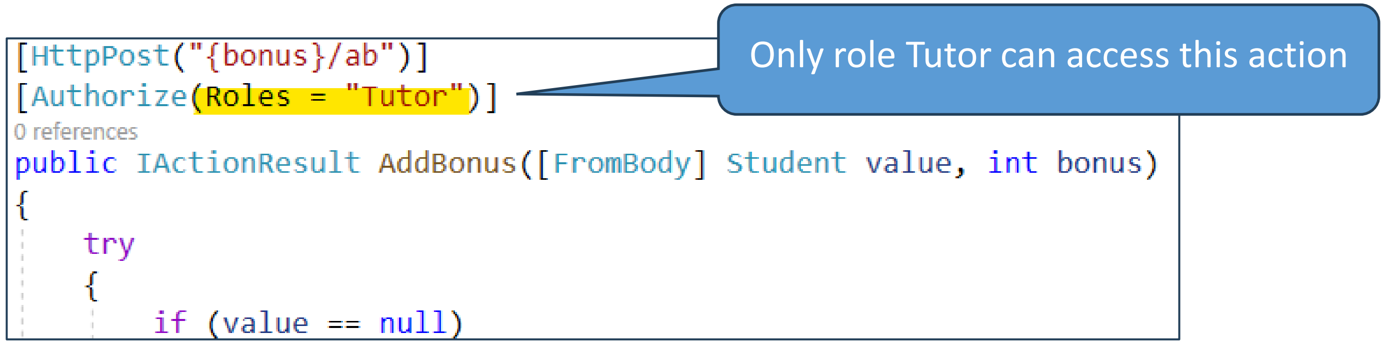
Students Controller – using the JWT



```
private Student GetCurrentStudent()
{
    var identity = HttpContext.User.Identity as ClaimsIdentity;
    if (identity == null)
    {
        return null;
    }
    IEnumerable<Claim> claims = identity.Claims;

    int id = int.Parse(claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier).Value);
    string name = claims.FirstOrDefault(c => c.Type == ClaimTypes.GivenName).Value;
    string role = claims.FirstOrDefault(c => c.Type == ClaimTypes.Role).Value;

    Student stu = new Student()
    {
        ID = id,
        Name = name,
        Role = role
    };
    return stu;
}
```

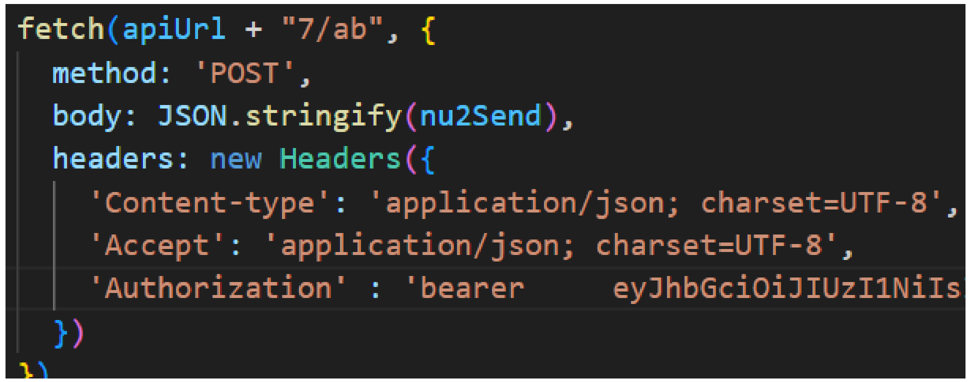
Get the data from JWT

Students Controller – using the JWT



```
[HttpPost("{bonus}/ab")]
[Authorize(Roles = "Tutor")]
0 references
public IActionResult AddBonus([FromBody] Student value, int bonus)
{
    try
    {
        if (value == null)
```

Only role Tutor can access this action



```
fetch(apiUrl + "7/ab", {
  method: 'POST',
  body: JSON.stringify(nu2Send),
  headers: new Headers({
    'Content-type': 'application/json; charset=UTF-8',
    'Accept': 'application/json; charset=UTF-8',
    'Authorization' : 'bearer     eyJhbGciOiJIUzI1NiIs
  })
})
```

Student role will get 403 Forbidden

Tutor role will get 204 No Content

Without authorization header will get 401 Unauthorized