**WEEK – 4 Assignments**

**ASP. NET Core 8. 0 Web API**

**Mandatory hands-on :-**

**5. WebApi\_Handson :**

**Objectives :**

· Explain CORS enablement for Web API access for local application

o What is CORS?, How to enable CORS thru Startup.cs, Install Cors nuget package to Web API application

· Demonstrate security in WebAPI

o Bearer and Jwt token authentication, Use Authorize attribute & send roles in Jwt token, Setting in Startup.cs for AddAuthentication and AddJwtBearer with validation attributes, UseAuthentication, AllowAnonymous to AuthController to generate token, Claims.

**1. JsonWebToken :**

**2. Use the JWT generated thru the AuthController to be used in POSTMAN request :**

[**Program.cs**](http://program.cs) **:**

| **using Microsoft.AspNetCore.Authentication.JwtBearer; using Microsoft.IdentityModel.Tokens; using System.Text;  var builder = WebApplication.CreateBuilder(args);  // Add services to the container. builder.Services.AddControllers();  // CORS policy builder.Services.AddCors(options => {  options.AddPolicy("AllowAll",  policy => policy.AllowAnyOrigin()  .AllowAnyHeader()  .AllowAnyMethod()); });  // JWT Authentication string securityKey = "mysuperdupersecretkeythatismorethan32chars!"; var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));  builder.Services.AddAuthentication(options => {  options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  options.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme; }) .AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, options => {  options.TokenValidationParameters = new TokenValidationParameters  {  ValidateIssuer = true,  ValidateAudience = true,  ValidateLifetime = true,  ValidateIssuerSigningKey = true,  ValidIssuer = "mySystem",  ValidAudience = "myUsers",  IssuerSigningKey = symmetricSecurityKey  }; });  var app = builder.Build();  app.UseCors("AllowAll"); app.UseAuthentication(); app.UseAuthorization();  app.MapControllers();  app.Run();** |
| --- |

[**EmployeeController.cs**](http://employeecontroller.cs) **:**

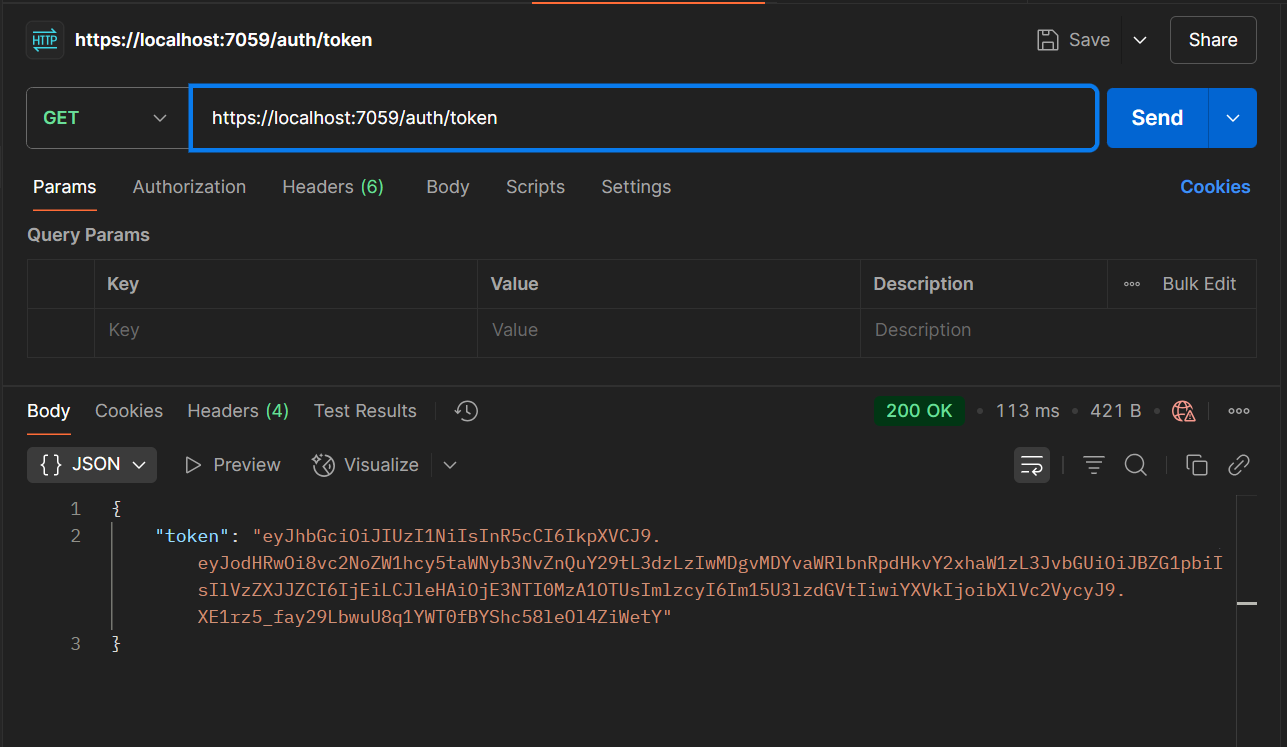
| **using Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc;  namespace JwtCorsDemo.Controllers {  [ApiController]  [Route("[controller]")]  [Authorize(Roles = "Admin,POC")] // Only users with Admin or POC roles can access  public class EmployeeController : ControllerBase  {  [HttpGet]  public IActionResult Get()  {  return Ok("You are authorized!");  }  } }** |
| --- |

[**AuthController.cs**](http://authcontroller.cs) **:**

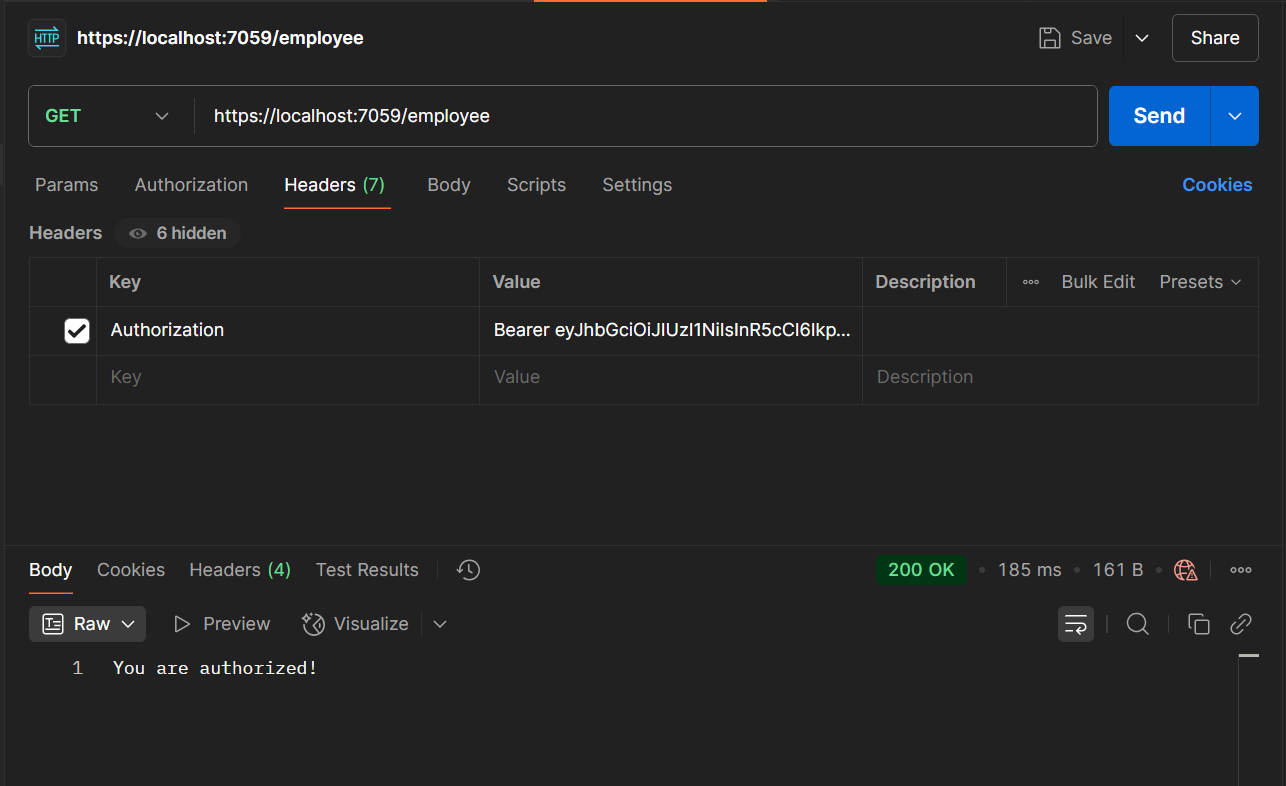
| using Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc; using Microsoft.IdentityModel.Tokens; using System.IdentityModel.Tokens.Jwt; using System.Security.Claims; using System.Text;  namespace JwtCorsDemo.Controllers {  [ApiController]  [Route("[controller]")]  [AllowAnonymous]  public class AuthController : ControllerBase  {  [HttpGet("token")]  public IActionResult GetToken()  {  var token = GenerateJSONWebToken(1, "Admin");  return Ok(new { token });  }   private string GenerateJSONWebToken(int userId, string userRole)  {  var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeythatismorethan32chars!"));  var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);   var claims = new List<Claim>  {  new Claim(ClaimTypes.Role, userRole),  new Claim("UserId", userId.ToString())  };   var token = new JwtSecurityToken(  issuer: "mySystem",  audience: "myUsers",  claims: claims,  expires: DateTime.Now.AddMinutes(10), // Change to 2 for testing expiration  signingCredentials: credentials);   return new JwtSecurityTokenHandler().WriteToken(token);  }  } } |
| --- |

**OUTPUT (POSTMAN) :**

**GET : auth/token :**

****

**GET : employees :**

****

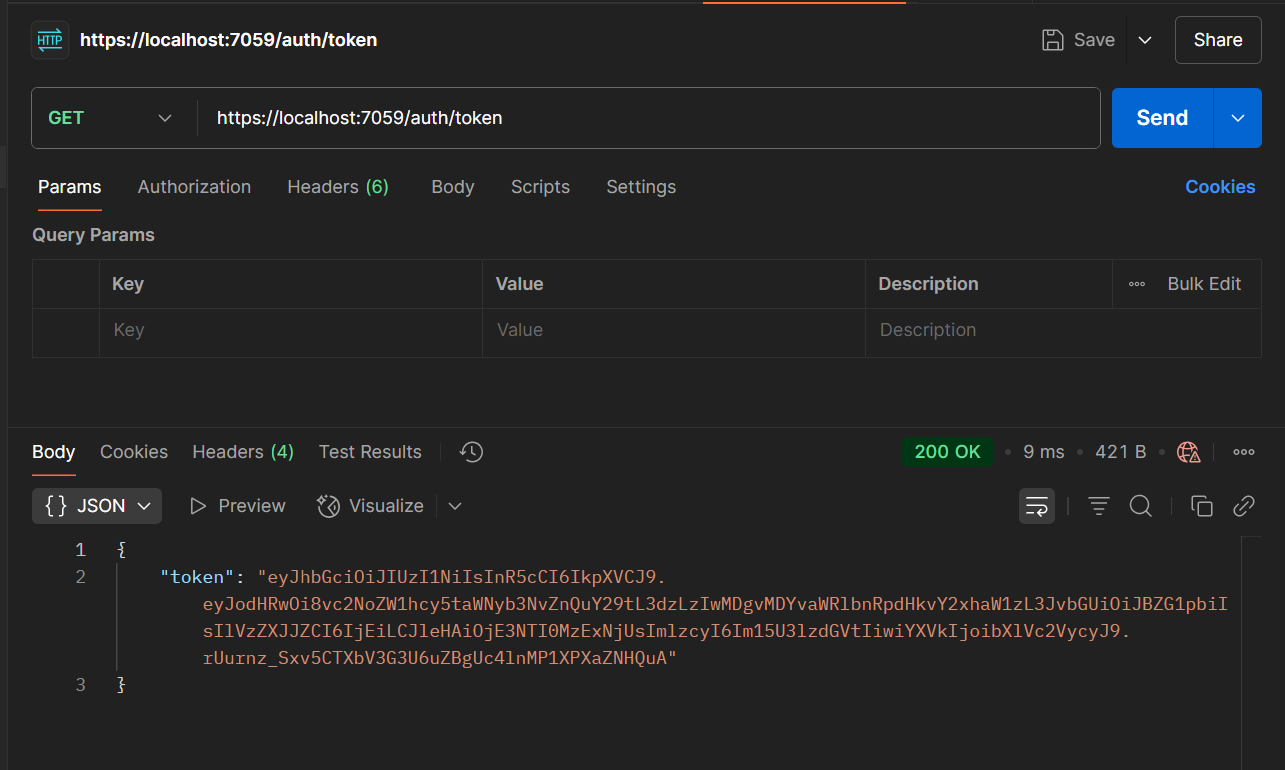
**3. Check for JWT expiration :**

[**AuthController.cs**](http://authcontroller.cs) **(Updated):**

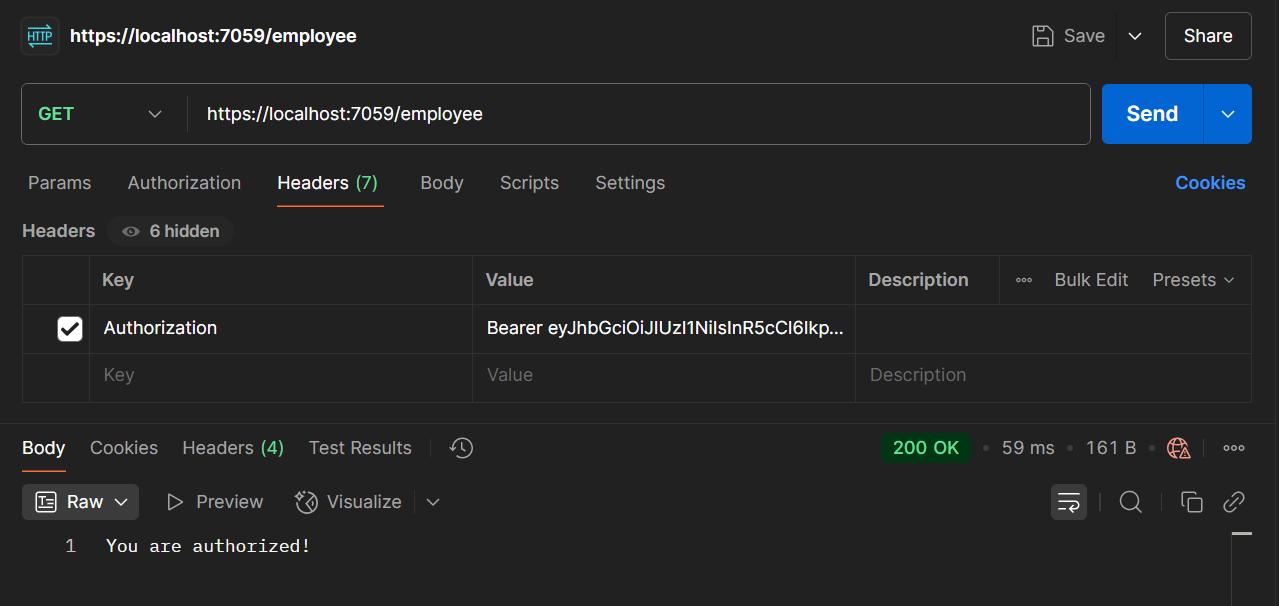
| using Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc; using Microsoft.IdentityModel.Tokens; using System.IdentityModel.Tokens.Jwt; using System.Security.Claims; using System.Text;  namespace JwtCorsDemo.Controllers {  [ApiController]  [Route("[controller]")]  [AllowAnonymous]  public class AuthController : ControllerBase  {  [HttpGet("token")]  public IActionResult GetToken()  {  var token = GenerateJSONWebToken(1, "Admin");  return Ok(new { token });  }   private string GenerateJSONWebToken(int userId, string userRole)  {  var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeythatismorethan32chars!"));  var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);   var claims = new List<Claim>  {  new Claim(ClaimTypes.Role, userRole),  new Claim("UserId", userId.ToString())  };   var token = new JwtSecurityToken(  issuer: "mySystem",  audience: "myUsers",  claims: claims,  expires: DateTime.Now.AddMinutes(2), // Change to 2 for testing expiration  signingCredentials: credentials);   return new JwtSecurityTokenHandler().WriteToken(token);  }  } } |
| --- |

**OUTPUT (POSTMAN) :**

**GET : auth/token**

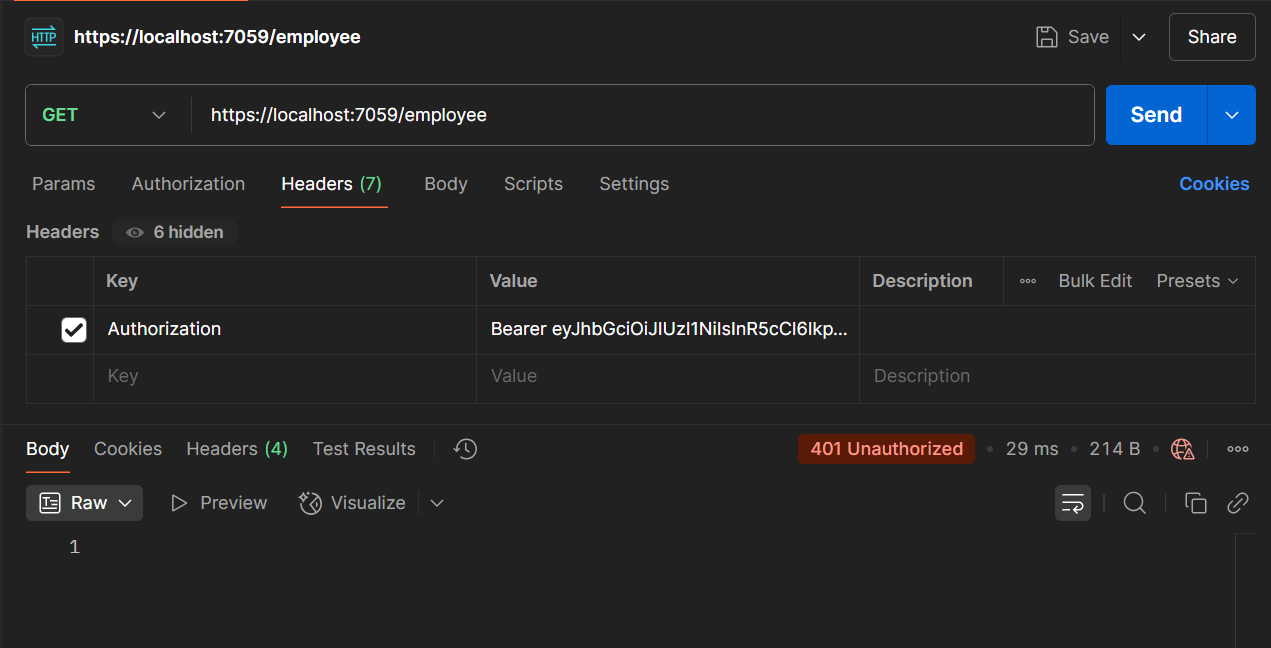
****

**GET : employee :**

****

**After 2 minutes :**

**GET : employee :**

****

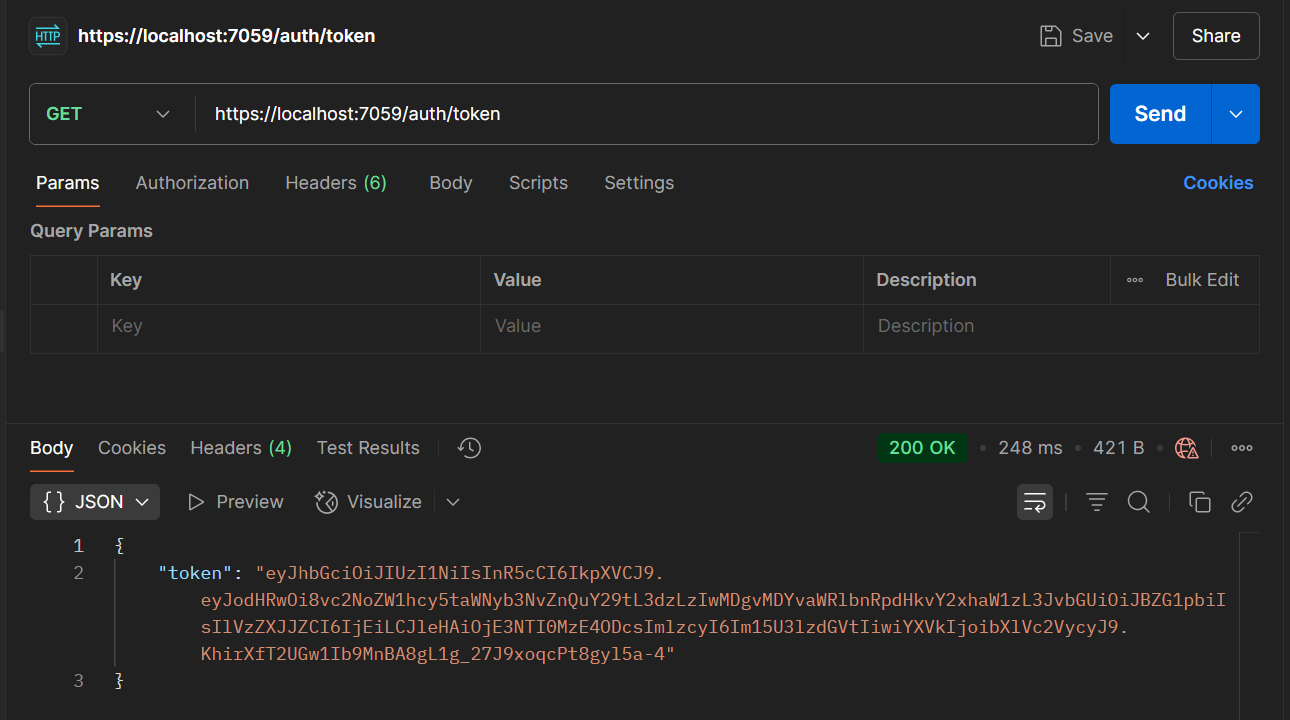
**4. Add the roles to be authorized in the Authorize attribute.**

[**EmployeeControIler.cs**](http://employeecontroiler.cs) **(updated):**

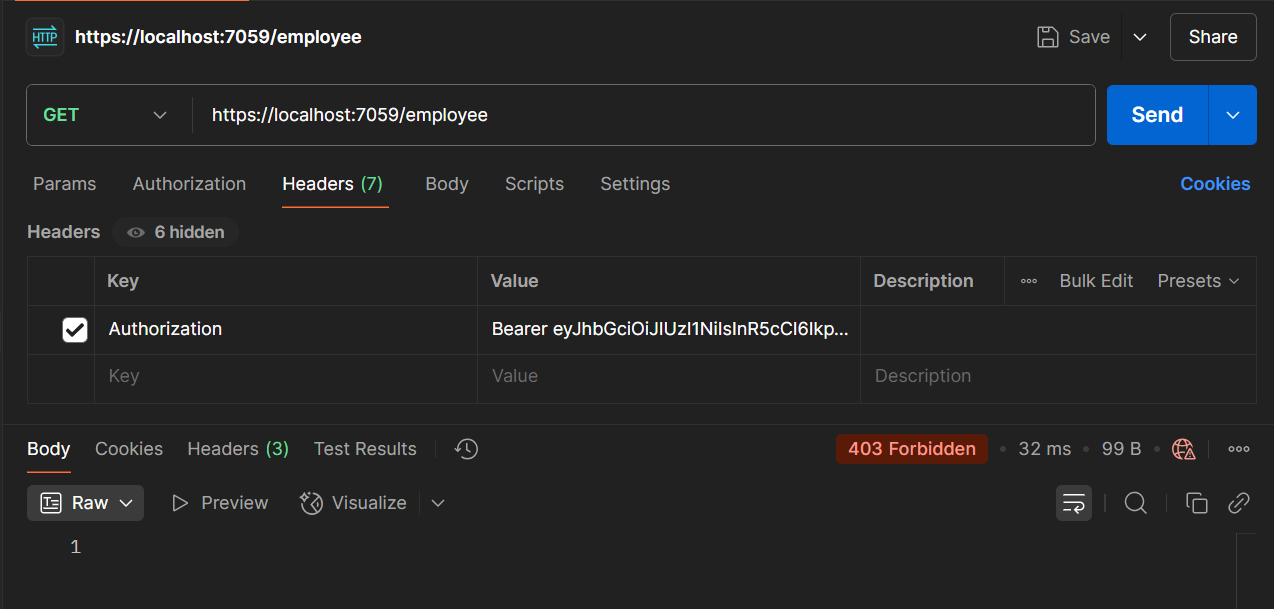
| using Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc;  namespace JwtCorsDemo.Controllers {  [ApiController]  [Route("[controller]")]  [Authorize(Roles = "POC")] // Only users with Admin or POC roles can access  public class EmployeeController : ControllerBase  {  [HttpGet]  public IActionResult Get()  {  return Ok("You are authorized!");  }  } } |
| --- |

**OUTPUT (POSTMAN) :**

**GET : auth/token :**

****

**GET : employee :**

****