1. Create new folder as model
2. In that create classes employeeV2.cs
   1. EmployeeV2.cs

| namespace WebApplication1.Models  {  public class EmployeeV2  {  public int Id { get; set; }  public string Name { get; set; } = string.Empty;  public int Salary { get; set; }  public bool Permanent { get; set; }  public Department Department { get; set; } = default!;  public List<Skill> Skills { get; set; } = new();  public DateTime DateOfBirth { get; set; }  }  public class Department  {  public int Id { get; set; }  public string Name { get; set; } = string.Empty;  }  public class Skill  {  public int Id { get; set; }  public string Name { get; set; } = string.Empty;  }  } |
| --- |

1. Create new controller
   1. EmployeeV2Controller.cs

| using Microsoft.AspNetCore.Mvc;  using WebApplication1.Filters;  using WebApplication1.Models;  using System.Linq;  namespace WebApplication1.Controllers  {  [ApiController]  [Route("api/emp2")]  [ServiceFilter(typeof(CustomAuthFilter))]  public class EmployeeV2Controller : ControllerBase  {  private readonly List<EmployeeV2> \_data = new()  {  new EmployeeV2 {  Id = 1,  Name = "Furnia",  Salary = 45000,  Permanent = true,  DateOfBirth = new DateTime(1996, 4, 15),  Department = new WebApplication1.Models.Department { Id = 1, Name = "IT" },  Skills = new List<Skill> {  new Skill { Id = 1, Name = "C#" },  new Skill { Id = 2, Name = "SQL" }  }  }  };  [HttpGet(Name = "GetAllEmployeesV2")]  [ProducesResponseType(typeof(IEnumerable<EmployeeV2>), 200)]  [ProducesResponseType(500)]  public ActionResult<IEnumerable<EmployeeV2>> GetStandard()  {  return Ok(\_data);  }  [HttpPost]  [ProducesResponseType(typeof(EmployeeV2), 201)]  public ActionResult<EmployeeV2> Post([FromBody] EmployeeV2 emp)  {  emp.Id = \_data.Max(e => e.Id) + 1;  \_data.Add(emp);  return CreatedAtAction(nameof(GetStandard), new { id = emp.Id }, emp);  }  }  } |
| --- |

1. Create Custom Authorization Filter
2. New folder as filter and add class
   1. customAuthFilter.cs

| using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Filters;  namespace WebApplication1.Filters  {  public class CustomAuthFilter : ActionFilterAttribute  {  public override void OnActionExecuting(ActionExecutingContext context)  {  var hasAuth = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);  if (!hasAuth)  {  context.Result = new BadRequestObjectResult("Invalid request - No Auth token");  return;  }  if (!token.ToString().Contains("Bearer"))  {  context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");  }  }  }  } |
| --- |

1. Create Custom Exception Filter
2. Add new class in filter folder
   1. CustomExceptionFilter.cs

| using Microsoft.AspNetCore.Mvc;  using Microsoft.AspNetCore.Mvc.Filters;  namespace WebApplication1.Filters  {  public class CustomExceptionFilter : IExceptionFilter  {  public void OnException(ExceptionContext context)  {  var exception = context.Exception;  var logPath = "logs/errors.txt";  Directory.CreateDirectory("logs");  File.AppendAllText(logPath, $"{DateTime.Now}: {exception.Message}\n");  context.Result = new ObjectResult("An internal error occurred.")  {  StatusCode = 500  };  }  }  } |
| --- |

