**WEEK – 4 Assignments**

**Mandatory hands-on :-**

**3. WebApi\_Handson :**

**Objectives:**

· Demonstrate creation of an Action method to return list of custom class entity

o Model class creation, Use AllowAnonymous attribute, Use HttpGet action method

· Explain the usage of FromBody attribute

o Read the model object from request, other than the query string parameter

· Demonstrate Custom filter

o Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request, Create filter for Custom exception - Need to install Microsoft.AspNetCore.Mvc.WebApiCompatShim package

**1. Web Api using custom model class :**

[**Employee.cs**](http://employee.cs) **:**

|  |
| --- |
| **public class Department {  public int Id { get; set; }  public string Name { get; set; } }  public class Skill {  public int Id { get; set; }  public string Name { get; set; } }  public class Employee {  public int Id { get; set; }  public string Name { get; set; }  public int Salary { get; set; }  public bool Permanent { get; set; }  public Department Department { get; set; }  public List<Skill> Skills { get; set; }  public DateTime DateOfBirth { get; set; } }** |

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

|  |
| --- |
| **var builder = WebApplication.CreateBuilder(args);  // Add services to the container. builder.Services.AddControllers(options => {  options.Filters.Add<CustomExceptionFilter>(); }); // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle builder.Services.AddEndpointsApiExplorer(); builder.Services.AddSwaggerGen();  var app = builder.Build();   // Configure the HTTP request pipeline. if (app.Environment.IsDevelopment()) {  app.UseSwagger();  app.UseSwaggerUI(); }  app.UseHttpsRedirection();  app.UseAuthorization();  app.MapControllers();  app.Run();** |

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

|  |
| --- |
| **using Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc; using System; using System.Collections.Generic;  [ApiController] [Route("api/[controller]")] [AllowAnonymous] [CustomAuthFilter] public class EmployeeController : ControllerBase {  private static List<Employee> \_employees = GetStandardEmployeeList();   private static List<Employee> GetStandardEmployeeList()  {  return new List<Employee>  {  new Employee  {  Id = 1,  Name = "Ankit Sahu",  Salary = 50000,  Permanent = true,  Department = new Department { Id = 1, Name = "HR" },  Skills = new List<Skill>  {  new Skill { Id = 1, Name = "C#" },  new Skill { Id = 2, Name = "SQL" }  },  DateOfBirth = new DateTime(1990, 1, 1)  },  new Employee  {  Id = 2,  Name = "Divyanshu Kumar",  Salary = 60000,  Permanent = false,  Department = new Department { Id = 2, Name = "IT" },  Skills = new List<Skill>  {  new Skill { Id = 3, Name = "JavaScript" }  },  DateOfBirth = new DateTime(1992, 2, 2)  }  };  }   [HttpGet]  [ProducesResponseType(StatusCodes.Status200OK)]  public ActionResult<List<Employee>> GetStandard()  {  return Ok(\_employees);  }   [HttpPost]  public IActionResult CreateEmployee([FromBody] Employee employee)  {  \_employees.Add(employee);  return CreatedAtAction(nameof(GetStandard), new { id = employee.Id }, employee);  }   [HttpPut("{id}")]  public IActionResult UpdateEmployee(int id, [FromBody] Employee employee)  {  var existing = \_employees.Find(e => e.Id == id);  if (existing == null) return NotFound();  existing.Name = employee.Name;  existing.Salary = employee.Salary;  existing.Permanent = employee.Permanent;  existing.Department = employee.Department;  existing.Skills = employee.Skills;  existing.DateOfBirth = employee.DateOfBirth;  return NoContent();  }   [HttpGet("throw")]  [ProducesResponseType(StatusCodes.Status500InternalServerError)]  public ActionResult ThrowException()  {  throw new Exception("Demo exception");  } }** |

**2. Create a Custom action filter for Authorization :**

[**CustomAuthFilter.cs**](http://customauthfilter.cs) **:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.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");  return;  }  base.OnActionExecuting(context);  } } |

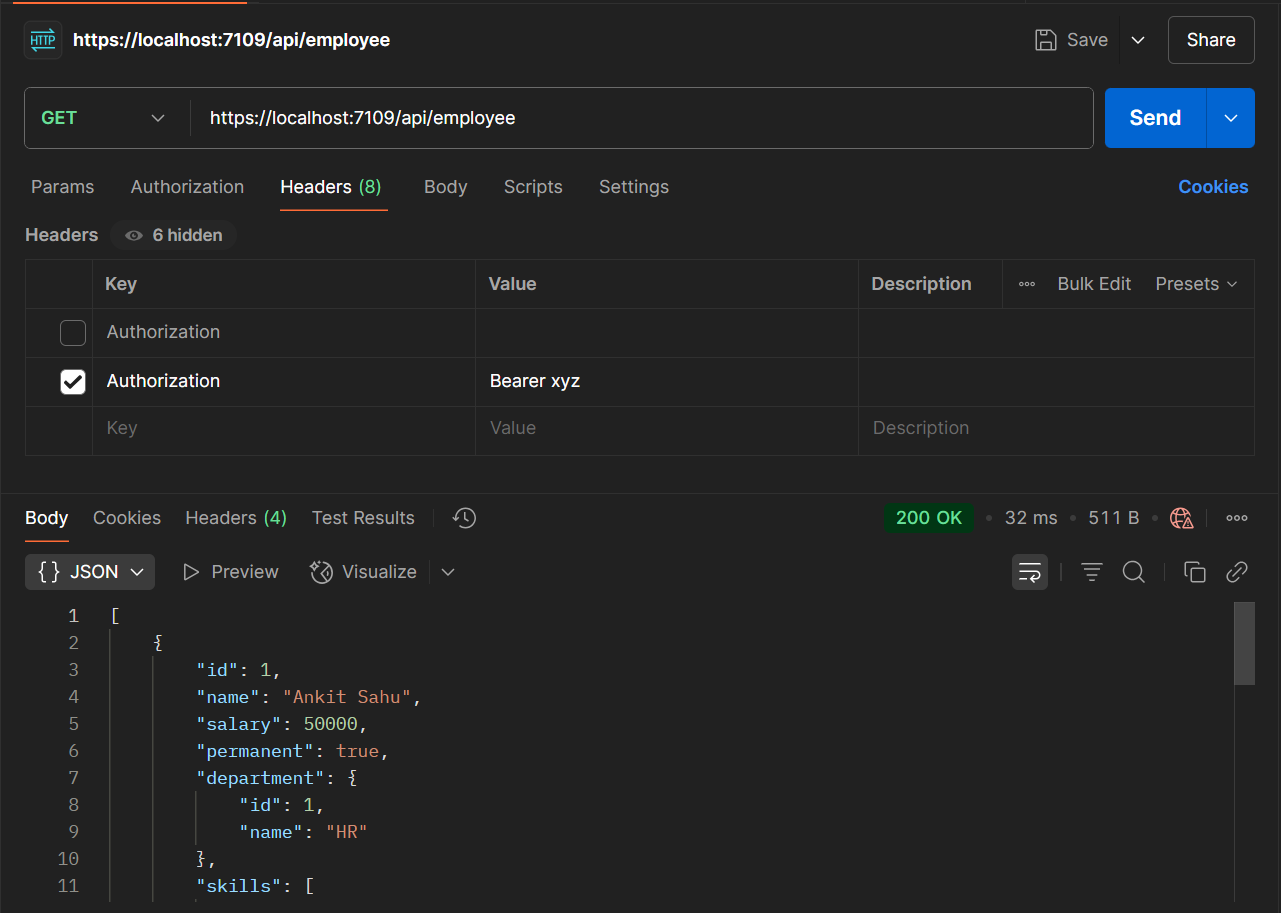
**3. Custom Exception filter :**

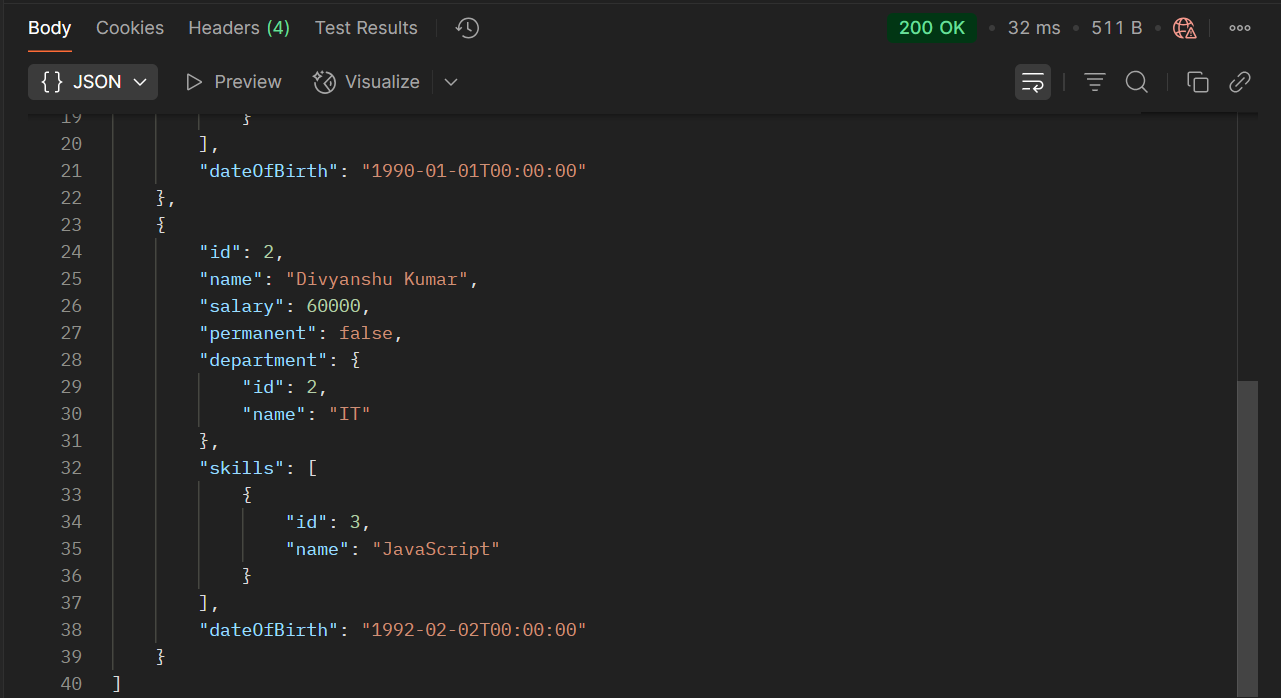
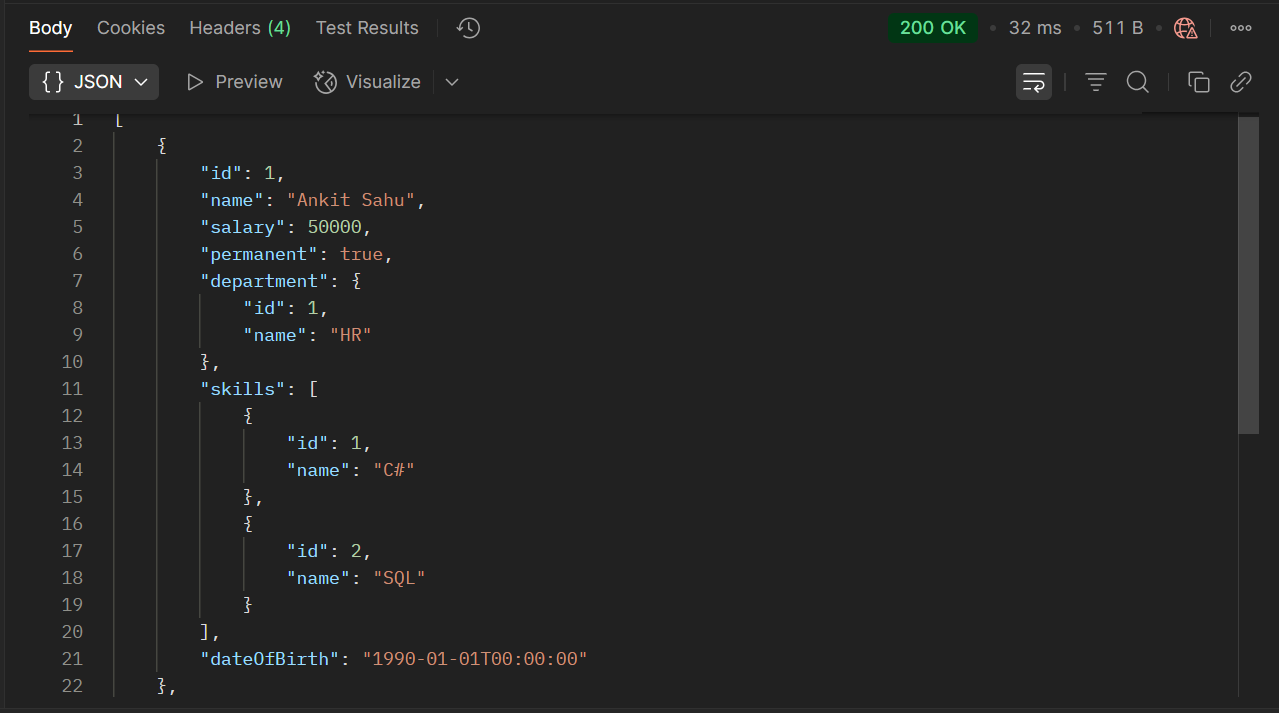
[**CustomExceptionFilter.cs**](http://customexceptionfilter.cs) **:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc; using Microsoft.AspNetCore.Mvc.Filters; using System.IO;   public class CustomExceptionFilter : IExceptionFilter {  public void OnException(ExceptionContext context)  {  var exception = context.Exception;  File.AppendAllText("exceptions.txt", $"{DateTime.Now}: {exception.Message}{Environment.NewLine}");   context.Result = new ObjectResult("An error occurred. Please contact support.")  {  StatusCode = 500  };  context.ExceptionHandled = true;  } } |

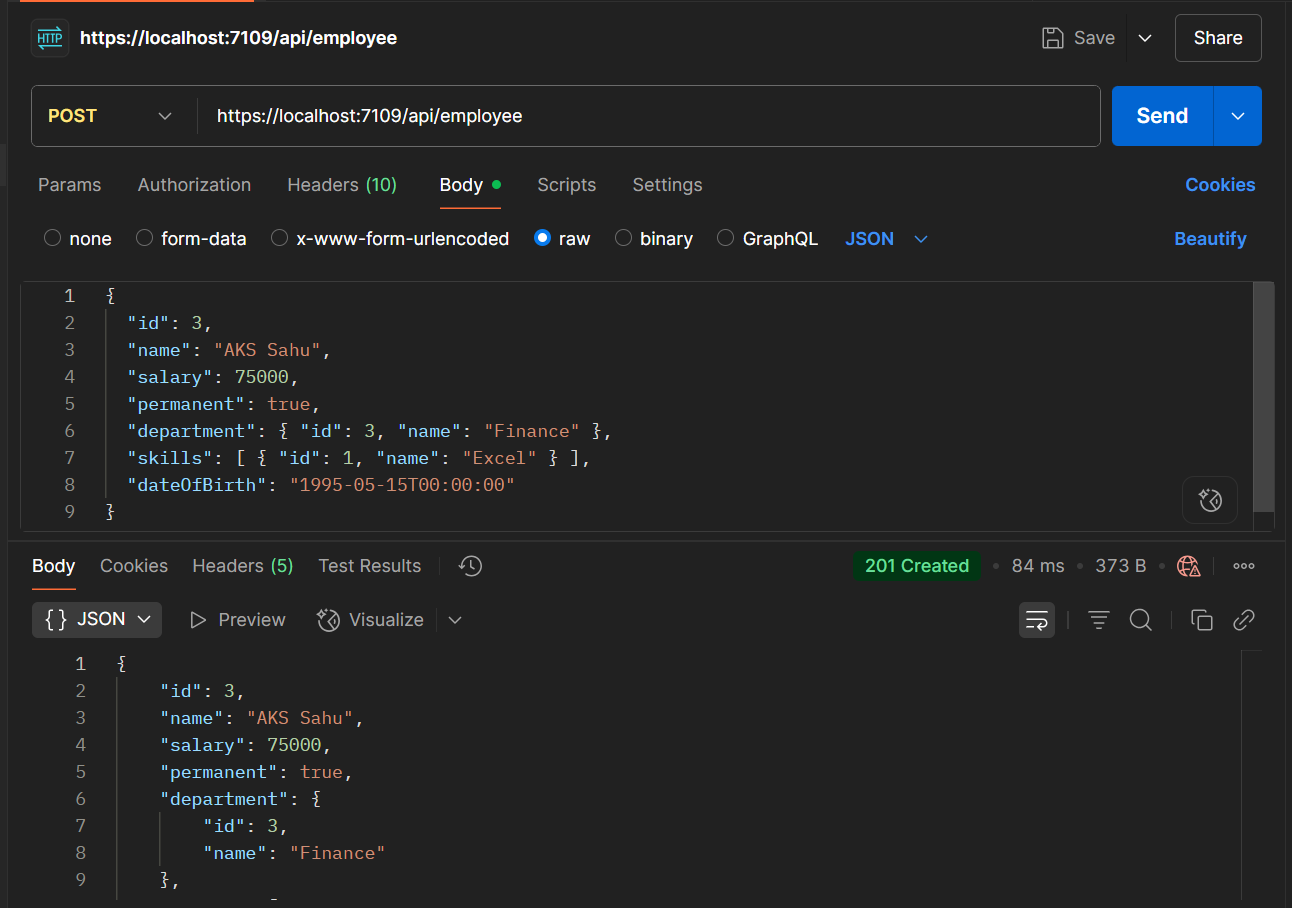
**OUTPUT (POSTMAN):**

**GET : api/employee :**

****

****

**POST : api/employee :**

****