* **Demonstrate creation of an Action method to return list of custom class entity**
  + **Model class creation, Use AllowAnonymous attribute, Use HttpGet action method**

**Answer-->**

## Step 1: Create the Model Class

Let’s create a simple model called Student.

// Models/Student.csnamespace DemoWebAPI.Models

{

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

public string Course { get; set; }

}

}

## Step 2: Create a Controller with [HttpGet] and [AllowAnonymous]

using Microsoft.AspNetCore.Authorization;using Microsoft.AspNetCore.Mvc;using DemoWebAPI.Models;using System.Collections.Generic;

namespace DemoWebAPI.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class StudentController : ControllerBase

{

// Sample data

private static List<Student> students = new List<Student>

{

new Student { Id = 1, Name = "Alice", Course = "Computer Science" },

new Student { Id = 2, Name = "Bob", Course = "Mathematics" },

new Student { Id = 3, Name = "Charlie", Course = "Physics" }

};

// GET: api/student

[HttpGet]

[AllowAnonymous] // 👈 Publicly accessible, no authentication required

public ActionResult<IEnumerable<Student>> GetStudents()

{

return Ok(students);

}

}

}

* **Explain the usage of FromBody attribute**
  + **Read the model object from request, other than the query string parameter**

**Answer-->**

## What is [FromBody]?

The [FromBody] attribute tells ASP.NET Core **to read data from the HTTP request body**, instead of from the URL, query string, or route.

It is typically used for **POST**, **PUT**, or **PATCH** requests when the client sends a **JSON payload** representing a model object.

## Syntax Example:

[HttpPost]

public IActionResult AddStudent([FromBody] Student student)

{

// Process the student object received in the request body

return Ok(student);

}

## Example Request (POST):

**URL**: POST /api/student

**Headers**:

Content-Type: application/json

**Body (raw JSON):**

{

"id": 101,

"name": "Ravi",

"course": "Engineering"

}

ASP.NET automatically deserializes this JSON into a Student object because of [FromBody].

## Without [FromBody]?

If you **omit** [FromBody] in some cases:

ASP.NET may try to read from **query string**, **form data**, or **route values**

You may get a null object if body parsing is expected but not specified

* **Demonstrate Custom filter**

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

**Answer-->**

## 1. Create a Custom Action Filter

### Step: Create a new class that inherits ActionFilterAttribute

using Microsoft.AspNetCore.Mvc.Filters;using Microsoft.Extensions.Logging;using System;

public class RequestLoggingFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

Console.WriteLine($"[RequestLoggingFilter] Request started at: {DateTime.Now}");

base.OnActionExecuting(context);

}

}

## 2. Apply the Filter to a Controller or Action

using Microsoft.AspNetCore.Mvc;

[ApiController]

[Route("api/[controller]")]public class ProductController : ControllerBase

{

[HttpGet]

[RequestLoggingFilter] // 👈 Custom filter added

public IActionResult GetProducts()

{

return Ok(new[] { "Laptop", "Mouse", "Keyboard" });

}

}

This will log a message before the action executes.

## 3. Create a Custom Exception Filter

### Step: Create a class that implements IExceptionFilter

using Microsoft.AspNetCore.Mvc;using Microsoft.AspNetCore.Mvc.Filters;using System;

public class GlobalExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

// we can log the exception here

var response = new

{

Message = "An unexpected error occurred.",

Details = context.Exception.Message

};

context.Result = new ObjectResult(response)

{

StatusCode = 500

};

context.ExceptionHandled = true;

}

}

## 4. Register the Exception Filter Globally in Startup.cs

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers(options =>

{

options.Filters.Add<GlobalExceptionFilter>(); // 👈 Register global exception handler

});

}

## 5. Throw a Custom Exception in Controller

[HttpGet("error")]public IActionResult GetError()

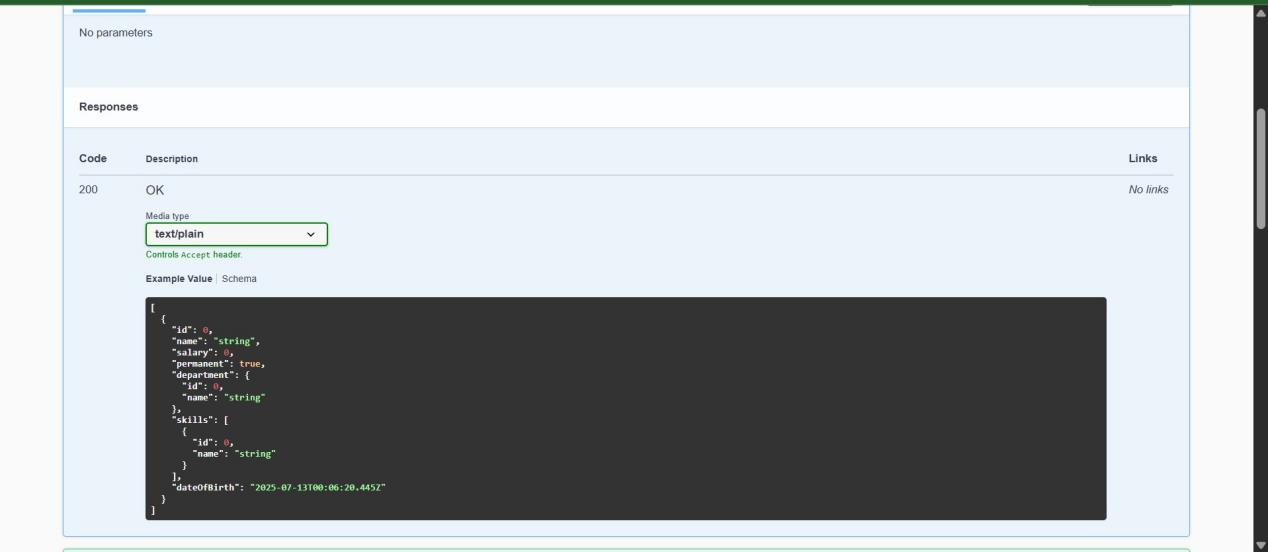
{

throw new Exception("Something broke!");

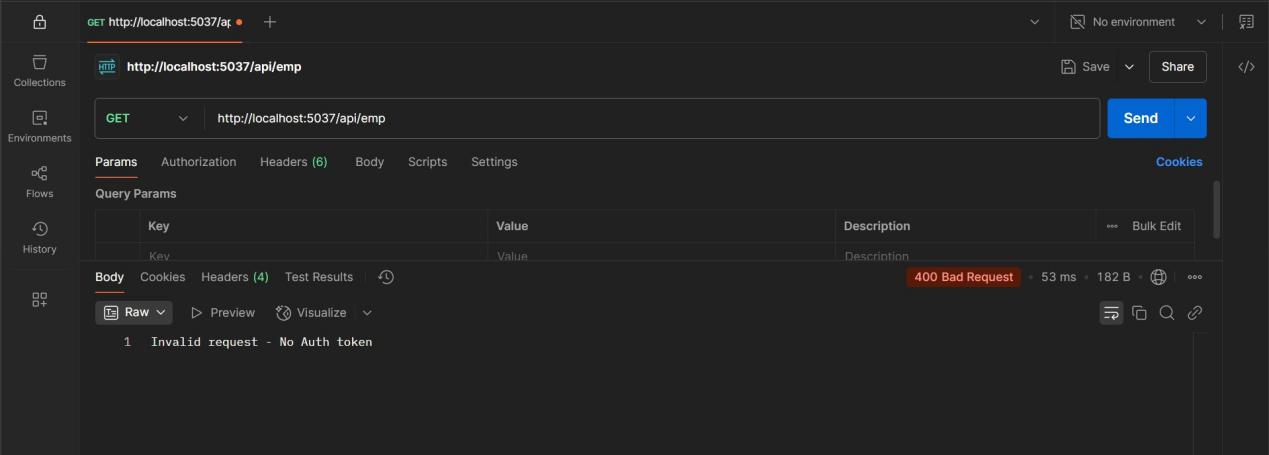
}

**Output:**

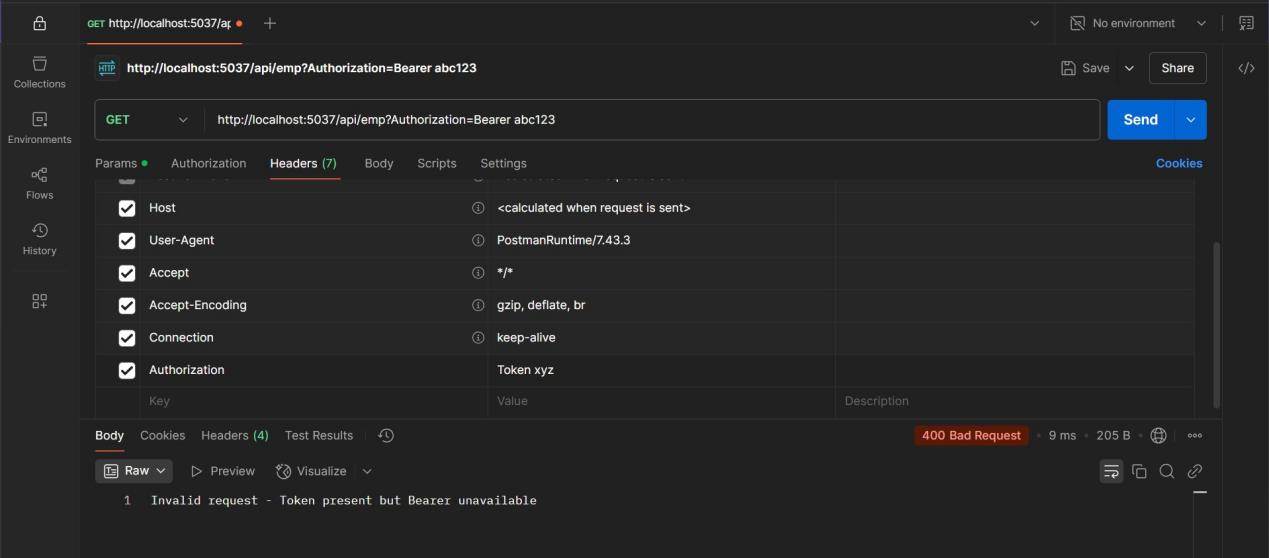
**1)**

****

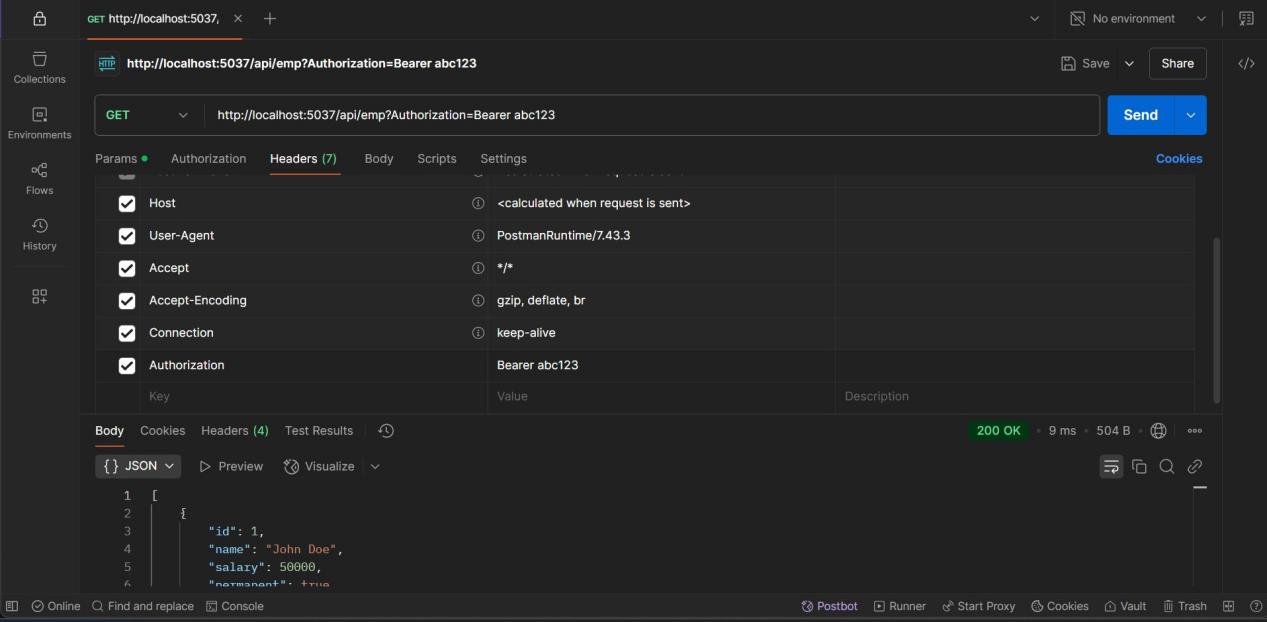
**2)**

**No Auth token:**

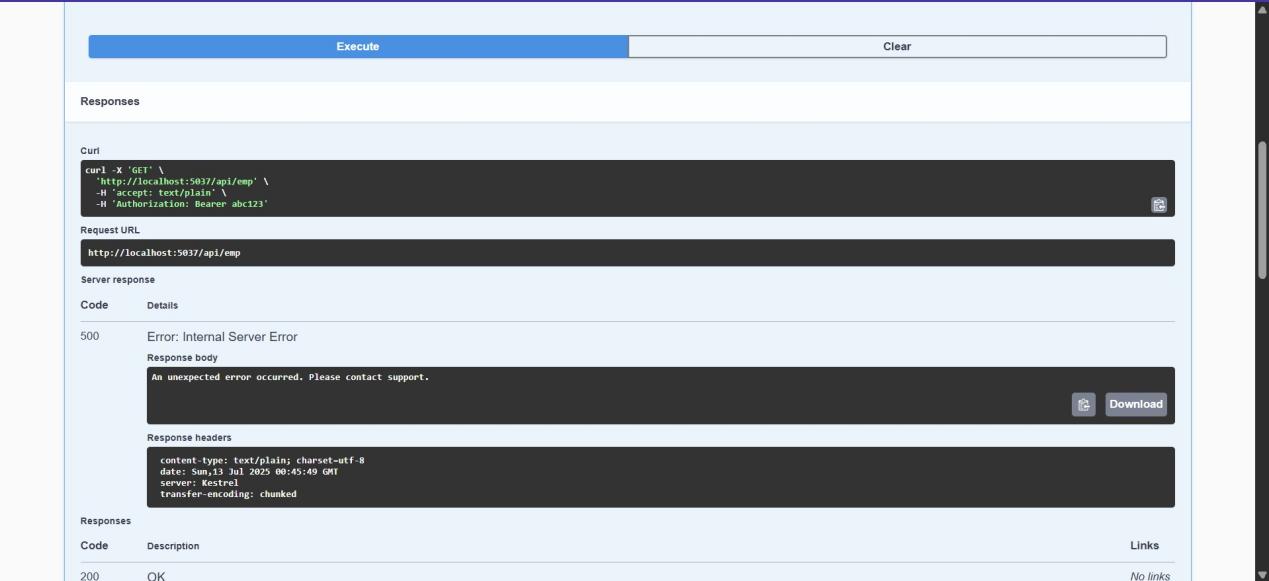
**Auth Token without “Bearer”**

****

**Auth token with “Bearer”**

****

**3)**

****