**1.** **First Web Api using .Net core**

Create a .Net core web application with API template. Use the option to create a controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs.

On creation of the Web API, execute the application and check if the GET action method result is returned as expected.

**Solution:**

* I Started creating a new .NET Core Web API project using Visual Studio.
* I opened Visual Studio and clicked the “Create a new project.”
* Search for and select “ASP.NET Core Web API”.
* Then provide the project name and location.

**In the configuration screen:**

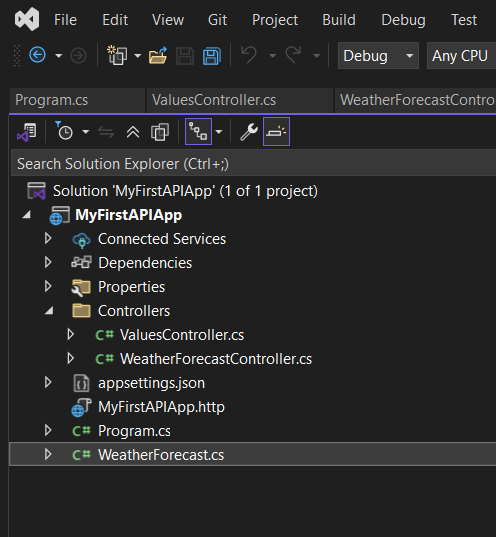
* I chose the desired .NET version.
* The option I used was controllers.
* Click Create.

**Add Controller with Read/Write Support**

* I created a new controller with actions that support CRUD operations.
* Right-click the Controllers folder -> Add -> Controller.
* I choose an API Controller with actions, using Entity Framework.
* Name the controller ValuesController.

\

**Solution:**



[ValuesController.cs](http://valuescontroller.cs)

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace YourNamespace.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

private static List<string> values = new List<string> { "value1", "value2" };

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return Ok(values);

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

if (id < 0 || id >= values.Count)

return NotFound("Value not found");

return Ok(values[id]);

}

[HttpPost]

public ActionResult Post([FromBody] string value)

{

if (string.IsNullOrWhiteSpace(value))

return BadRequest("Value cannot be empty");

values.Add(value);

return CreatedAtAction(nameof(Get), new { id = values.Count - 1 }, value);

}

[HttpPut("{id}")]

public ActionResult Put(int id, [FromBody] string newValue)

{

if (id < 0 || id >= values.Count)

return NotFound("Value not found");

if (string.IsNullOrWhiteSpace(newValue))

return BadRequest("New value cannot be empty");

values[id] = newValue;

return NoContent(); // 204 No Content

}

[HttpDelete("{id}")]

public ActionResult Delete(int id)

{

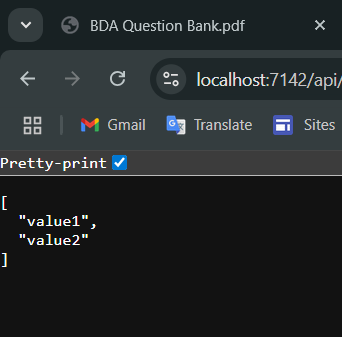
if (id < 0 || id >= values.Count)

return NotFound("Value not found");

values.RemoveAt(id);

return NoContent();}}}

**Result:**



**Solution 2:**

* I created the employeeAPI project.
* I created the class file [employee.cs](http://employee.cs) under the model folder.
* And I created the [employeeController.cs](http://employeecontroller.cs) under the controller.

[employeeController.cs](http://employeecontroller.cs)

using Microsoft.AspNetCore.Mvc;

using EmployeeApi.Models;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeApi.Controllers

{

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

[ApiController]

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>

{

new Employee { Id = 1, Name = "Alice", Department = "HR", Salary = 50000 },

new Employee { Id = 2, Name = "Bob", Department = "IT", Salary = 70000 }

};

[HttpGet]

public ActionResult<IEnumerable<Employee>> GetEmployees() => employees;

[HttpGet("{id}")]

public ActionResult<Employee> GetEmployee(int id)

{

var emp = employees.FirstOrDefault(e => e.Id == id);

if (emp == null) return NotFound();

return emp;

}

[HttpPost]

public ActionResult<Employee> CreateEmployee(Employee employee)

{

employee.Id = employees.Max(e => e.Id) + 1;

employees.Add(employee);

return CreatedAtAction(nameof(GetEmployee), new { id = employee.Id }, employee);

}

[HttpPut("{id}")]

public IActionResult UpdateEmployee(int id, Employee updatedEmployee)

{

var emp = employees.FirstOrDefault(e => e.Id == id);

if (emp == null) return NotFound();

emp.Name = updatedEmployee.Name;

emp.Department = updatedEmployee.Department;

emp.Salary = updatedEmployee.Salary;

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult DeleteEmployee(int id)

{

var emp = employees.FirstOrDefault(e => e.Id == id);

if (emp == null) return NotFound();

employees.Remove(emp);

return NoContent();

}

}

}

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

namespace EmployeeApi.Models

{

public class Employee

{

public int Id { get; set; }

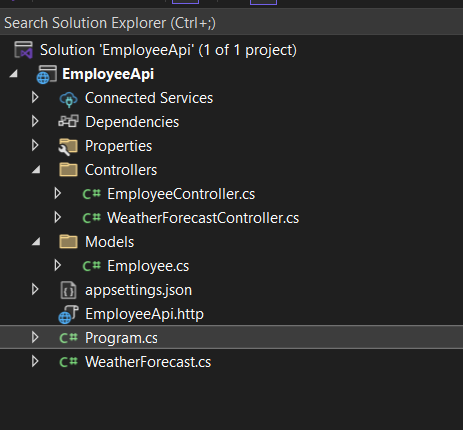
public required string Name { get; set; }

public required string Department { get; set; }

public decimal Salary { get; set; }

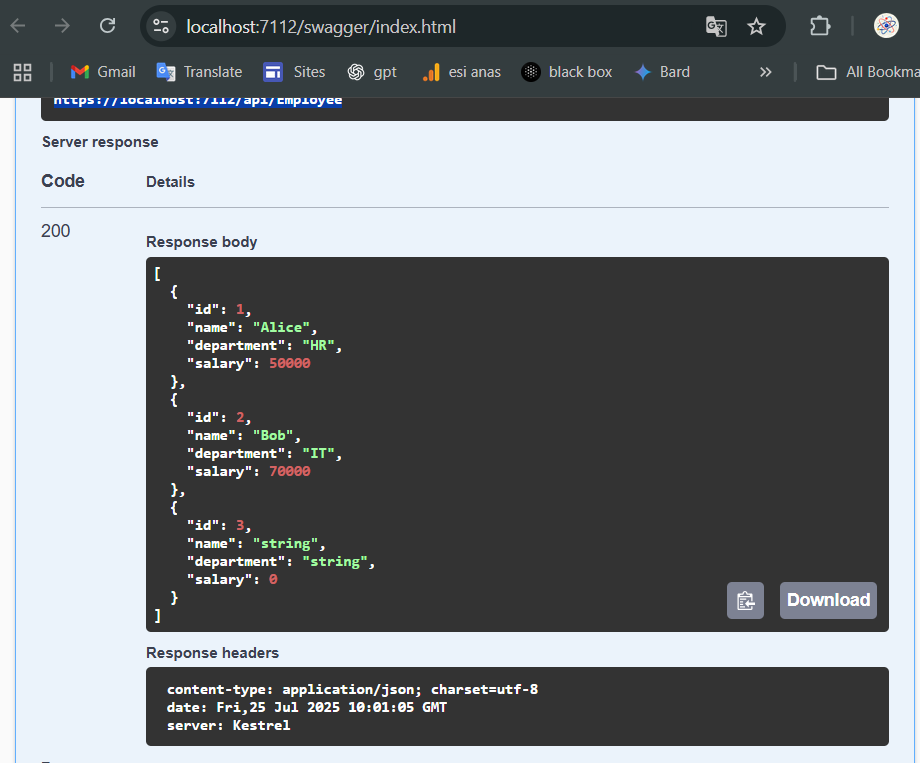
}

}

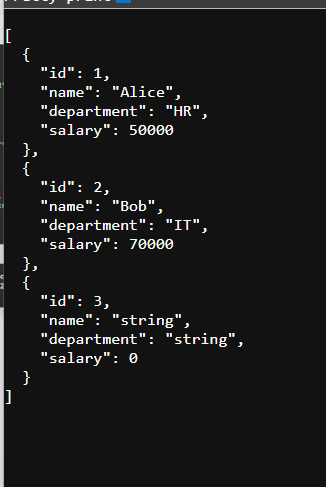


**Result:**

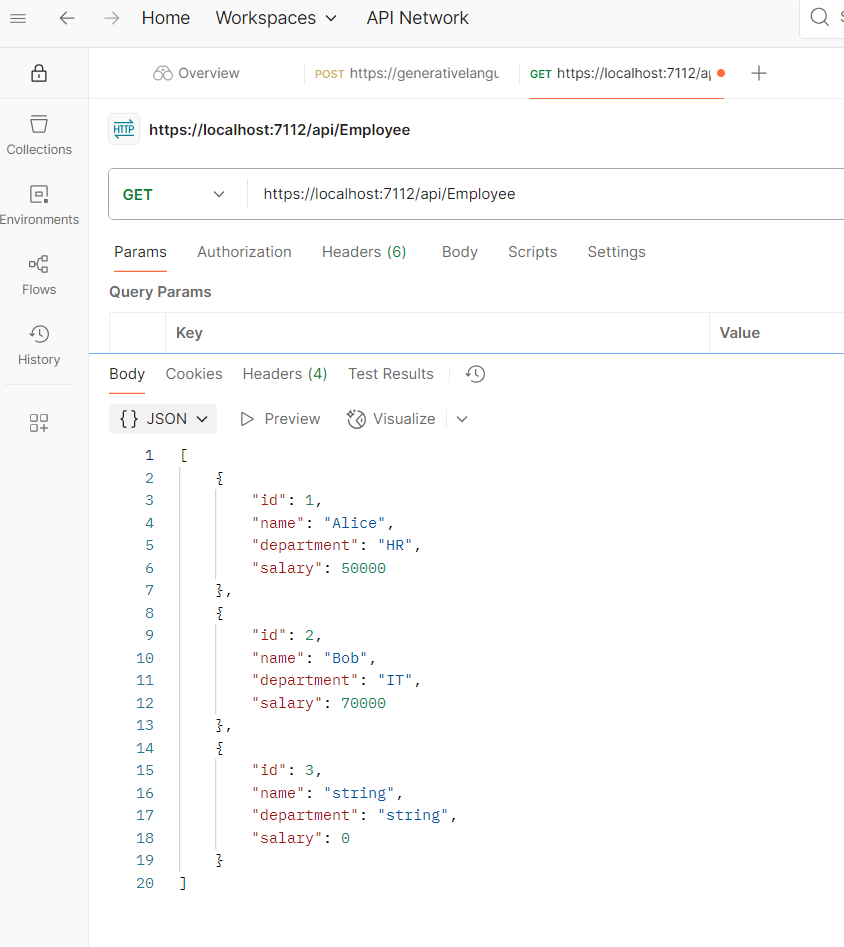
**With swagger:**



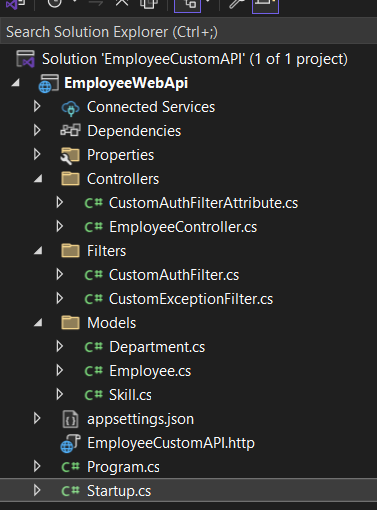
**With local host link:**



**With postman:**

****

**Solution 3:**

****

* I created the project structure in the above format and I added the filters and models for this solution and controllers also.
* And I added the [startup.cs](http://startup.cs) to include the services.

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

using EmployeeCustomAPI.Models;

using EmployeeWebApi.Filters;

using EmployeeWebApi.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

namespace EmployeeWebApi.Controllers

{

[ApiController]

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

[CustomAuthFilter]

[ServiceFilter(typeof(CustomExceptionFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees;

public EmployeeController()

{

if (\_employees == null)

{

\_employees = GetStandardEmployeeList();

}

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

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 = "Jane",

Salary = 60000,

Permanent = false,

Department = new Department { Id = 2, Name = "IT" },

Skills = new List<Skill> { new Skill { Id = 3, Name = "Java" } },

DateOfBirth = new DateTime(1992, 5, 15)

}

};

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public IActionResult Get()

{

return Ok(\_employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Employee emp)

{

var existing = \_employees.Find(e => e.Id == id);

if (existing == null)

return NotFound();

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.Permanent = emp.Permanent;

existing.DateOfBirth = emp.DateOfBirth;

return Ok(existing);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var emp = \_employees.Find(e => e.Id == id);

if (emp == null)

return NotFound();

\_employees.Remove(emp);

return NoContent();

}

[HttpGet("standard")]

public ActionResult<List<Employee>> GetStandard()

{

return GetStandardEmployeeList();

}

}

}

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

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeCustomAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public CustomExceptionFilter()

{

}

public void OnException(ExceptionContext context)

{

context.Result = new ObjectResult(new

{

Message = "Something went wrong.",

Exception = context.Exception.Message

})

{

StatusCode = 500

};

}

}

}

[CustomExceptionfilter.cs](http://customexceptionfilter.cs)

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.AspNetCore.Mvc;

using System;

namespace EmployeeWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

context.Result = new ObjectResult(new

{

Error = "Something went wrong.",

Details = context.Exception.Message

})

{

StatusCode = 500

};

}}}

In the models i included the department, skills and employee class files

[department.cs](http://department.cs)

namespace EmployeeCustomAPI.Models

{

public class Department

{

public int Id { get; set; }

public required string Name { get; set; }

}

}

[skills.cs](http://skills.cs)

namespace EmployeeWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public required string Name { get; set; }

}

}

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

using EmployeeCustomAPI.Models;

using System;

using System.Collections.Generic;

namespace EmployeeWebApi.Models

{

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; }

}

}

**And in the main project folder i updated the** [**program.cs**](http://process.cs) **and add the** [**startup.cs**](http://startup.cs)

[program.cs](http://process.cs)

using EmployeeWebApi.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

[Startup.cs](http://startup.cs)

using EmployeeWebApi.Filters;

public class Startup

{

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddScoped<CustomExceptionFilter>();

services.AddEndpointsApiExplorer();

services.AddSwaggerGen();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseSwagger();

app.UseSwaggerUI();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

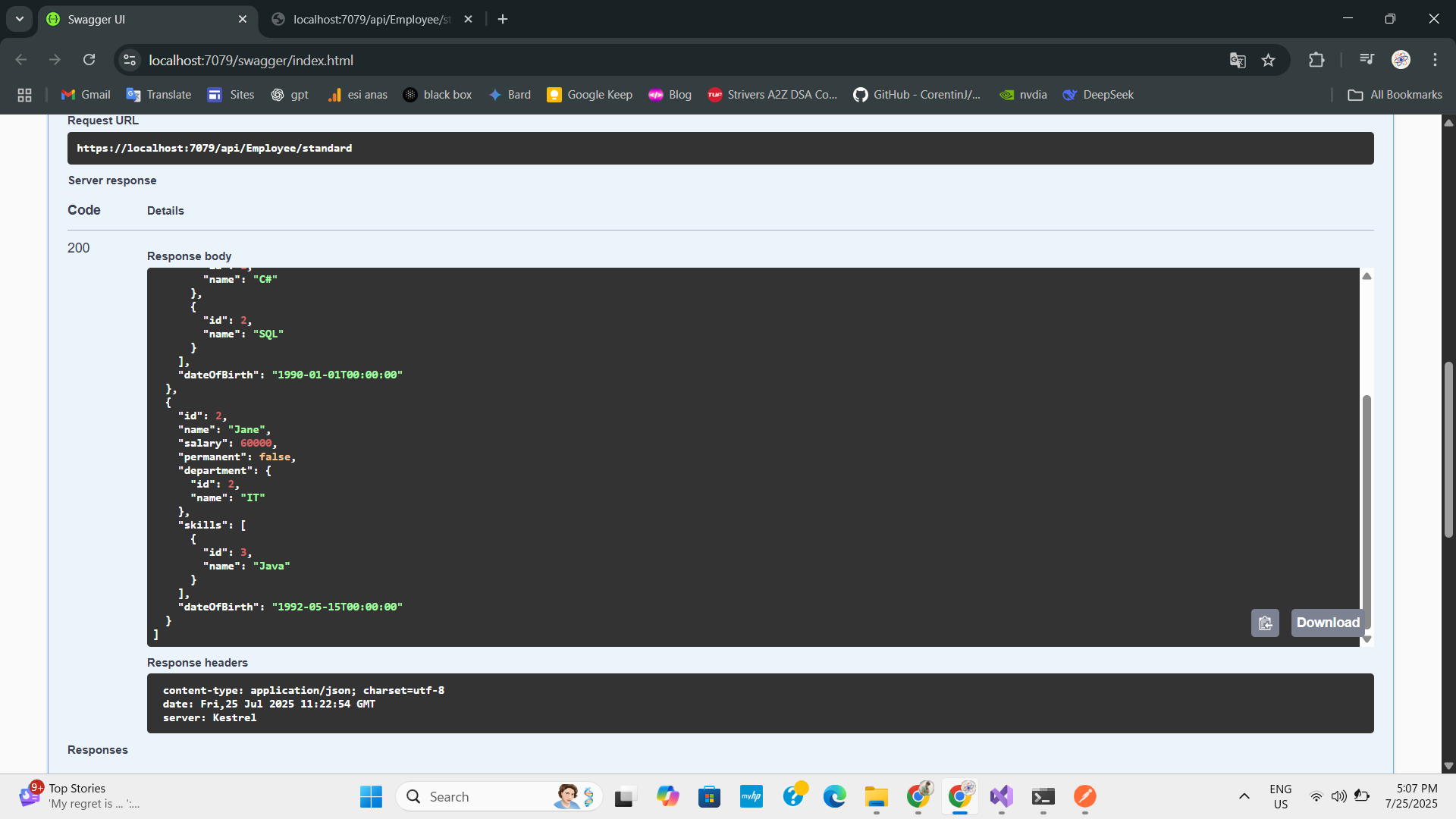
});

}

}

**Result:**

**In swagger by using GET method:**

****

**With postman:**

