**Week4: ASP.NET Core Web API:**

**1.Web API Hands\_on:**

[**TodoItem.cs**](http://todoitem.cs)**:**

namespace TodoApi.Models

{

public class TodoItem

{

public long Id { get; set; }

public string Name { get; set; }

public bool IsComplete { get; set; }

}

}

[**TodoConntext.cs**](http://todoconntext.cs)**:**

using Microsoft.EntityFrameworkCore;

using TodoApi.Models;

namespace TodoApi.Data

{

public class TodoContext : DbContext

{

public TodoContext(DbContextOptions<TodoContext> options)

: base(options) { }

public DbSet<TodoItem> TodoItems { get; set; }

}

}

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

using TodoApi.Data;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<TodoContext>(opt =>

opt.UseInMemoryDatabase("TodoList"));

builder.Services.AddControllers();

var app = builder.Build();

app.MapControllers();

app.Run();

[**TodoController.cs**](http://todocontroller.cs)**:**

using Microsoft.AspNetCore.Mvc;

using TodoApi.Models;

using TodoApi.Data;

using Microsoft.EntityFrameworkCore;

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

[ApiController]

public class TodoItemsController : ControllerBase

{

private readonly TodoContext \_context;

public TodoItemsController(TodoContext context)

{

\_context = context;

}

// GET: api/todoitems

[HttpGet]

public async Task<ActionResult<IEnumerable<TodoItem>>> GetTodoItems()

{

return await \_context.TodoItems.ToListAsync();

}

// GET: api/todoitems/5

[HttpGet("{id}")]

public async Task<ActionResult<TodoItem>> GetTodoItem(long id)

{

var todoItem = await \_context.TodoItems.FindAsync(id);

if (todoItem == null)

return NotFound();

return todoItem;

}

// POST: api/todoitems

[HttpPost]

public async Task<ActionResult<TodoItem>> PostTodoItem(TodoItem item)

{

\_context.TodoItems.Add(item);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetTodoItem), new { id = item.Id }, item);

}

// PUT: api/todoitems/5

[HttpPut("{id}")]

public async Task<IActionResult> PutTodoItem(long id, TodoItem item)

{

if (id != item.Id)

return BadRequest();

\_context.Entry(item).State = EntityState.Modified;

await \_context.SaveChangesAsync();

return NoContent();

}

// DELETE: api/todoitems/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteTodoItem(long id)

{

var todoItem = await \_context.TodoItems.FindAsync(id);

if (todoItem == null)

return NotFound();

\_context.TodoItems.Remove(todoItem);

await \_context.SaveChangesAsync();

return NoContent();

}

}

**2.Web API Hands\_on:**

**WeatherForecastController**

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class ValuesController : ControllerBase

{

// GET: api/values

[HttpGet]

public IActionResult Get()

{

return Ok(new string[] { "value1", "value2" });

}

// GET: api/values/5

[HttpGet("{id}")]

public IActionResult Get(int id)

{

return Ok("value" + id);

}

// POST: api/values

[HttpPost]

public IActionResult Post([FromBody] string value)

{

return Ok("Added: " + value);

}

// PUT: api/values/5

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string value)

{

return Ok("Updated: " + value);

}

// DELETE: api/values/5

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

return Ok("Deleted: " + id);

}

}

**Configure Swagger for API Testing:**

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

**3.Web API Hands\_on:**

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

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public string Result { get; set; }

public string Department { get; set; }

public List<string> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

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

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee { Id = 1, Name = "Alice", Salary = 50000, Result = "Pass", Department = "IT", Skills = new List<string>{"C#", "SQL"}, DateOfBirth = new DateTime(1990,1,1) },

new Employee { Id = 2, Name = "Bob", Salary = 60000, Result = "Pass", Department = "HR", Skills = new List<string>{"Excel", "Communication"}, DateOfBirth = new DateTime(1988,5,15) }

};

}

**Implement Read/Write Functions:**

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>();

[HttpGet]

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

public IActionResult GetStandard()

{

if (!employees.Any())

employees = GetStandardEmployeeList();

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

employees.Add(emp);

return Ok(emp);

}

// Add PUT and DELETE as needed

}

**Creating the filter:**

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);

}

}

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

// Log exception to a file

File.AppendAllText("exceptions.log", context.Exception.ToString());

context.Result = new ObjectResult("An error occurred. Please contact admin.")

{

StatusCode = 500

};

}

}

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

services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

**4.Web API Hands\_on:**

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

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public string Department { get; set; }

}

**Data Store:**

public static List<Employee> Employees = new List<Employee>

{

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

new Employee { Id = 2, Name = "Bob", Salary = 60000, Department = "IT" }

};

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

[ApiController]

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

public class EmployeeController : ControllerBase

{

// PUT: api/employee/{id}

[HttpPut("{id}")]

public IActionResult UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

// Validation: id <= 0

if (id <= 0)

return BadRequest("Invalid employee id");

// Find employee by id

var employee = Employees.FirstOrDefault(e => e.Id == id);

// If not found, BadRequest

if (employee == null)

return BadRequest("Invalid employee id");

// Update hardcoded list

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Department = updatedEmployee.Department;

// Return updated employee data

return Ok(employee);

}

}

**5.Web API Hands\_on:**

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

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public string Department { get; set; }

}

**Employee Data:**

public static List<Employee> Employees = new List<Employee>

{

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

new Employee { Id = 2, Name = "Bob", Salary = 60000, Department = "IT" }

};

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

[ApiController]

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

public class EmployeeController : ControllerBase

{

// PUT: api/employee/{id}

[HttpPut("{id}")]

public IActionResult UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

// Validate id

if (id <= 0)

return BadRequest("Invalid employee id");

// Find employee by id

var employee = Employees.FirstOrDefault(e => e.Id == id);

// If not found, return BadRequest

if (employee == null)

return BadRequest("Invalid employee id");

// Update employee details

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Department = updatedEmployee.Department;

// Return updated employee

return Ok(employee);

}

}