This cheat sheet is for the course Learn C# Full Stack Development with Angular and ASP.NET by Jannick Leismann.

ASP.NET CRUD API CONTROLLER ENDPOINT

This code represents an ASP.NET Core Web API controller for managing employees. The controller uses an IEmployeeRepository to interact with the underlying data store, following the Repository design pattern.

Namespaces

```
EmployeeManagement.Models;
EmployeeManagement.Repositories;
Microsoft.AspNetCore.Mvc;
Microsoft.EntityFrameworkCore.InMemory.Storage.Internal;
```

These namespaces are used to include the necessary models, repository interfaces, and ASP.NET Core MVC functionalities.

Controller

```
[Route("api/[controller]")]
[ApiController]
public class EmployeesController : ControllerBase
{
    private readonly IEmployeeRepository _employeeRepository;

    public EmployeesController(IEmployeeRepository employeeRepository)
    {
        _employeeRepository = employeeRepository;
    }
}
```

Given that it has the **[ApiController]** attribute defined, the EmployeesController class is an **API controller**.

Since the controller is called E**mployeesController**, the route specified by the **[Route("api/[controller]")]** property will be **api/employees**.

By using **constructor injection**, the IEmployeeRepository **dependency** is **added** to the controller.

Actions

GetAllEmployeesAsync

```
[HttpGet]

public async Task<ActionResult<IEnumerable<Employee>>>
GetAllEmployeesAsync()

{

    var employees = await _employeeRepository.GetAllAsync();

    return Ok(employees);

}
```

Handles **GET** requests to **api/employees**.

Calls the repository to get all employees and returns them with a **200 OK** response.

GetEmployeeById

```
[HttpGet("{id}")]

public async Task<ActionResult<Employee>> GetEmployeeById(int id)

{

    var employee = await _employeeRepository.GetByIdAsync(id);

    if (employee == null)

    {

        return NotFound();

    }

    return Ok(employee);

}
```

Handles GET requests to **api/employees/{id}**.

Calls the repository to get an employee by the given id. If the employee doesn't exist, it returns a **404 Not Found** response. Otherwise, it returns the employee with a **200 OK response**.

CreateEmployee

```
[HttpPost]
public async Task<ActionResult<Employee>> CreateEmployee(Employee
employee)
{
    await _employeeRepository.AddEmployeeAsync(employee);
    return CreatedAtAction(nameof(GetEmployeeById), new { id = employee.Id
}, employee);
}
```

Handles **POST** requests to **api/employees**.

Calls the repository to add a new employee. It returns a **201 Created** response, with the **Location header** set to the URI of the newly created employee, using the **GetEmployeeById** action to generate the URI.

DeleteEmployeeById

```
[HttpDelete("{id}")]
public async Task<ActionResult> DeleteEmployeeById(int id)
{
    await _employeeRepository.DeleteEmployeeAsync(id);
    return NoContent();
}
```

Handles **DELETE** requests to **api/employees/{id}**.

Calls the repository to delete an employee by the given id. It returns a **204 No Content** response.

UpdateEmployeeAsync

```
[HttpPut("{id}")]
public async Task<ActionResult<Employee>> UpdateEmployeeAsync(int id,
Employee employee)
{
    if (id != employee.Id)
    {
        return BadRequest();
    }

    await _employeeRepository.UpdateEmployeeAsync(employee);

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

Handles **PUT** requests to **api/employees/{id}**.

Calls the repository to update an existing employee. If the id in the URL doesn't match the id of the employee object, it returns a **400 Bad Request** response.

If the update is successful, it returns a **201 Created** response with the URI of the updated employee.

The EmployeesController provides standard CRUD operations for employee management:

- **GET** to retrieve all employees or a specific employee by ID.
- **POST** to create a new employee.
- **PUT** to update an existing employee.
- **DELETE** to remove an employee by ID.

The controller delegates the actual data operations to the **IEmployeeRepository**, promoting a **separation of concern**s and making the code easier to test and maintain.