INT204 server side II

1.api/employees GET ALL Employees

- EmployeeControllor

```
@GetMapping("")
public List<Employee> getEmployees() {
    return service.getAllEmployees();
}
```

- EmployeeService

```
1 usage
public List<Employee> getAllEmployees() {
    return repository.findAll();
} // select * from employee
```

ทำการส่ง request ไปที่ /employees ,controller จะสั่งให้ตัว employeeservice ไปเรียกใช้ method get แล้วนำข้อมูล employee ทั้งหมดมาแสดง

2.api/employees/1 GET an employee with id = 1

- EmployeeControllor

```java
@GetMapping("/{employeeNumber}")
public Employee getEmployee(@PathVariable int employeeNumber) {
    return service.getEmployee(employeeNumber);
}
```

- EmployeeService

```java
1 usage
public Employee updateEmployees(int employeeNumber, Employee updateEmployee) {
    Employee employee = repository.findById(employeeNumber).map(e -> mapEmployee(e, updateEmployee))
            .orElseThrow(() -> new RuntimeException(employeeNumber + " does not exist !!"));
    return repository.saveAndFlush(employee);
}
```

ทำการส่ง request ไปที่ /employees/1 , controller ที่เป็น id = 1 แล้วเข้าไปหาใน employee และมีข้อมูลตรงกัน แล้วนำข้อมูลของidนั้นมาแสดง หากไม่มีข้อมูลที่ตรงกับที่หา จะแสดงข้อความที่กำหนดว่า employeenumberนั้นไม่มี

3.api/offices/1/employees POST Add new employee for office id = 1

- OfficesController

```java
@PostMapping("/{officeCode}/employees")
@ResponseStatus(HttpStatus.CREATED)
public Employee insertEmployeeOffice(@PathVariable String officeCode, @RequestBody Employee newEmployee) {
    return service.insertEmployeeOffice(officeCode, newEmployee);
}
```

- OfficeService

```java
1 usage
public Employee insertEmployeeOffice(String officeCode, Employee newEmployee) {
    Office office = getOffice(officeCode);
    newEmployee.setOffice(office);
    return employeeRepository.saveAndFlush(newEmployee);
}
```

ทำการใช้ method POST เมื่อถูก request ไปที่ /offices/1/employees ,controller จะนำค่า officeCode ไปหาข้อมูลใน office ที่มี officeCode ที่มีข้อมูลตรงกัน และนำ request body ไปสร้างตัว new employee และ บันทึกข้อมูลใหม่

4. api/employees/1 PUT Update an employee with id = 1

- EmployeeController

```java
@PutMapping("/{employeeNumber}")
public Employee updateEmployees(@RequestBody Employee updateEmployee, @PathVariable int
        employeeNumber) {
    if (employeeNumber == updateEmployee.getId()) {
        return service.updateEmployees(employeeNumber, updateEmployee);
    } else {
        throw new RuntimeException("EmployeeNumber is not match !!!");
    }
}
```

- EmployeeService

```java
1 usage
public Employee updateEmployees(int employeeNumber, Employee updateEmployee) {
    Employee employee = repository.findById(employeeNumber).map(e -> mapEmployee(e, updateEmployee))
            .orElseThrow(() -> new RuntimeException(employeeNumber + " does not exist !!"));
    return repository.saveAndFlush(employee);
}
```

ทำการใช้ method POST เมื่อถูก request ไปที่ /employees/1/,controller ที่เป็น id = 1 และไปหาที่ employees ถ้าเจอข้อมูล จะนำค่าใน request body ไปทำการ updateข้อมูล และทำการบันทึกข้อมูล

5. api/employees/1 DELETE Delete an employee with id = 1

- EmployeeController

```
    @DeleteMapping("/{employeeNumber}")
    public void deleteEmployeesByNumber(@PathVariable int employeeNumber) {
        service.deleteEmployeesByNumber(employeeNumber);
    }
}
```

- EmployeeService

```
    1 usage
    public void deleteEmployeesByNumber(int employeeNumber) {
        repository.deleteById(employeeNumber);
    }
```

เมื่อทำการส่ง request ไปที่ /employees/1 ,controller ที่เป็น id =1 ไปหาข้อมูลที่ employee ที่มี id ตรงกัน และจะทำการลบ employeeนั้นออก