


BITA/5/21/036/TZ

QUESTION 2

REPORT AND SOURCE CODE

a)

Meet the Spring team this August at SpringOne.

 **spring initializr**

Project

☐ Gradle - Groovy

☐ Gradle - Kotlin

☒ Maven

Language

☒ Java

☐ Kotlin

☐ Groovy

Spring Boot

☐ 3.2.0 (SNAPSHOT)

☐ 3.2.0 (M1)

☐ 3.1.3 (SNAPSHOT)

☒ 3.1.2

☐ 3.0.10 (SNAPSHOT)

☐ 3.0.9

☐ 2.7.15 (SNAPSHOT)

☐ 2.7.14

Project Metadata

Group

Dependencies

ADD DEPENDENCIES... CTRL + B

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

b)

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 20 ☒ 17 ☐ 11 ☐ 8

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

JDBC API SQL
Database Connectivity API that defines how a client may connect and query a database.

MySQL Driver SQL
MySQL JDBC driver.

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

c)

```
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <scope>runtime</scope>
</dependency>
```

d)

```
package com.Employee Management System;

import lombok.Data;

import javax.persistence.*;

@Table
@Entity
@Data

public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int employee_number;
    private String first_name;
    private String last_name;
    private String email;

    public Customer() {
    }

    public Customer(int customer_ID, String phone_no, String cus_name, String food_name, String email) {
        employee_number = employee_number;
        this.first_name = first_name;
        this.last_name = last_name;
        this.email = email;
    }
}
```

```

public Customer() {
}

public Customer(int customer_ID, String phone_no, String cus_name, String food_name, String email) {
    employee_number = employee_number;
    this.first_name = first_name;
    this.last_name = last_name;
    this.email = email;
}

public int employee_number() {
    return employee_number;
}

public void set employee_number(int employee_number) {
    employee_number = employee_number;
}

public String get first_name() {
    return first_name;
}

public void set first_name(String first_name) {
    this.first_name = first_name;
}

public String get last_name() {
    return last_name;
}

public void set last_name(String last_name) {
    this.last_name = last_name;
}

```

```

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

```

f)

```

package com.Employee Management System.Repository;

import com.Resta.RestaManagement.Module.Employee;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomerRepository extends JpaRepository<Customer,Integer> {
}

```

h)

```
package com.Employee Management System.Controller;

import com.Resta.Employee Management System.ResourceNotFoundException;
import com.Resta.Employee Management System.Module.employee;
import com.Resta.Employee Management system.Repository.employeeRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
@CrossOrigin
@RestController
@RequestMapping("/api/emp")

public class employee Controller {

    @Autowired
    public employeeRepository employee
    @GetMapping("/employee")
    private List<Employee>viewEmployee(){
        return employeeRepository.findAll();
    }

    @PostMapping("/employee")
    public Employee addEmployee(@RequestBody Employee employee){
        return employeeRepository.save(employee);
    }

    @GetMapping("/employee/{id}")
    public ResponseEntity<Employee> getEmployeeByID(@PathVariable int id) {
        Employee employee = Employee.findById(id)
```

```

@GetMapping("/employee/{id}")
public ResponseEntity<Employee> getEmployeeByID(@PathVariable int id) {
    Employee employee = Employee.findById(id)
        .orElseThrow(() -> new ResourceNotFoundException("Invalid Id: " + id));
    return ResponseEntity.ok(employee);
}

@DeleteMapping("/employee/{id}")
public ResponseEntity<Map<String,Boolean>> deleteEmployee(@PathVariable int id)
{
    //finding the customer with provided id
    Employee employee = EmployeeRepository.findById(id)

        // throw resource not found exception if not found
        .orElseThrow(() -> new ResourceNotFoundException("Invalid Id: "+id));

    // deleting the customer if found
    employeeRepository.delete(employee);

    // display message for success deletion
    Map<String,Boolean> response = new HashMap<>();
    response.put("employee was Deleted : ",Boolean.TRUE);
    return ResponseEntity.ok(response);
}

@PutMapping("/employee/{id}")
public ResponseEntity<Employee> updateEmployee(@PathVariable("id") int employee_id, @RequestBody Employee
{

    Employee employee1 = employeeRepository.findById(employee_number)
        .orElseThrow(() -> new ResourceNotFoundException("Invalid Id: "+employee_id));

```

```

    employee1.setEmployee_number(employee.getEmployeeByID());
    employee1.setEmail(employee.getEmail());
    employee1.setFirst_name(employee.getFirst_name());
    employee1.setLast_name(employee.getLast_name());

    // saving the new values

    return ResponseEntity.ok(employeeRepository.save(employee1));
}

```

NewImport

POST localhost:8080/api/emp/Employee_db

localhost:8080/api/emp/Employee_db

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

noneform-datax-www-form-urlencodedrawbinaryJSON

```
1  {
2    "employee_number": "1",
3    "email": "kazija@gmail.com",
4    "first_name": "kei",
5    "last_name_name": "mohd"
6  }
7
8
```

BodyCookiesHeaders (8)Test Results404 Not Found10 ms368 BSave Response

PrettyRawPreviewVisualizeJSON