**Database configuration**

|  |
| --- |
| create database dans\_mid\_test;  use dans\_mid\_test;  CREATE TABLE department (  dept\_no varchar(3) NOT NULL,  dept\_name varchar(45),  primary key (dept\_no )  );  create table employee(  emp\_no VARCHAR (15),  first\_name VARCHAR (45),  last\_name VARCHAR(45),  gender ENUM ('MALE', 'FEMALE'),  birth\_date DATE,  dept\_no varchar(3),  foreign key (dept\_no) references department(dept\_no)  ); |

**setup jdbc**

|  |
| --- |
| spring.jpa.hibernate.ddl-auto=none  spring.datasource.url=jdbc:mysql://localhost:3306/dans\_mid\_test  spring.datasource.username=root  spring.datasource.password=root  logging.level.root=WARN |

**maven dependency**

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <scope>runtime</scope>  <optional>true</optional>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <scope>runtime</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->    <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>5.3.4</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>5.4.28.Final</version>  </dependency>  <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->  <dependency>  <groupId>com.h2database</groupId>  <artifactId>h2</artifactId>  <version>1.4.200</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>net.sf.supercsv</groupId>  <artifactId>super-csv</artifactId>  <version>2.4.0</version>  </dependency>  </dependencies> |

**Entity**  
Employee

|  |
| --- |
| @Id  **private** String emp\_no;  **private** String first\_name;  **private** String last\_name;  **private** String gender;  **private** Date birth\_date;  **private** String dept\_no;  **public** Employee() {  // **TODO** Auto-generated constructor stub  }  **public** Employee(String emp\_no, String first\_name, String last\_name, String gender, Date birth\_date, String dept\_no) {  **super**();  **this**.emp\_no = emp\_no;  **this**.first\_name = first\_name;  **this**.last\_name = last\_name;  **this**.gender = gender;  **this**.birth\_date = birth\_date;  **this**.dept\_no = dept\_no;  } |

Department

|  |
| --- |
| **public** **class** Department {  @Id  **private** String dept\_no;  **private** String dept\_name;    **public** Department() {  // **TODO** Auto-generated constructor stub  }  **public** Department(String dept\_no, String dept\_name) {  **super**();  **this**.dept\_no = dept\_no;  **this**.dept\_name = dept\_name;  } |

**Repository**  
EmployeeRepository

|  |
| --- |
| **public** **interface** EmployeeRepository **extends** JpaRepository<Employee, String>{    } |

DepartmentRepository

|  |
| --- |
| **public** **interface** DepartmentRepository **extends** JpaRepository<Department, String>{  } |

**Service**EmployeeService

|  |
| --- |
| **public** **interface** EmployeeService {    **public** String createEmployee(@RequestBody Employee employee);  **public** Employee updateEmployee(@RequestBody Employee employee);  **public** List<Employee> deleteEmployee(@PathVariable String id);  **public** List<Employee> retrieveAllEmployee();  **public** Employee retrieveEmployeeId(@PathVariable String id);  **public** **void** downloadAllEmployee(HttpServletResponse response) **throws** IOException;  } |

EmployeeServiceImpl  
createEmployee

|  |
| --- |
| @Override  @PostMapping("/Employee")  **public** String createEmployee(@RequestBody Employee employee) {  // **TODO** Auto-generated method stub  // used for validation  **try** {  **if** (employee.getFirst\_name()==**null**) {  **throw** **new** NullPointerException("Please fill in first name value");  }  **if** (employee.getLast\_name()==**null**) {  **throw** **new** NullPointerException("Please fill in last name value");  }  **if** (employee.getBirth\_date()==**null**) {  **throw** **new** NullPointerException("Please fill in birth date value");  }  **if** (employee.getGender()==**null** ) {  **throw** **new** NullPointerException("Please fill in first name date value");  }  }  **catch** (Exception e) {  **return** e.toString();  }    // setup date format for emp\_id  SimpleDateFormat formatter = **new** SimpleDateFormat("ddMMYYYY");  String empDate = formatter.format(employee.getBirth\_date());    // setup random number  Random rand = **new** Random();  String randomNumber = String.*format*("%04d", rand.nextInt(10000));    // merge all the manipulated data  String emp\_no = employee.getDept\_no() + empDate + randomNumber;  employee.setEmp\_no(emp\_no);  employeeRepository.save(employee);    **return** employee.getFirst\_name() + " " + employee.getLast\_name() + " has been added to database";    } |

  
updateEmployee

|  |
| --- |
| @PutMapping("/Employee")  **public** Employee updateEmployee(@RequestBody Employee employee) {  // get the data from current id  Optional<Employee> employeeCheck= employeeRepository.findById(employee.getEmp\_no());    // validate only filled data will be change  **if** (employee.getFirst\_name()==**null**) {  employee.setFirst\_name(employeeCheck.get().getFirst\_name());  }  **if** (employee.getLast\_name()==**null**) {  employee.setLast\_name(employeeCheck.get().getLast\_name());  }  **if** (employee.getBirth\_date()==**null**) {  employee.setBirth\_date(employeeCheck.get().getBirth\_date());  }  **if** (employee.getDept\_no()==**null**) {  employee.setDept\_no(employeeCheck.get().getDept\_no());  }  **if** (employee.getGender()==**null**) {  employee.setGender(employeeCheck.get().getGender());  }  // save the data into database  employeeRepository.save(employee);  **return** employee;  } |

deleteEmployee

|  |
| --- |
| @Override  @DeleteMapping("/Employee/{empNo}")  **public** List<Employee> deleteEmployee(@PathVariable String empNo) {  // **TODO** Auto-generated method stub  // get the data from current empNo  Optional<Employee> employees = employeeRepository.findById(empNo);  // if not found throw exception  **if** (!employees.isPresent()) {  **throw** **new** EmployeeNotFoundException("id-" + empNo);  }    // delete the data by empNo  employeeRepository.deleteById(empNo);    // get the list employee for the return  List<Employee> employeeList = employeeRepository.findAll();  **return** employeeList;  } |

retrieveAllEmployee

|  |
| --- |
| @Override  @GetMapping("/Employee")  **public** List<Employee> retrieveAllEmployee() {  // **TODO** Auto-generated method stub  // get all the employee  **return** employeeRepository.findAll();  } |

retrieveEmployeeId

|  |
| --- |
| @Override  @GetMapping("/Employee/{empNo}")  **public** Employee retrieveEmployeeId(@PathVariable String empNo) {  // **TODO** Auto-generated method stub  // get the employee detail by empNo  Optional<Employee> employees = employeeRepository.findById(empNo);  // if not found, throw exception  **if** (!employees.isPresent()) {  **throw** **new** EmployeeNotFoundException("id-" + empNo);  }    // return the data from above  **return** employees.get();  } |

downloadAllEmployee

|  |
| --- |
| @Override  @GetMapping("/Employee/download")  **public** **void** downloadAllEmployee(HttpServletResponse response) **throws** IOException {  // **TODO** Auto-generated method stub  // set the content type (text : postman, csv : browser)  response.setContentType("text/csv");    // set the data for header  DateFormat dateFormatter = **new** SimpleDateFormat("yyyy-MM-dd\_HH-mm-ss");  String currentDateTime = dateFormatter.format(**new** Date());    // set configuration csv  String headerKey = "Content-Disposition";  String headerValue = "attachment; filename=users\_" + currentDateTime + ".csv";  response.setHeader(headerKey, headerValue);    // get all the data from employee table  List<Employee> employees = employeeRepository.findAll();    // set up the csv body configuration  ICsvBeanWriter csvWriter = **new** CsvBeanWriter(response.getWriter(), CsvPreference.***STANDARD\_PREFERENCE***);  String[] csvHeader = {"Employee ID", "First Name", "Last Name", "Birth Date", "Dept No"};  String[] nameMapping = {"emp\_no", "first\_name", "last\_name", "birth\_date", "dept\_no"};    csvWriter.writeHeader(csvHeader);    // type up csv  **for** (Employee employee : employees) {  csvWriter.write(employee, nameMapping);  }    csvWriter.close();  } |