Date : 10/02/2021
Spring Boot 9AM
Mr. RAGHU

Spring Data JPA: findBy

=> findBy is a abstract method that derives a query at runtime
based on details in method name.

=> findBy supports both SQL(MySQL/Oracle) and NoSQL(MongoDB)
Databases.
=> findBy best suitable for simple queries.
=> findBy supports only SELECT operations.
=> No Manual Query required.

=> @Query works for SQL DBs only, supports both SELECT and Non-SELECT
operations. Custom/manual query must be given. Supports Complex
Queries.

=> findBy method must be written inside Repository Interface.
By following below syntax

```
ReturnType   findBy<VariableNameConditions>(<DataTypes> <params>);
```

Simple one:

```
ReturnType   findBy<VariableName>(<params>);
```

--Examples---------

```
class Employee {
  int empId;
  String empName;
  double empSal;
}
```

1.

```
List<Employee> findByempName(String en);  //valid
List<Employee> findByEmpName(String empName);  //valid + naming rule
```

Internally converted to:
SQL: select * from Employee where empName=en;

Hint: findBy => SELECT * FROM <TABLE>
variable => where variable=param

2.

```
List<Employee> findByEmpSal(double empSal);
```

Generated SQL:

```
 select * from Employee where empSal=empSal;
```

*) Here != in java , <> in database (not equals)
Ex: x>3 and x<3
can be written as
x<>3 (x can be gt 3 and lt 3 but not 3)

Ex:
SELECT * FROM EMPTAB WHERE EID>=? or ename IS Not Null and empSal in (__,__,__)

```
List<Employee>
findByEmpIdGreaterThanEqualOrEmpNameIsNotNullAndEmpSalIn(int
eid,List<Double> sals);
```

Above method is very lengthy (not a standard even) use @Query.

==========full code==========

1. Model

```
package in.nareshit.raghu.model;

import javax.persistence.Entity;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
public class Employee {
	@Id
	private Integer empId;
	private String empName;
	private Double empSal;
	private String empDept;
}
```

2. Repository

```
package in.nareshit.raghu.repo;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import in.nareshit.raghu.model.Employee;

public interface EmployeeRepository
	extends JpaRepository<Employee, Integer> {

	//SELECT * FROM EMPLOYEE WHERE EMP_DEPT=ed
	//List<Employee> findByempDept(String ed); //valid
	List<Employee> findByEmpDept(String empDept); //valid + naming
rules

	/***
	 *  No Condition Symbol       =
	 *  LessThan                  <
	 *  LessThanEqual             <=
	 *  GreaterThan               >
	 *  GreaterThanEqual          >=
```

```
         *  Between                    x,y
         *
         */

        //SELECT * FROM EMPLOYEE WHERE EMPSAL<=empSal
        List<Employee> findByempSalLessThanEqual(double empSal);

        //SELECT * FROM EMPLOYEE WHERE EMPSAL>empSal
        List<Employee> findByempSalGreaterThan(double empSal);

        //SELECT * FROM EMPLOYEE WHERE EMPSAL between empSal1 and
empSal2
        List<Employee> findByEmpSalBetween(double empSal1,double
empSal2);

        /**
         * Not    !=
         *
         * In     IN operator
         *
         * NotIn  NOT IN Operator
         */
        //SELECT * FROM EMPLOYEE WHERE EMPDEPT != ?
        List<Employee> findByEmpDeptNot(String empDept);

        //SELECT * FROM EMPLOYEE WHERE EMPDID IN (_,_,_,_,..)
        List<Employee> findByEmpIdIn(List<Integer> ids);

        //SELECT * FROM EMPLOYEE WHERE EMPDID NOT IN (_,_,_,_,..)
        List<Employee> findByEmpIdNotIn(List<Integer> ids);

         /**
         *  And
         *
         * Or
         */
        //SELECT * FROM EMPLOYEE WHERE EMPSAL>=? and EmpDept!=?
        List<Employee>
findByEmpSalGreaterThanEqualAndEmpDeptNot(Double empSal,String
empDept);

        //SELECT * FROM EMPLOYEE WHERE EMPSAL>=? or EmpDept!=?
        List<Employee> findByEmpSalGreaterThanEqualOrEmpDeptNot(Double
empSal,String empDept);

        /***
         *
         * Is Null , Is Not Null
        */
        //SELECT * FROM EMPLOYEE WHERE EMPNAME IS NULL
        List<Employee> findByEmpNameIsNull();
        //SELECT * FROM EMPLOYEE WHERE EMPNAME IS NOT NULL
        List<Employee> findByEmpNameIsNotNull();

        /**
         *  Like, NotLike
         * (StartingWith , EndingWith, Containing)
```

```
         *
         */
        //SELECT * FROM EMPLOYEE WHERE EMPNAME like 'exp'
        List<Employee> findByEmpNameLike(String expression);

        //SELECT * FROM EMPLOYEE WHERE EMPNAME like 'exp'
        List<Employee> findByEmpNameNotLike(String expression);

        //SELECT * FROM EMPLOYEE WHERE EMPNAME like 'exp%'
        List<Employee> findByEmpNameStartingWith(String expression);

        //SELECT * FROM EMPLOYEE WHERE EMPNAME like '%exp'
        List<Employee> findByEmpNameEndingWith(String expression);

        //SELECT * FROM EMPLOYEE WHERE EMPNAME like '%exp%'
        List<Employee> findByEmpNameContaining(String expression);
}
```

3. Data Insert runner

```
package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;

import in.nareshit.raghu.model.Employee;
import in.nareshit.raghu.repo.EmployeeRepository;

//@Component
public class DataInsertRunner implements CommandLineRunner {
        @Autowired
        private EmployeeRepository repo;

        public void run(String... args) throws Exception {
                repo.save(new Employee(10, "SAM", 500.0, "DEV"));
                repo.save(new Employee(11, "RAM", 600.0, "DEV"));
                repo.save(new Employee(12, "SYED", 450.0, "QA"));
                repo.save(new Employee(13, "VINAY", 280.0, "QA"));
                repo.save(new Employee(14, "JAI", 120.0, "DEV"));
                repo.save(new Employee(15, "SUN", 590.0, "BA"));


        }

}
```

4. test runner

```
package in.nareshit.raghu.runner;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import in.nareshit.raghu.repo.EmployeeRepository;

@Component
public class TestFindByRunner implements CommandLineRunner {
        @Autowired
```

```
        private EmployeeRepository repo;

        public void run(String... args) throws Exception {
                //List<Employee> emps = repo.findByEmpDept("DEV");
                //emps.forEach(System.out::println);


//repo.findByempSalLessThanEqual(300.0).forEach(System.out::println);

//repo.findByempSalGreaterThan(300.0).forEach(System.out::println);

                //repo.findByEmpSalBetween(300.0,
500.0).forEach(System.out::println);


//repo.findByEmpDeptNot("DEV").forEach(System.out::println);


//repo.findByEmpIdIn(Arrays.asList(10,12,14,16,18,20,22)).forEach(Syst
em.out::println);


//repo.findByEmpIdNotIn(Arrays.asList(12,14,18,20,22)).forEach(System.
out::println);


//repo.findByEmpSalGreaterThanEqualAndEmpDeptNot(120.0,
"QA").forEach(System.out::println);
                //repo.findByEmpSalGreaterThanEqualOrEmpDeptNot(120.0,
"QA").forEach(System.out::println);



//repo.findByEmpNameIsNull().forEach(System.out::println);

//repo.findByEmpNameIsNotNull().forEach(System.out::println);



repo.findByEmpNameNotLike("S%").forEach(System.out::println);

//repo.findByEmpNameLike("S%").forEach(System.out::println);

//repo.findByEmpNameStartingWith("S").forEach(System.out::println);


//repo.findByEmpNameLike("%M").forEach(System.out::println);

//repo.findByEmpNameEndingWith("M").forEach(System.out::println);


//repo.findByEmpNameLike("%A%").forEach(System.out::println);

//repo.findByEmpNameContaining("A").forEach(System.out::println);
```

```
        }

}
```

5. application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot9am
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```