

# Spring Boot JPA - Custom Query

Some time case arises, where we need a custom query to fulfil one test case. We can use `@Query` annotation to specify a query within a repository. Following is an example. In this example, we are using JPQL, Java Persistence Query Language.

We've added name query custom methods in Repository in JPA Named Query chapter. Now let's add another method using `@Query` and test it.

## Repository - EmployeeRepository.java

Add a method to get list of employees order by their names.

```
package com.tutorialspoint.repository;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import com.tutorialspoint.entity.Employee;

@Repository
public interface EmployeeRepository extends CrudRepository<Employee, Integer> {
    public List<Employee> findByName(String name);
    public List<Employee> findByAge(int age);
    public Employee findByEmail(String email);

    @Query(value = "SELECT e FROM Employee e ORDER BY name")
    public List<Employee> findAllSortedByName();
}
```

Let's test the methods added by adding their test cases in test file. Last two methods of below file tests the custom query method added.

Following is the complete code of EmployeeRepositoryTest.

```
package com.tutorialspoint.repository;

import static org.junit.jupiter.api.Assertions.assertEquals;
import java.util.ArrayList;
import java.util.List;
import javax.transaction.Transactional;
import org.junit.jupiter.api.Test;
```

```
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import com.tutorialspoint.entity.Employee;
import com.tutorialspoint.sprintbooth2.SprintBootH2Application;

@ExtendWith(SpringExtension.class)
@Transactional
@SpringBootTest(classes = SprintBootH2Application.class)
public class EmployeeRepositoryTest {
    @Autowired
    private EmployeeRepository employeeRepository;
    @Test
    public void testFindById() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        Employee result = employeeRepository.findById(employee.getId()).get();
        assertEquals(employee.getId(), result.getId());
    }
    @Test
    public void testFindAll() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        List<Employee> result = new ArrayList<>();
        employeeRepository.findAll().forEach(e -> result.add(e));
        assertEquals(result.size(), 1);
    }
    @Test
    public void testSave() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        Employee found = employeeRepository.findById(employee.getId()).get();
        assertEquals(employee.getId(), found.getId());
    }
    @Test
    public void testDeleteById() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        employeeRepository.deleteById(employee.getId());
        List<Employee> result = new ArrayList<>();
        employeeRepository.findAll().forEach(e -> result.add(e));
        assertEquals(result.size(), 0);
    }
    private Employee getEmployee() {
        Employee employee = new Employee();
    }
}
```

```
        employee.setId(1);
        employee.setName("Mahesh");
        employee.setAge(30);
        employee.setEmail("mahesh@test.com");
        return employee;
    }

    @Test
    public void testFindByName() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        List<Employee> result = new ArrayList<>();
        employeeRepository.findByName(employee.getName()).forEach(e -> result.add(e));
        assertEquals(result.size(), 1);
    }

    @Test
    public void testFindByAge() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        List<Employee> result = new ArrayList<>();
        employeeRepository.findByAge(employee.getAge()).forEach(e -> result.add(e));
        assertEquals(result.size(), 1);
    }

    @Test
    public void testFindByEmail() {
        Employee employee = getEmployee();
        employeeRepository.save(employee);
        Employee result = employeeRepository.findByEmail(employee.getEmail());
        assertNotNull(result);
    }

    @Test
    public void testFindAllSortedByName() {
        Employee employee = getEmployee();
        Employee employee1 = new Employee();
        employee1.setId(2);
        employee1.setName("Aarav");
        employee1.setAge(20);
        employee1.setEmail("aarav@test.com");
        employeeRepository.save(employee);
        employeeRepository.save(employee1);
        List<Employee> result = employeeRepository.findAllSortedByName();
        assertEquals(employee1.getName(), result.get(0).getName());
    }
}
```

## Run the test cases

Right Click on the file in eclipse and select **Run a JUnit Test** and verify the result.

