



Spring Boot MVC JPA H2 Query Methods

Agenda

1

Spring Boot MVC JPA H2 Query Methods

Objectives

At the end of this module, you will be able to:

- Understand, What is query methods in spring boot.

Spring Boot MVC JPA H2 Query Methods



Crud Repository Interface

- The CRUD methods from “CrudRepository” interface are.
 - save(Entity entity)
 - count()
 - delete(Entity entity)
 - deleteAll()
 - deleteById(Integer id)
 - findAll()
 - findById(Integer id)

Scenario - 1



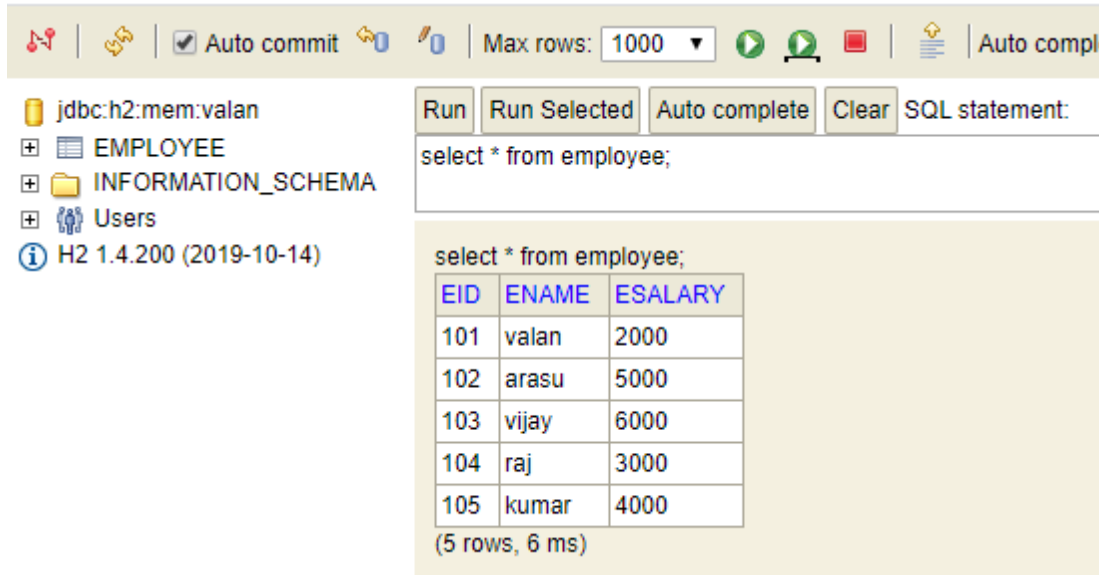
Scenario - 1

- Let us consider our model class is like below.

```
package com.wipro.pack.model;
import javax.persistence.Entity;
import javax.persistence.Id;
@Entity
public class Employee {
    @Id
    private int eid;
    private String ename;
    private int esalary;
    //Getters & Setters
    @Override
    public String toString() {
        return "Employee [eid=" + eid + ", ename=" + ename + ", esalary=" + esalary + "];"
    }
}
```

Scenario - 1

- Also let us load some records in H2 Employee table by using "data.sql" file. For example like below.



The screenshot displays the H2 Database GUI interface. On the left, the database structure is shown with 'jdbc:h2:mem:valan' as the connection, and tables 'EMPLOYEE' and 'INFORMATION_SCHEMA' are visible. The main area shows the execution of the SQL statement 'select * from employee;'. The results are displayed in a table with 5 rows and 3 columns: EID, ENAME, and ESALARY. The execution took 6 ms.

Max rows: 1000

Run Run Selected Auto complete Clear SQL statement:

```
select * from employee;
```

EID	ENAME	ESALARY
101	valan	2000
102	arasu	5000
103	vijay	6000
104	raj	3000
105	kumar	4000

(5 rows, 6 ms)

Scenario - 1

- In this case, If we want to find or delete any employee records by using "eid" then I can use "CrudRepository" interface methods findById() and deleteById().
- Suppose, If we want to find or delete any employee records by using “ename” or “esalary” then "CrudRepository" interface methods will not help. Because there is no such a method.
- Now will see the solution for this scenario in the next slide.

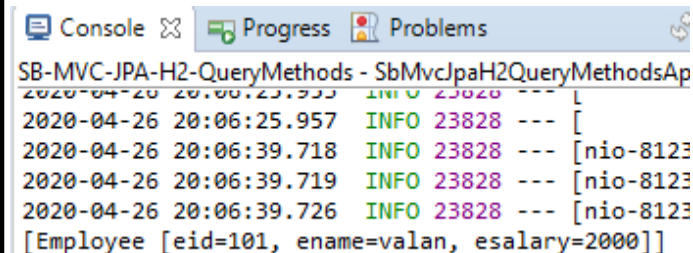
Scenario - 1

- Below is the solution.

```
package com.wipro.pack.dao;
import java.util.List;
import org.springframework.data.repository.CrudRepository;
import com.wipro.pack.model.Employee;
public interface EmployeeDao extends CrudRepository<Employee, Integer>{
    List<Employee> findByEname(String name);
}
```

- Just to check the result, will have a controller like below.

```
@Controller
public class EmployeeController {
    @Autowired
    EmployeeDao dao;
    @RequestMapping("/home")
    public String home() {
        System.out.println(dao.findByEname("valan"));
        return "";
    }
}
```



The screenshot shows the 'Console' tab of an IDE. The title bar reads 'SB-MVC-JPA-H2-QueryMethods - SbMvcJpaH2QueryMethodsAp'. The log output consists of several lines of timestamps, log levels, and IDs, followed by a JSON array representing an employee object.

```
2020-04-26 20:06:25.957 INFO 23828 --- [
2020-04-26 20:06:39.718 INFO 23828 --- [nio-8123]
2020-04-26 20:06:39.719 INFO 23828 --- [nio-8123]
2020-04-26 20:06:39.726 INFO 23828 --- [nio-8123]
[Employee [eid=101, ename=valan, esalary=2000]]
```

Scenario - 1

- Here the rule is that the method must begin with `findBy` or `getBy` and then the property name.

```
List<Employee> findByEname(String name);  
  
List<Employee> getByEname(String name);
```

- Same way, you can have your own `deleteBy` or `removeBy` method.

```
void deleteByEname(String name);  
  
void deleteByEsalary(int salary);  
  
void removeByEsalary(int salary);
```

Scenario - 2



Scenario - 2

- In this case, If we want to find or delete any employee records by using some conditions.
- Like salary greater than or less than.
- Will see the solution for this scenario in the next slide.

Scenario - 2

- Below is the solution for our “EmployeeDao” interface.

```
List<Employee> findByEsalaryGreaterThan(int salary);  
  
List<Employee> findByEsalaryLessThan(int salary);  
  
List<Employee> deleteByEsalaryGreaterThan(int salary);  
  
List<Employee> deleteByEsalaryLessThan(int salary);
```

Scenario - 3



Scenario - 3

- If we want to write our own query. Then we can use @Query to write JPQL.

```
@Query("from employee where ename=?1 order by esalary")  
List<Employee> findByEnameSorted(String name);
```


Summary

In this session, you have learned about:

- What is query methods in spring boot.



Thank you