***Introduction to more queries with JPA***

* Getting records by particular column
* AND, OR, IN Queries
* Pagination
* Sorting

***Get records by column using JPA***

***Note:*** inside the controller

**@GetMapping(“getByFirstName/{firstName}”)**

Public List<StudentResponse> getByFirstName (@PathVariable String firstName){

List<Student> studentList = studentService.getByFirstName(firstName);

}

Note: on the service lair we have the method, and on the repository too

**@Repository**

Public interface StudentRepository extends JpaRepository<Student, Long>{

List<Student> findByFirstName(String firstName);

}

***Checking SQL queries generated by JPA***

Note: on **application.properties**

**Spring.datasource.url=jdbc:mysql://localhost:3306/spring**

**Spring.datasource.username=root**

**Spring.datasource.password=root**

**Spring.jpa.database=mysql**

**Spring.jpa.show-sql=true**

**Note:** *checkout when you run the app spring generates the query*

***And query using JPA***

Pass multiple variables in the URL on the controller

@GetMapping(“getByFirstNameAndLastName/{firstName}/{lastName}”)

Public StudentResponse getByFirstNameAndLastName (@PathVariable String firstName, @PathVariable String lastName){

new StudentResponse(studentService.getByFirstNameAndLastName(firstName, lastName));

}

@Repository

Public interface …. Extends JpaRepository<… , …>{

Declare methods

Student findByLastNameAndFirstName (String lastname, String firstname)

}

Note: in the service

Public Student getByFirstNameAndLastName(String firstname, String lastname){

Return studentRepository.findByLastNameAndFirstName(lastname, firstname);

}

**OR Query Using JPA**

**@Repository**

Public interface StudentRepository extends JpaRepository<Student, Long>{

List<Student> findByFirstNameOrLastName (String firstName, String lastName)

}

**@Service**

Public List<Student> getByFirstNameOrLastName (String firstName, String lastName){

Return studentRepository.findByFirstNameOrLastName(firstName, lastName);

}

**@RestController**

@GetMapping(“getByFirstNameOrLastName/{firstName}/{lastName}”)

Public List<StudentResponse> getByFirstNameOrLastName (@PathVariable String firstName, @PathVariable String lastName){

List<Student> studentList = studentService.getByFirstNameOrLastName(firstName, lastName);

List<StudentResponse> studentResponseList = new ArrayList<StudentResponse>();

studentList.stream().forEach(student -> {studentResponseList.add(new StudentResponse(student));

});

Return studentResponseList;

}

***In query using JPA***

Create a class with the setters and getters methods

@Getter

@Setter

Public class InQueryRequest{

Private List<String> firstNames;

}

@GetMapping(“getByFirstNameIn”)

Public List<StudentResponse> getByFirstNameIn(@RequestBody InQueryRequest inQueryRequest){

List<Student> studentList = studentService.getByFirstNameIn (inQueryRequest);

List<StudentResponse> studentResponseList = new ArrayList<StudentResponse>();

studentList.stream().forEach(student -> {

studentResponseList.add(new studentResponse(student));

});

return studentResponseList;

}

@Repository

…. (in the interface extended)

List<Student> findByFirstNameIn(List<String> firstNames);

@Service

….

Public List <Student> getByFirstNameIn (InQueryRequest inQueryRequest){

Return studentRepository.findByFirstNameIn(inQueryRequest.getFirstNames());

}

***What is pagination?***

Page number, skip and limit (page size)

***Pagination using JPA***

@RestController

@GetMapping (“getAllWithPagination”)

Public List<StudentResponse> getAllStudentsWithPagination (@RequestParam int pageNo, @RequestParam int pageSize){

List<Student> studentList = studentService.getAllStudentsWithPagination(pageNo, pageSize);

List<StudentResponse> studentResponseList = new ArrayList<StudentResponse>();

studentList.stream().forEach(student -> {studentResponseList.add(new StudentResponse(student));

});

Return studentResponseList;

}

@Service

Public List<Student> getAllStudentsWithPagination (int pageNo, int pageSize){

Pageable pageable = PageRequest.of(pageNo-1, pageSize);

Return studentRepository.findAll(pageable).getContent();

}

***Sorting using JPA***

@RestController

@GetMapping(“getAllWithSorting”)

Public List<Student> studentList = studentService.getAllStudentsWithSorting();

Public List<StudentResponse> studentResponseList = new ArrayList<StudentResponse>();

studentList.stream().forEach(student -> {

studentResposneList.add(new StudentResponse (student));

});

Return studentResponseList;

}

@Service

Public List<Student> getAllStudentsWithSorting() {

Sort sort = Sort.by(Sort.Direction.ASC, “firstName”);

Return studentRepository.findAll(sort);

}

***Note:*** in the service in the place of firstName you can add multiple fields with a comma and it will sort everything

***@Transient Annotation***

It tells spring boot that does not represent a column or any annotation you are using

Note: you can use it in the constructor and has to be in the @RestController or any field and you can concatenate with the rest of the methods.

@RestController

…..

@Column

Private Field1

@Column

Private Field2

@Transient

Private Field3

Public Constructor{

This. “…” = “….”. method();

This. “…” = “….”. method();

This. “transient” = “…”.method1() + “ ” + “…”. Method2();

}

***Like Query Using JPA***

@RestController

….

@GetMapping(“like/{firstName}”)

Public List<StudentResponse> like (@PathVariable String firstName){

List<Student> studentList = studentService.like(firstName);

List<StudentResponse> studentResponseList = new ArrayList<StudentResponse>();

studentList.stream().forEach(student -> {

studentResponseList.add(new StudentResponse(student));

});

Return studentResponseList;

}

@Service

Public List<Student> like(String firstName){

Return studentRepository.findByFirstNameContains(firstName);

}

@Repository

….

List<Student> findByFirstNameContains(String firstName);

***Starts with query using Jpa***

@Repository

List<Student> findByFirstNameStartsWith(String firstName);

@Service

Public List<Student> startsWith(String firstName){

Return studentRepository.findByFirstNameStartsWith(firstName);

}

@RestController

@GetMapping(“startsWith/{firstName}”)

Public List<StudentResponse> startsWith(@PathVariable String firstName){

List<Student> studentList = studentService.startsWith(firstName);

List<StudentResponse> studentResponseList = new ArrayList<StudentResponse>();

studentList.stream().forEach(student -> {

studentResponseList.add(new StudentResponse(student));

});

Return studentResponseList;

}