Spring Data JPA

프로젝트 : B21cSpringDataJPA2

준비사항

- 의존설정 : JDBC API, Lombok, Oracle Driver, Spring Web, Spring Data JPA
- JSP 사용을 위한 설정을 한다.
- Refresh Gradle Project 를 눌러 적용한다.

JpaRepository

- Spring Data JPA에서 제공하는 인터페이스
- 기본적인 CRUD (Create, Read, Update, Delete) 기능을 자동으로 제공
- 메서드 명명 규칙에 따라 다양한 쿼리를 확장할 수 있음
- 형식 : JpaRepository<엔티티, PK타입>

1. 기본 키워드

- findBy : 조회 작업을 수행. 일반적으로 여러 결과를 반환하는 메서드를 정의할 때 사용.
- readBy : 조회 작업을 수행하며, 읽기 작업이라는 것을 강조하고 싶을 때 사용.
- queryBy : 쿼리 작업을 강조하여 데이터를 조회할 때 사용. findBy와 동일하게 동작.
- getBy : 단일 결과를 기대하는 조회 쿼리. 그러나 여러 결과를 반환하는 List로도 사용할 수 있음.
- countBy : 특정 조건에 맞는 엔티티의 개수를 반환.
- deleteBy : 조건에 맞는 데이터를 삭제.

2. 쿼리 확장을 위한 메서드 명명 규칙

- Spring Data JPA는 메서드 이름을 분석하여 자동으로 쿼리를 생성
- 기본 키워드를 시작으로 뒤에 조건을 명시하여 확장
    - findBy : 조회 쿼리를 생성하는 기본 키워드
    - By : 조건을 지정할 때 사용되는 구분자
    - 컬럼명 : 엔티티의 필드명을 사용하여 조건을 설정.
    - 연산자 : `And`, `Or`, `Like`, `LessThan`, `GreaterThan`, `IsNull`, `In`, `Between` 등 추가 가능

3. 메서드 예시

컬럼명은 name, email, userid 라고 가정했을때 작성할 수 있는 메서드 명

- name 컬럼을 기준으로 하는 메서드
    - `findByName(String name) : name 으로 조회`
    - `findByNameLike(String name) : name 으로 LIKE 조회`

    - findByNameStartingWith(String prefix) : name이 특정 문자열로 시작하는 경우 조회
    - findByNameEndingWith(String suffix) : name이 특정 문자열로 끝나는 경우 조회
    - findByNameContaining(String part) : name에 특정 문자열이 포함된 경우 조회
    - findByNameIgnoreCase(String name) : 대소문자를 구분하지 않고 name으로 조회
- email 컬럼을 기준으로 하는 메서드
    - findByEmailIsNull(): email 값이 NULL인 레코드 조회
    - findByEmailIsNotNull(): email 값이 NULL이 아닌 레코드 조회
- 두개 이상의 컬럼을 사용하는 메서드
    - findByNameAndEmail(String name, String email) : name과 email 값이 모두 일치하는 경우 조회
    - findByNameOrEmail(String name, String email) : name 또는 email 중 하나가 일치하는 경우 조회
    - findByUseridAndName(String userid, String name): userid와 name이 모두 일치하는 경우 조회.
    - findByUseridOrEmail(String userid, String email): userid 또는 email이 일치하는 경우 조회.
- 그 외 연산자 및 정렬
    - findByUseridIn(List<String> userids) : 주어진 리스트 중 하나라도 일치하는 경우 조회
    - findByEmailIsNull() 혹은 findByEmailIsNotNull() : email이 null 혹은 null 이 아닌 경우 조회
    - findByNameOrderByUseridAsc(String name): name이 일치하는 결과를 userid 기준으로 오름차순 정렬
    - findByNameOrderByUseridDesc(String name): name이 일치하는 결과를 userid 기준으로 내림차순 정렬
- 페이징 적용
    - findByName(String name, Pageable pageable): name이 일치하는 결과를 페이지 단위로 반환

프로퍼티 : application.properties

```
spring.application.name=B21cSpringDataJPA2

#포트설정
server.port=8586

#JSP 설정
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp

#오라클 접속 정보
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=musthave
spring.datasource.password=1234
spring.datasource.hikari.maximum-pool-size=5

#JPA설정 : persistence.xml과 동일한 내용임
spring.jpa.database-platform=org.hibernate.dialect.OracleDialect
spring.jpa.open-in-view=false
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
# none, create, create-drop, update, validate
spring.jpa.properties.hibernate.hbm2ddl.auto=none

#show sql data binding
logging.level.org.hibernate.SQL=debug
logging.level.org.hibernate.orm.jdbc.bind=trace
```

엔티티 : Member.java

```java
package com.edu.springboot.jpa;

import jakarta.persistence.Entity;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@Builder
@Entity(name="JPAMEMBER02")
public class Member {
    @Id
    @GeneratedValue
    private Long id;
    private String name;
    private String email;
}
```

레포지토리 : edu/spring/boot/jpa/MemberRepository.java

```java
package com.edu.springboot.jpa;

import java.util.List;

@Repository
public interface MemberRepository extends JpaRepository<Member, Long>
{
    // findBy 뒤에 컬럼명을 붙여주면 이를 이용한 검색이 가능하다.
    Optional<Member> findByName(String keyword);
    Optional<Member> findByEmail(String keyword);

    // 다음과 같이 다양한 확장이 가능하다.
    List<Member> findByNameLike(String keyword);
    List<Member> findByNameLikeOrderByNameDesc(String keyword);
    List<Member> findByNameLikeOrderByNameAscEmailDesc(String keyword);

    //정렬의 조건때문에 메서드명이 길어지게 되므로 Sort객체로 검색의 조건을 추가할 수 있다.
    List<Member> findByNameLike(String keyword, Sort sort);
}
```

서비스 : /B21bSpringDataJPA1/src/main/java/edu/spring/boot/jpa/MemberService.java

```java
package com.edu.springboot.jpa;

import java.util.List;

@Service
public class MemberService {

    @Autowired
    private MemberRepository memberRepository;

    public void insert() {
        Member member;

        member = Member.builder().name("이순신")
                .email("test1@test.com").build();
        memberRepository.save(member);

        member = Member.builder().name("강감찬")
                .email("test2@test.com").build();
        memberRepository.save(member);

```

이와 같은 패턴으로 10개정도 입력합니다.

을지문덕, 계백, 김유신, 연개소문, 양만춘, 김종서, 최영, 선덕여왕, 신사임당, 성삼문, 이방원 등

```java
    //전체조회
    public List<Member> selectAll() {
        return memberRepository.findAll();
    }
    //아이디로 검색
    public Optional<Member> selectId(Long search) {
        Optional<Member> member = memberRepository.findById(search);
        return member;
    }
    //이름으로 검색
    public Optional<Member> selectName(String search) {
        Optional<Member> member = memberRepository.findByName(search);
        return member;
    }
    //이메일로 검색
    public Optional<Member> selectEmail(String search) {
        Optional<Member> member = memberRepository.findByEmail(search);
        return member;
    }
```

```
67      //이름을 like로 검색
68      public List<Member> selectNameLike(String search) {
69          List<Member> member = memberRepository.findByNameLike(search);
70          return member;
71      }
72      //이름을 like로 검색 후 내림차순 정렬
73      public List<Member> selectNameLikeNameDesc(String search) {
74          List<Member> member = memberRepository
75                  .findByNameLikeOrderByNameDesc(search);
76          return member;
77      }
78      //Sort 빈을 통해 정렬
79      public List<Member> selectNameLike(String search, Sort sort) {
80          List<Member> member = memberRepository
81                  .findByNameLike(search, sort);
82          return member;
83      }
84  }
85
```

컨트롤러 : /B21bSpringDataJPA1/src/main/java/edu/spring/boot/MainController.java

```
 1  package com.edu.springboot;
 2
 3  import java.util.List;
15
16  @Controller
17  public class MainController {
18
19      @GetMapping("/")
20      public String main() {
21          return "main";
22      }
23
24      @Autowired
25      MemberService memberService;
26
27      //레코드 입력
28      @GetMapping("/insert.do")
29      public String insert(Member member, Model model) {
30          memberService.insert();
31          model.addAttribute("title", "Insert");
32          model.addAttribute("result", "레코드 입력 성공");
33          model.addAttribute("mode", 0);//0:메세지만, 1:단일행, 2:복수행
34          return "total_view";
35      }
```

```
36      //전체 조회
37      @GetMapping("/selectAll.do")
38      public String selectAll(Model model) {
39          List<Member> result = memberService.selectAll();
40          model.addAttribute("title", "Select All");
41          model.addAttribute("result", result);
42          model.addAttribute("mode", 2);//0:메세지만, 1:단일행, 2:복수행
43          return "total_view";
44      }
45      //아이디로 검색
46      @GetMapping("/selectById.do")
47      public String selectById(@RequestParam("id") Long search,
48              Model model) {
49          Optional<Member> result = memberService.selectId(search);
50          model.addAttribute("title", "Select By Id");
51          model.addAttribute("result", result.get());
52          model.addAttribute("mode", 1);//0:메세지만, 1:단일행, 2:복수행
53          return "total_view";
54      }
55      //이름으로 검색
56      @GetMapping("/selectByName.do")
57      public String selectByName(@RequestParam("name") String search,
58              Model model) {
59          Optional<Member> result = memberService.selectName(search);
60          model.addAttribute("title", "Select By Name");
61          model.addAttribute("result", result.get());
62          model.addAttribute("mode", 1);//0:메세지만, 1:단일행, 2:복수행
63          return "total_view";
64      }
65      //이메일로 검색
66      @GetMapping("/selectByEmail.do")
67      public String selectByEmail(@RequestParam("email") String search,
68              Model model) {
69          Optional<Member> result = memberService.selectEmail(search);
70          model.addAttribute("title", "Select By Email");
71          model.addAttribute("result", result.get());
72          model.addAttribute("mode", 1);//0:메세지만, 1:단일행, 2:복수행
73          return "total_view";
74      }
```

```
75      //이름으로 like 검색
76      @GetMapping("/selectByNameLike.do")
77      public String selectByNameLike(
78              @RequestParam("name") String search, Model model) {
79          String name = search + "%";
80          List<Member> result = memberService.selectNameLike(name);
81          model.addAttribute("title", "Select By Like Name");
82          model.addAttribute("result", result);
83          model.addAttribute("mode", 2);//0:메세지만, 1:단일행, 2:복수행
84          return "total_view";
85      }
```

```
86      //이름으로 like 검색 후 정렬
87      @GetMapping("/selectByNameLikeNameDesc.do")
88      public String selectByNameLikeNameDesc(
89              @RequestParam("name") String search, Model model) {
90          String name = search + "%";
91          List<Member> result = memberService
92                  .selectNameLikeNameDesc(name);
93          model.addAttribute("title", "Select By Like Name Desc");
94          model.addAttribute("result", result);
95          model.addAttribute("mode", 2);//0:메세지만, 1:단일행, 2:복수행
96          return "total_view";
97      }
98      //위와 동일지만 Sort 클래스로 정렬
99      @GetMapping("/selectByNameLikeOrder.do")
00      public String selectByNameLikeOrder(
01              @RequestParam("name") String search, Model model) {
02          String name = search + "%";
03
04          //정렬로 인해 메서드명이 길어지는것을 Sort를 통해 줄일 수 있음
05          Sort sort = Sort.by(Sort.Order.desc("name"));
06  //      Sort sort = Sort.by(Sort.Order.desc("name"),
07  //              Sort.Order.asc("email"));
08
09          List<Member> result = memberService
10                  .selectNameLike(name, sort);
11          model.addAttribute("title", "Select By "
12                  + "Like Name Desc(Sort사용)");
13          model.addAttribute("result", result);
14          model.addAttribute("mode", 2);//0:메세지만, 1:단일행, 2:복수행
15
16          return "total_view";
17      }
18  }
```

뷰 : /B21bSpringDataJPA1/src/main/webapp/WEB-INF/views/main.jsp

```jsp
<body>
    <h2>Spring Data JPA</h2>
    <ul>
        <li><a href=/insert.do>데이터 추가</a></li>
        <li><a href=/selectAll.do>전체 조회</a></li>
        <li><a href=/selectById.do?id=1>
                개별 조회 - byId</a></li>
        <li><a href=/selectByName.do?name=을지문덕>
                개별 조회 - byName</a></li>
        <li><a href=/selectByEmail.do?email=test7@test.com>
                개별 조회 - byEmail</a></li>
        <li><a href=/selectByNameLike.do?name=김>
                리스트 조회 - Name Like</a></li>
        <li><a href=/selectByNameLikeNameDesc.do?name=김>
                리스트 조회 - Name Like Name Desc</a></li>
        <li><a href=/selectByNameLikeOrder.do?name=김>
                리스트 조회 - Name Like Name Desc(Sort사용)</a></li>
    </ul>
</body>
</html>
```

뷰 : /B21cSpringDataJPA2/src/main/webapp/WEB-INF/views/total_view.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="jakarta.tags.core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h2>Spring boot 프로젝트</h2>
    <ul>
        <li><a href="/">루트</a></li>
    </ul>

    <h2>Spring Data JPA - ${ title }</h2>
    <c:if test="${ mode eq 0 }">
        <p>
            ${ result }
        </p>
    </c:if>
```

```
22      <c:if test="${ mode eq 1 }">
23          <p>
24              아이디 : ${result.id}<br>
25              이름 : ${result.name}<br>
26              이메일 : ${result.email}
27          </p>
28      </c:if>
29      <c:if test="${ mode eq 2 }">
30          <c:forEach items="${ result }" var="member">
31              <p>
32                  아이디 : ${member.id}<br>
33                  이름 : ${member.name}<br>
34                  이메일 : ${member.email}
35              </p>
36          </c:forEach>
37      </c:if>
38  </body>
39  </html>
```