JPA 만나기

에이콘 테이블에 CRUD 하기

```
CREATE TABLE acorntbl_jpa (
    id VARCHAR(10) PRIMARY KEY,
    pw VARCHAR(10),
    name VARCHAR(10)
);
```

에이콘회원정보 CRUD하기

■ Spring Data Jpa 가 제공하는 JpaRepository 상속받기
  –기본CRUD 기능제공
  –페이징 기능 제공

```
public interface MemberRepository extends JpaRepository<Member, String> {
}
```

■ SQL 작성없이 다음의 기능을 사용할 수 있다 (물론 JPA가 대신 작성해 준다)

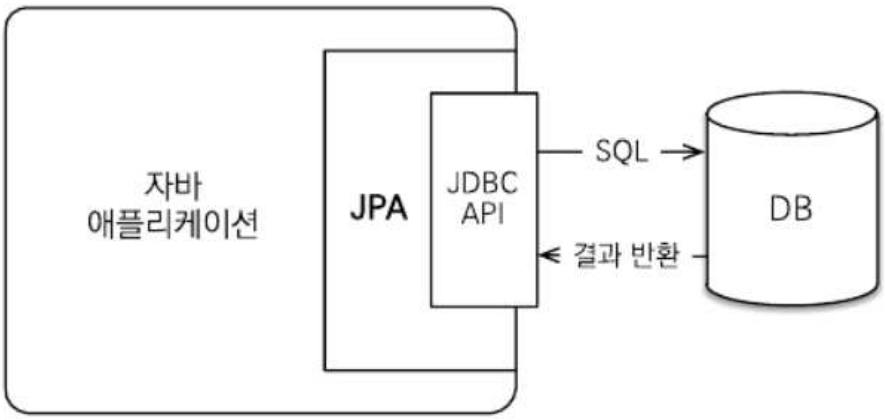
전체조회	memberRepository.findAll()
한 개 조회	memberRepository.findById(“testid”)
등록	memberRepository.save(member)
변경	memberRepository.save(member)
삭제	memberRepository.deleteById(“testid”)
페이징	int page=0; //1페이지 int size =5 ; Pageable pageable = PageRequest.of(page, size) Page<Member> page =memberRepository.findAll(Pageable pageable) System.out.println(page.getContent()); System.out.println(page.hasPrevious()); System.out.println(page.hasNext()); System.out.println(page.getTotalPages());

```
--crud   => create(insert ) , read( select ), u(update) , d(delete)
select *  from  acorntbl_jpa;


select *  from acorntbl_jpa ;
insert into acorntbl_jpa values('usung0518', '1234', '권지언');
insert into acorntbl_jpa values('boseong00', '1234', '김보성');
insert into acorntbl_jpa values('Jys123', '9876', ' 정연수');
insert into acorntbl_jpa values('sumni', '1234', '이수민');
insert into acorntbl_jpa values('dongwoo12', '1234', '이동우');
insert into acorntbl_jpa values('gusrl123', 'dks123', '윤현기');
insert into acorntbl_jpa values('sulivun_03', '1234', '박시우');
insert into acorntbl_jpa values('yerin0373', '1234', '박예린');
insert into acorntbl_jpa values('yzei', '1234', '황예지');
insert into acorntbl_jpa values('jitae', '1214', '최지태');
insert into acorntbl_jpa values('sookyung', '1004', '박수경');
insert into acorntbl_jpa values('yunseok', '123', '오윤석');
insert into acorntbl_jpa values('LHJ0319', '1234', ' 이정호');
insert into acorntbl_jpa values('gudxor8251', '1234', '임형택');
insert into acorntbl_jpa values('che', '1234', '최하은');
insert into acorntbl_jpa values('umin', '1234', '김유민');
insert into acorntbl_jpa values('gill', '0000', '김민환');
commit;

select * from acorntbl;
```

JPA란 (JAVA Persistence API) : 자바ORM표준



객체를 관계형 데이터베이스에 저장하려면 많은 시간과 코드를 소비해야한다
개발자는 객체와 데이터베이스 사이에서 매핑코드와 SQL을 작성해야 한다
이런 문제를 해결하기 위해서 JPA 기술을 제공

JPA는 SQL 작성없이 객체를 데이터베이스에 직접 저장할 수 있게 도와주고
객체와 관계형 데이터베이스의 차이도 해결해 준다

ORM(Object - Relational Mapping)이란?
객체와 관계형 데이터베이스를 매핑한다는 뜻
ORM 프레임워크는 객체와 테이블을 매핑해서 패러다임의 불일치를 개발자 대신해서 해결 해 준다

=〉 객체와 관계형 데이터베이스의 테이블 간의 데이터를 자동으로 변화해주는 기술 !!

■ 회원등록하기

1. JDBC
2. Mybatis Mapper
3. JPA - Hibernate

```
repository.save(member);
```

◆ JDBC

```
public class UserRegistration {

    // JDBC URL, 사용자명, 비밀번호
    private  String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    private  String DB_USER = "username";
    private  String DB_PASSWORD = "password";

    // 사용자 등록 메서드
    public void registerUser(String username, String password, String email) {
        // SQL 쿼리
        String sql = "INSERT INTO users (username, password, email) VALUES (?, ?, ?)";

        try (
            // 데이터베이스 연결
            Connection connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
            // PreparedStatement 생성
            PreparedStatement preparedStatement = connection.prepareStatement(sql);
        ) {
            // 값 설정
            preparedStatement.setString(1, username);
            preparedStatement.setString(2, password);
            preparedStatement.setString(3, email);

            // 쿼리 실행
            int rowsInserted = preparedStatement.executeUpdate();

            if (rowsInserted > 0) {
                System.out.println("회원가입이 성공적으로 완료되었습니다.");
            }
        } catch (SQLException e) {
            System.out.println("회원가입 중 오류 발생: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {

        UserRegistration registration = new UserRegistration();
        registration.registerUser("길동", "1234"  ,hong@example.com");
    }
}
```

◆ Mybatis

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="UserMapper">
    <insert id="insertUser" parameterType="map">
        INSERT INTO users (username, password, email)
        VALUES (#{username}, #{password}, #{email})
    </insert>
</mapper>
```

```
@Service
public class UserService {

    @Autowired
    private SqlSession sqlSession;

    public void registerUser(String username, String password, String email) {
        Map<String, Object> paramMap = new HashMap<>();
        paramMap.put("username", username);
        paramMap.put("password", password); // 비밀번호 암호화 권장
        paramMap.put("email", email);

        // MyBatis 매퍼 호출
        sqlSession.insert("UserMapper.insertUser", paramMap);
        System.out.println("회원가입이 완료되었습니다.");
    }
}
```

◆ Spring Data JPA

```
        User user = new User();
        user.setUsername(username);
        user.setPassword(password);
        user.setEmail(email);
        userRepository.save(user);
```

핵심개념

-Entity : 데이터베이스 테이블에 매핑되는 자바 클래스

-Entity Manager : 엔티티의 라이프 사이클을 관리한다

-JPQL (java Persistence Query Language)

: JPA에서 제공하는 객체지향 쿼리 언어로 , SQL과 비슷하지만
데이터베이스의 테이블이 아닌 엔티티 객체를 대상으로 동작

ORM 동작방식

1.클래스와 테이블 매핑

```
@Entity
class Member{
    @Id
   String id ;
   String pw;
   String name;
}
```

2. 객체와 데이터 변환

-객체의 정보를 데이터베이스 테이블의 레코드로 변환

-데이터베이스에서 가져온 결과를 객체로 변환하여 사용

3. 쿼리 추상화

- SQL쿼리를 직접 작성하는 대신, 객체 기반의 매서드를 호출해 CRUD(조회, 등록, 변경, 삭제) 함

◆ 에이콘 테이블 CRUD with JPA

```
@Slf4j
@Service
public class MemberService {


    @Autowired
    MemberRepository repository;

    //등록

    public Member register( MemberDTO member){

        Member memberEntity = Member.createMember(member);
        Member saveMember = repository.save(memberEntity);
        return saveMember;
    }


    //조회
    public List<Member>  getMemberList(){
        List<Member> list  = repository.findAll();
        return  list;
    }

    //변경  (변경감지)  엔티티를 변경하면 변경됨
    @Transactional
    public void  modifyMember( String id, String newpw){

        Optional<Member> optional = repository.findById(id);

        Member member =null;
        if(optional.isPresent()){
            member = optional.get();
            member.setPw(newpw);
          //  repository.save( member);   // 같은  키로  save  => update
        }


        log.info("변경: {}", newpw);

    }
```

```
public Member getMember(String id) {

   Member member = repository.findById( id).orElse( new Member());

   /*
    Optional<Member> optional = repository.findById(id);
    Member member = optional.orElse(new Member());  //
    */

return  member;

}


  public Page<Member> getMembersByName(String name, int page, int size) {
      Pageable pageable = PageRequest.of(page, size);
       return repository.findByNameContaining(name, pageable);
  }



public Page<Member> findAll(  int page, int size) {
    Pageable pageable = PageRequest.of(page, size);
    return repository.findAll(pageable);
}



public List<Member> getMembersByName(String name ) {
    //쿼리매서드 사용
    return repository.findByNameContaining(name);
}


///
public List<Member> getMembersByNameJPQL(String name ) {
    // @Query에서 사용한 이름과 비밀번호를 기준으로 조회
    // 직접쿼리작성하기
    return repository.findBy22222(name);
}


//회원 삭제하기
```

```
    public void  deleteMemberById(String id){
        repository.deleteById( id);
    }

}
```

```java
@Repository
public interface MemberRepository extends JpaRepository<Member, String> {

    //쿼리매서드 제공
    //SELECT * FROM acorntbl_jpa WHERE name LIKE '%입력값%';
    Page<Member> findByNameContaining(String name, Pageable pageable);
    List<Member> findByNameContaining(String name );


    //List<Member> findByName(String name);
    //JPQL 작성
    // sql
    // select * from acorntbl_jpa   where  name =?
    //SELECT * FROM acorntbl_jpa WHERE name LIKE CONCAT('%', ?, '%');
    //SELECT * FROM acorntbl_jpa WHERE name LIKE '%값%';

    //  작성한  sql  =>  jpql
   // @Query(value = "SELECT m  FROM Member  m   WHERE m.name = :name ")
    @Query("SELECT m FROM Member m WHERE m.name LIKE %:name%")
    List<Member> findBy22222(@Param("name") String name );


}
```