

# JPA 기초

## 두번째실습

설정파일 : resources/META-INF/persistence.xml

<class>myjpa4.Member4</class> ⇒ 이 부분을 myjpa2.Member2 로 수정 후 테스트

create 는 그대로 유지한다.

```
10 <persistence-unit name="MyJPA" transaction-type="RESOURCE_LOCAL">
11     <!-- 엔티티로 사용할 클래스의 풀 경로 -->
12     <class>myjpa4.Member4</class>
13
14     <!-- 명시적으로 나열된 클래스들만 엔티티로 인식 -->
15     <exclude-unlisted-classes>true</exclude-unlisted-classes>
16
```

/B21aJPA\_Basic/src/main/java/myjpa2/Member2.java

```
1 package myjpa2;
2
3 import jakarta.persistence.Access;
11
12 @Entity
13 @Table(name="JpaMember2")
14 public class Member2 {
15
16     @Id
17     @SequenceGenerator(
18         name = "mySequence01",
19         sequenceName = "JpaMember2_SEQ",
20         initialValue = 1,
21         allocationSize = 1
22     )
23     @GeneratedValue(generator = "mySequence01")
24     private Long id;
25
26     @Access(AccessType.FIELD)
27     private String username;
28
29     @Access(AccessType.PROPERTY)
30     private String password;
31
32     @Transient
33     private long timestamp1;
34     transient private long timestamp2;
35
```

```

36     public Member2() {}
37     public Member2(String username, String password) {
38         super();
39         this.username = username;
40         this.password = password;
41     }
42
43     public String getPassword() {
44         return password;
45     }
46     public void setPassword(String password) {
47         this.password = password;
48     }
49 }
50

```

[/B21aJPA\\_Basic/src/main/java/myjpa2/UseMember2.java](#)

```

1  package myjpa2;
2
3  import jakarta.persistence.EntityManager;
7
8  public class UseMember2 {
9
10     public static void main(String[] args) {
11
12         EntityManagerFactory emf =
13             Persistence.createEntityManagerFactory("MyJPA");
14         EntityManager em = emf.createEntityManager();
15         EntityTransaction transaction = em.getTransaction();
16
17         try {
18             transaction.begin();
19             Member2 member2 = new Member2("홍길동2", "1234");
20             em.persist(member2);
21             transaction.commit();
22         }
23         catch (Exception e) {
24             e.printStackTrace();
25             transaction.rollback();
26         }
27         finally {
28             em.close();
29         }
30
31         emf.close();
32     }
33 }

```

## 세번째실습

설정파일 : resources/META-INF/persistence.xml

<class>myjpa4.Member4</class> ⇒ 이 부분을 myjpa3.Member3 로 수정 후 테스트.

```
10 <persistence-unit name="MyJPA" transaction-type="RESOURCE_LOCAL">
11     <!-- 엔티티로 사용할 클래스의 풀 경로 -->
12     <class>myjpa4.Member4</class>
13
14     <!-- 명시적으로 나열된 클래스들만 엔티티로 인식 -->
15     <exclude-unlisted-classes>true</exclude-unlisted-classes>
16
```

/B21aJPA\_Basic/src/main/java/myjpa3/Member3.java

```
1 package myjpa3;
2
3 import java.time.LocalDate;
4
15 @Entity
16 @Table(name="JpaMember3")
17 public class Member3 {
18     @Id
19     private String email;
20
21     private String name;
22
23     @Column(name = "create_date")
24     private LocalDate createDate;
25
26     public Member3() {}
27     public Member3(String email, String name, LocalDate createDate) {
28         super();
29         this.email = email;
30         this.name = name;
31         this.createDate = createDate;
32     }
33     public String getEmail() {
34         return email;
35     }
36     public String getName() {
37         return name;
38     }
39     public LocalDate getCreateDate() {
40         return createDate;
41     }
42
43     public void changeName(String newName) {
44         this.name = newName;
45     }
46 }
```

/B21aJPA\_Basic/src/main/java/myjpa3/UseMember01\_insert.java

```
1 package myjpa3;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10 public class UseMember01_insert {
11
12     public static void main(String[] args) {
13
14         //영속성 인스턴스 생성
15         EntityManagerFactory emf =
16             Persistence.createEntityManagerFactory("MyJPA");
17         EntityManager em = emf.createEntityManager();
18         EntityTransaction transaction = em.getTransaction();
19
20         try {
21             transaction.begin();
22
23             //insert 처리
24             Member3 member3 =
25                 new Member3("hong@spring.com", "홍길동3",
26                     LocalDate.now());
27             em.persist(member3);
28
29             transaction.commit();
30         }
31         catch (Exception e) {
32             e.printStackTrace();
33             transaction.rollback();
34         }
35         finally {
36             em.close();
37         }
38
39         emf.close();
40     }
41 }
```

설정파일 : resources/META-INF/persistence.xml

insert 이므로 create 를 유지한다.

```
1 package myjpa3;
2
3 import jakarta.persistence.EntityManager;
4
5
6
7 public class UseMember02_select {
8
9     public static void main(String[] args) {
10
11         //영속성 인스턴스 생성
12         EntityManagerFactory emf =
13             Persistence.createEntityManagerFactory("MyJPA");
14         EntityManager em = emf.createEntityManager();
15         //select의 경우 트랜잭션은 생성하지 않는다.
16         // EntityManager transaction = em.getTransaction();
17
18         //조건에 맞는 레코드를 인출한다.
19         Member3 member3 = em.find(Member3.class, "hong@spring.com");
20         System.out.println("member3="+ member3);
21
22         if(member3 != null) {
23             System.out.println("이름:" + member3.getName());
24             System.out.println("날짜:" + member3.getCreateDate());
25         }
26         else {
27             System.out.println("존재하지 않습니다.");
28         }
29
30         emf.close();
31         em.close();
32     }
33 }
34
```

설정파일 : resources/META-INF/persistence.xml

select 이므로 none 으로 수정 후 테스트한다.

create 를 유지하면 테이블 삭제 후 새롭게 만들어 지므로 레코드가 없다고 출력된다.

```
1 package myjpa3;
2
3 import jakarta.persistence.EntityManager;
4
5
6
7
8 public class UseMember03_update {
9
10     public static void main(String[] args) {
11
12         //영속성 인스턴스 생성
13         EntityManagerFactory emf =
14             Persistence.createEntityManagerFactory("MyJPA");
15         EntityManager em = emf.createEntityManager();
16         EntityTransaction transaction = em.getTransaction();
17
18         try {
19             transaction.begin();
20
21             Member3 member3 = em.find(Member3.class,
22                 "hong@spring.com");
23             if(member3 == null) {
24                 System.out.println("존재하지 않습니다.");
25                 transaction.rollback();
26                 return;
27             }
28
29             member3.changeName("전우치");
30             transaction.commit();
31             System.out.println("이름을 변경했습니다.");
32         }
33         catch (Exception e) {
34             transaction.rollback();
35             throw e;
36         }
37
38         emf.close();
39         em.close();
40     }
41 }
```

설정파일 : resources/META-INF/persistence.xml

none 으로 유지한 후 테스트한다.

```
1 package myjpa3;
2
3 import jakarta.persistence.EntityManager;
4
5
6
7
8 public class UseMember04_delete {
9
10     public static void main(String[] args) {
11         //영속성 인스턴스 생성
12         EntityManagerFactory emf =
13             Persistence.createEntityManagerFactory("MyJPA");
14         EntityManager em = emf.createEntityManager();
15         EntityTransaction transaction = em.getTransaction();
16
17         try {
18             transaction.begin();
19
20             //조건에 맞는 레코드 검색
21             Member3 member3 = em.find(Member3.class,
22                 "hong@spring.com");
23             if(member3 == null) {
24                 System.out.println("존재하지 않습니다.");
25                 transaction.rollback();
26                 return;
27             }
28             //레코드 삭제 및 동기화
29             em.remove(member3);
30             transaction.commit();
31             System.out.println("삭제했습니다.");
32         }
33         catch (Exception e) {
34             transaction.rollback();
35             throw e;
36         }
37
38         emf.close();
39         em.close();
40     }
41 }
```

설정파일 : resources/META-INF/persistence.xml

none 으로 유지한 후 테스트한다.

## 네번째실습

설정파일 : resources/META-INF/persistence.xml

<class>myjpa4.Member4</class> ⇒ 이 부분을 myjpa4.Member4 로 수정. create 로 수정 후 테스트.

```
10< persistence-unit name="MyJPA" transaction-type="RESOURCE_LOCAL">
11    <!-- 엔티티로 사용할 클래스의 풀 경로 -->
12    <class>myjpa4.Member4</class>
13
14    <!-- 명시적으로 나열된 클래스들만 엔티티로 인식 -->
15    <exclude-unlisted-classes>true</exclude-unlisted-classes>
16
```

/B21aJPA\_Basic/src/main/java/myjpa4/Member4.java

```
1 package myjpa4;
2
3import java.time.LocalDate;
4
14
15 @Entity
16 @Table(name="JpaMember4")
17 public class Member4 {
18
19     @Id
20     private String email;
21     private String name;
22     @Column(name = "create_date")
23     private LocalDate createDate;
24
25     public Member4() {}
26     public Member4(String email, String name, LocalDate createDate) {
27         super();
28         this.email = email;
29         this.name = name;
30         this.createDate = createDate;
31     }
32     public String getEmail() {
33         return email;
34     }
35     public String getName() {
36         return name;
37     }
38     public LocalDate getCreateDate() {
39         return createDate;
40     }
41
42     public void changeName(String newName) {
43         this.name = newName;
44     }
45 }
```



create 인지 확인해야 함.

```
1 package myjpa4;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10 public class Use01_DummyInsert {
11
12     public static void main(String[] args) {
13
14         //영속성 인스턴스 생성
15         EntityManagerFactory emf =
16             Persistence.createEntityManagerFactory("MyJPA");
17         EntityManager em = emf.createEntityManager();
18         EntityTransaction transaction = em.getTransaction();
19
20         try {
21             transaction.begin();
22
23             //insert 처리
24             Member4 member4;
25             member4 = new Member4("test1@spring.com", "홍길동",
26                 LocalDate.now());
27             em.persist(member4);
28             member4 = new Member4("test2@spring.com", "이순신",
29                 LocalDate.now());
30             em.persist(member4);
31             member4 = new Member4("test3@spring.com", "세종대왕",
32                 LocalDate.now());
33             em.persist(member4);
34             member4 = new Member4("test4@spring.com", "강감찬",
35                 LocalDate.now());
36             em.persist(member4);
37             member4 = new Member4("test5@spring.com", "을지문덕",
38                 LocalDate.now());
39             em.persist(member4);
40             member4 = new Member4("test6@spring.com", "정조대왕",
41                 LocalDate.now());
42
43             em.persist(member4);
44             member4 = new Member4("test7@spring.com", "신사임당",
45                 LocalDate.now());
46             em.persist(member4);
47             member4 = new Member4("test8@spring.com", "선덕여왕",
48                 LocalDate.now());
49             em.persist(member4);
50
51             transaction.commit();
52             System.out.println("입력이 완료되었습니다.");
53         }
54     }
55 }
```

```
53         catch (Exception e) {
54             e.printStackTrace();
55             transaction.rollback();
56         }
57         finally {
58             em.close();
59         }
60
61         //실행전 xml설정파일에서 create로 변경
62         emf.close();
63     }
64 }
```

none 으로 수정 후 테스트

```
1 package myjpa4;
2
3*import java.util.List;
10
11 public class Use02_TypedQuery {
12
13*    public static void main(String[] args) {
14
15        EntityManagerFactory emf =
16            Persistence.createEntityManagerFactory("MyJPA");
17        EntityManager em = emf.createEntityManager();
18        EntityTransaction transaction = em.getTransaction();
19
20        try {
21            transaction.begin();
22
23            String SQL = "SELECT m FROM Member4 m ORDER BY m.name";
24            TypedQuery<Member4> query =
25                em.createQuery(SQL, Member4.class);
26            List<Member4> result = query.getResultList();
27
28            transaction.commit();
29
30            if(result.isEmpty()) {
31                System.out.println("레코드가 없습니다.");
32            }
33
34            else {
35                result.forEach(user ->
36                    System.out.printf("| %s | %s | %tY-%<tm-%<td | \n",
37                        user.getEmail(), user.getName(),
38                        user.getCreateDate()));
39            }
40        } catch (Exception e) {
41            e.printStackTrace();
42            transaction.rollback();
43        }
44
45        //실행전 xml설정파일에서 none으로 변경
46        em.close();
47        emf.close();
48    }
49 }
```

none 인지 확인 후 테스트. 29라인의 검색어는 상황에 맞게 수정해볼것.

```
1 package myjpa4;
2
3 import java.util.List;
10
11 public class Use03_Parameter {
12
13     public static void main(String[] args) {
14
15         //none으로 변경한 후 실행
16         EntityManagerFactory emf =
17             Persistence.createEntityManagerFactory("MyJPA");
18         EntityManager em = emf.createEntityManager();
19         EntityTransaction transaction = em.getTransaction();
20
21         try {
22             transaction.begin();
23
24             String SQL = "SELECT m FROM Member4 m "
25                 + " WHERE m.name = :name "
26                 + " ORDER BY m.name";
27             TypedQuery<Member4> query = em
28                 .createQuery(SQL, Member4.class)
29                 .setParameter("name", "양만춘");
30             List<Member4> result = query.getResultList();
31
32             transaction.commit();
33
34             if(result.isEmpty()) {
35                 System.out.println("레코드가 없습니다.");
36             }
37             else {
38                 result.forEach(user ->
39                     System.out.printf("| %s | %s | %tY-%<tm-%<td | \n",
40                         user.getEmail(), user.getName(),
41                         user.getCreateDate()));
42             }
43         }
44
45         catch (Exception e) {
46             e.printStackTrace();
47             transaction.rollback();
48         }
49
50         em.close();
51         emf.close();
52     }
53 }
```

none 인지 확인 후 테스트.

```
1 package myjpa4;
2
3 import java.util.List;
10
11 public class Use04_Like {
12
13     public static void main(String[] args) {
14
15         //none으로 변경한 후 실행
16         EntityManagerFactory emf =
17             Persistence.createEntityManagerFactory("MyJPA");
18         EntityManager em = emf.createEntityManager();
19         EntityTransaction transaction = em.getTransaction();
20
21         try {
22             transaction.begin();
23
24             String SQL = "SELECT m FROM Member4 m "
25                 + " WHERE m.email like :email "
26                 + " ORDER BY m.name";
27             TypedQuery<Member4> query = em
28                 .createQuery(SQL, Member4.class)
29                 .setParameter("email", "%spring.com%");
30             List<Member4> result = query.getResultList();
31
32             transaction.commit();
33
34             if(result.isEmpty()) {
35                 System.out.println("레코드가 없습니다.");
36             }
37             else {
38
39                 result.forEach(user ->
40                     System.out.printf("| %s | %s | %tY-%<tm-%<td |\n",
41                         user.getEmail(), user.getName(),
42                         user.getCreateDate()));
43             }
44         }
45
46         catch (Exception e) {
47             e.printStackTrace();
48             transaction.rollback();
49         }
50
51         em.close();
52         emf.close();
53     }
54 }
```