

초격차 패키지 Online.

JPA 기본기

PART1 | ORM, JPA, JPQL 개요

기본기 정리하고 넘어가기

PART2 | 기존의 기술들 - iBATIS, MyBatis, JdbcTemplate

JPA 를 사용하기 전에 널리 사용하던 기술들

PART3 | Hibernate vs. Spring Data JPA

둘이 무슨 차이예요?

PART4 | in memory 테스트 DB - H2

날아가도 안전한 테스트 DB를 손쉽게 만드는 방법

JPA 기본기

3 Hibernate vs. Spring Data JPA

Hibernate vs. Spring Data JPA

3. Hibernate vs. Spring Data JPA

하이버네이트? 스프링 JPA?

둘 다 Spring Data JPA 를 사용하면서 볼 수 있는 단어들

- 정확히 무슨 차이인지 알아보자

Hibernate

3. Hibernate vs. Spring Data JPA

Hibernate

"MORE THAN AN ORM, DISCOVER THE HIBERNATE GALAXY."

- 자바생태계를 대표하는 ORM framework
- 스프링 부트에서 채택한 메인 ORM framework
- JPA 표준 스펙을 구현한 JPA Provider
- 고성능, 확장성, 안정성을 표방
- 다양한 하위 제품들로 나뉨
 - Hibernate ORM (최신: 5.5, 스프링 부트: 5.4.32)
 - Hibernate Validator
 - Hibernate Reactive

Hibernate: HQL

3. Hibernate vs. Spring Data JPA

Hibernate Query Language

하이버네이트가 사용하는 SQL 스타일 비표준 쿼리 언어

- 객체 모델에 초점을 맞춰 설계됨
- JPQL 의 바탕이 됨 (JPQL 은 HQL 의 subset)
- JPQL 은 완벽한 HQL 문장이지만, 반대로는 성립하지 않음

```
Query query = entityManager.createQuery(  
    "select p from Person p where p.name like :name"  
);  
  
TypedQuery<Person> typedQuery = entityManager.createQuery(  
    "select p from Person p where p.name like :name", Person.class  
);
```

Hibernate: Criteria

3. Hibernate vs. Spring Data JPA

Criteria query

type-safety 를 제공하는 JPQL의 대안 표현법

```
CriteriaBuilder builder = entityManager.getCriteriaBuilder();

CriteriaQuery<Person> criteria = builder.createQuery( Person.class );
Root<Person> root = criteria.from( Person.class );
criteria.select( root );
criteria.where( builder.equal( root.get( Person_.name ), "John Doe" ) );

List<Person> persons = entityManager.createQuery( criteria ).getResultList();
```

Hibernate: Native SQL Query

3. Hibernate vs. Spring Data JPA

Native SQL Query

특정 DB에 종속된 SQL 도 사용 가능

```
List<Object[]> persons = entityManager.createNativeQuery(  
    "SELECT * FROM Person" )  
.getResultList();
```

Hibernate: 예제 - persistence.xml

3. Hibernate vs. Spring Data JPA

설정 파일은 META-INF/persistence.xml 로 작성

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/
xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
  <persistence-unit name="org.hibernate.tutorial.jpa">
    ...
  </persistence-unit>
</persistence>
```


Hibernate: 예제 - Entity Class

3. Hibernate vs. Spring Data JPA

엔티티 클래스로 객체와 테이블 관계를 정의

```
@Entity
@Table( name = "EVENTS" )
public class Event {
    @Id
    @GeneratedValue(generator="increment")
    @GenericGenerator(name="increment", strategy = "increment")
    public Long getId() {
        return id;
    }

    ...

    public String getTitle() {
        return title;
    }

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "EVENT_DATE")
    public Date getDate() {
        return date;
    }
}
```

Hibernate: 예제 - logic

3. Hibernate vs. Spring Data JPA

```
protected void setUp() throws Exception {
    sessionFactory = Persistence.createEntityManagerFactory( "org.hibernate.tutorial.jpa" );
}
```

```
EntityManager entityManager = sessionFactory.createEntityManager();
entityManager.getTransaction().begin();
entityManager.persist( new Event( "Our very first event!", new Date() ) );
entityManager.persist( new Event( "A follow up event", new Date() ) );
entityManager.getTransaction().commit();
entityManager.close();
```

```
EntityManager entityManager = sessionFactory.createEntityManager();
entityManager.getTransaction().begin();
List<Event> result = entityManager.createQuery( "from Event", Event.class ).getResultList();
for ( Event event : result ) {
    System.out.println( "Event (" + event.getDate() + ") : " + event.getTitle() );
}
entityManager.getTransaction().commit();
entityManager.close();
```

Spring Data JPA

3. Hibernate vs. Spring Data JPA














Spring Data JPA

스프링에서 제공하는 JPA 추상화 모듈

- JPA 구현체의 사용을 한 번 더, Repository 라는 개념으로 추상화
- JPA 구현체의 사용을 감추고, 다양한 지원과 설정 방법을 제공
- JPA 기본 구현체로 Hibernate 사용
- Querydsl 지원

Spring Data JPA: compile dependencies

3.
Hibernate vs.
Spring Data JPA

Compile Dependencies (13)		
Category/License		Group
Apache 2.0		com.querydsl » querydsl-jpa (optional)
AOP EPL 2.0		org.aspectj » aspectjrt
JPA EDL 1.0 EPL 2.0		org.eclipse.persistence » org.eclipse.persistence.jpa (optional)
O/R Mapping LGPL 2.1		org.hibernate » hibernate-core (optional)
O/R Mapping Apache 2.0		org.springframework » spring-orm
Dep Injection Apache 2.0		org.springframework » spring-context
AOP Apache 2.0		org.springframework » spring-aop
Transactions Apache 2.0		org.springframework » spring-tx
Dep Injection Apache 2.0		org.springframework » spring-beans
Core Utils Apache 2.0		org.springframework » spring-core
AOP Apache 2.0		org.springframework » spring-aspects (optional)
Apache 2.0		org.springframework.data » spring-data-commons
BSD 3-clause		org.threeten » threetenbp (optional)

Spring Data JPA 사용하면서 알아야 할 사실

3. Hibernate vs. Spring Data JPA

Spring Data JPA 를 사용한다면

JPA, 하이버네이트를 몰라도 되어야 한다

- EntityManager 를 직접 사용하지 않는다
- JPQL 을 직접 사용하지 않는다
- persist(), merge(), close() 를 직접 사용하지 않는다
- 트랜잭션을 getTransaction(), commit(), rollback() 으로 관리하지 않는다
- 코드가 하이버네이트를 직접 사용하고 있다면
 - 꼭 필요한 코드인지, 아니면 Spring Data JPA 로 할 수 있는 일인지 확인하세요
 - 그 코드는 하이버네이트와 직접적인 연관 관계를 가지게 됨
 - 추상화의 이점을 포기하게 되는 셈

Reference

3. Hibernate vs. Spring Data JPA

- <https://hibernate.org/>
- [https://en.wikipedia.org/wiki/Hibernate_\(framework\)](https://en.wikipedia.org/wiki/Hibernate_(framework))
- <https://docs.spring.io/spring-data/jpa/docs/2.5.5/reference/html/#reference>