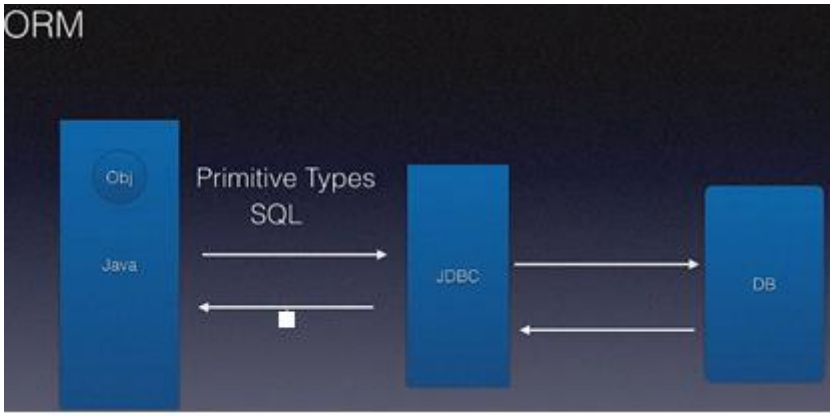
1. ORM(Object Relational Mapping) 소개

가. ORM 개념(스프링 프레임워크에서만이 아닌 일반적인 개념)

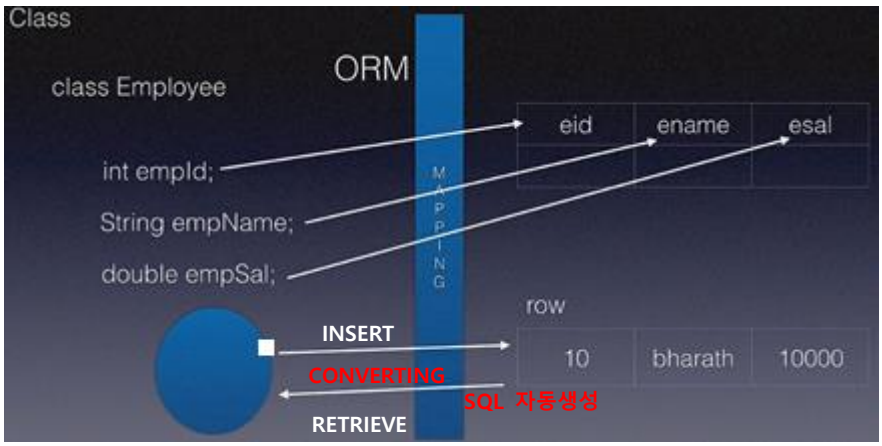
- JDBC, Spring JDBC 데이터베이스 오퍼레이션은
- 오브젝트안에 있는 원시타입을 가지고 개발자가 SQL 문장을 만들고 실행
- 실행결과 되돌아온 데이터를 자바오브젝트로 변경
- 개발자가 해야 하는 많은 일



나. ORM 툴의 등장: 개발자가 해야 하는 많은 일이 너무 많다

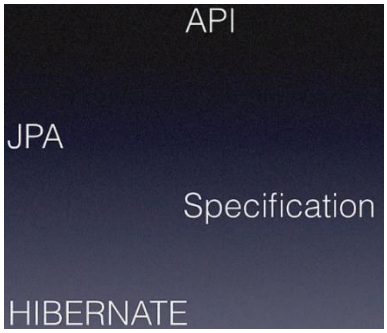
다. ORM 일반적인 개념

- 엔티티 클래스라 불리는 클래스 만들기(자바 POJO 클래스, bean)
- 매핑을 제공: 클래스 안에 있는 모든 필드는 데이터베이스 안의 테이블 칼럼과 매핑
- 일단 매핑을 하면, ORM 툴은 자동적으로 우리의 객체를 데이터베이스의 레코드로 변경
- 어떠한 SQL도 작성하지 않음
- 우리가 매핑 정보를 제공하면, ORM 툴이 즉석에서 필요한 SQL 문장을 생성



라. 자바의 ORM 개념

- JPA(JAVA EE의 ORM 표준
- Java Persistence API)
- JPA: 일부 애노테이션, 인터페이스 클래스들
- JPA 표준 구현체: 하이버네이트(HIBERNATE)

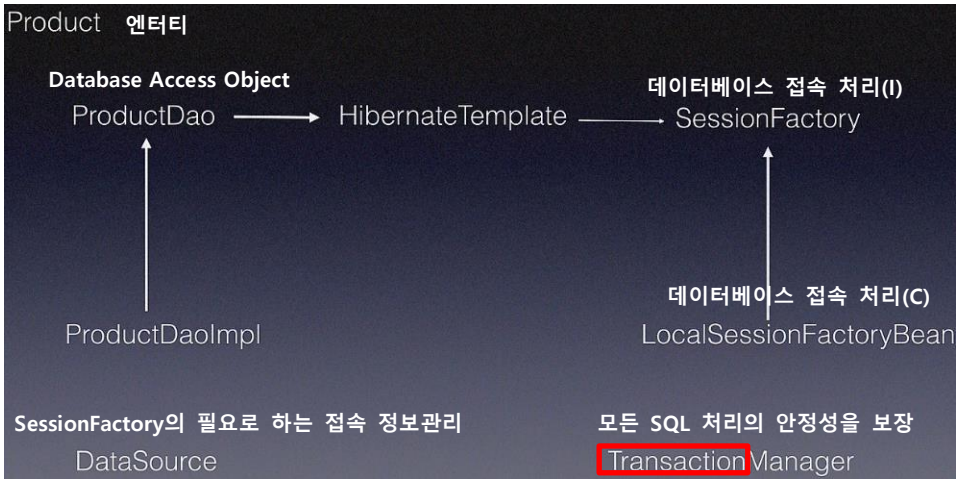


2. 스프링 프레임워크 ORM 소개: HibernateTemplate

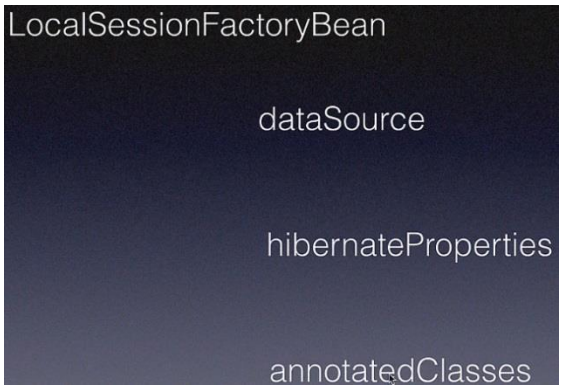
가. 개요

- HibernateTemplate
- 모든 복잡하고 지겨운 코드를 숨김: 하이버네트 세션을 생성해서
- 오퍼페이션 실행: SAVE, UPDATE, DELETE, GET
- HibernateTemplate 사용->메소드호출->내부적으로 필요한 SQL 문장을 생성하고 실행

나. 엔터티 DAO HibernateTemplate(SQL) SessionFactory(데이터베이스 접속:DataSource 필요)



다. LocalSessionFactoryBean: 세 개의 인자 또는 세 개의 빈을 가짐



라. Hibernate Properties

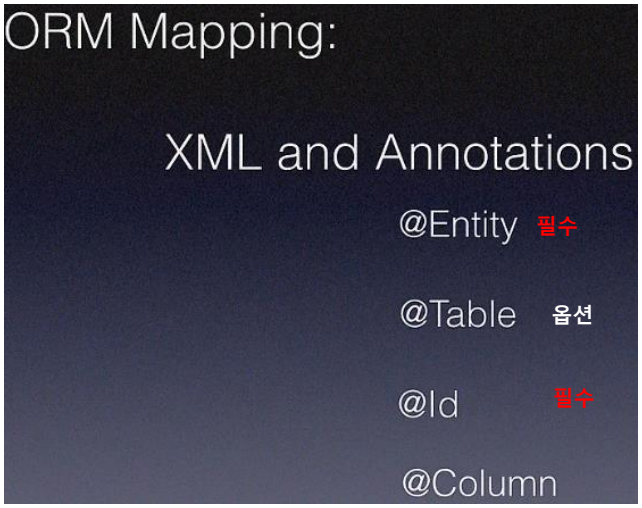
- Key : value
- Hibernate.dialect = org.hibernate.dialect.MYSQLDialect(특정 연산을 위해 오브젝트로부터 SQL 생성하는 책임을지는 클래스: 예)save->insert, update->update 등등)
- Hibernate.show_sql(디폴트는 보여주지 않음) = true

3. 엔터티를 데이터베이스 테이블에 매핑하기

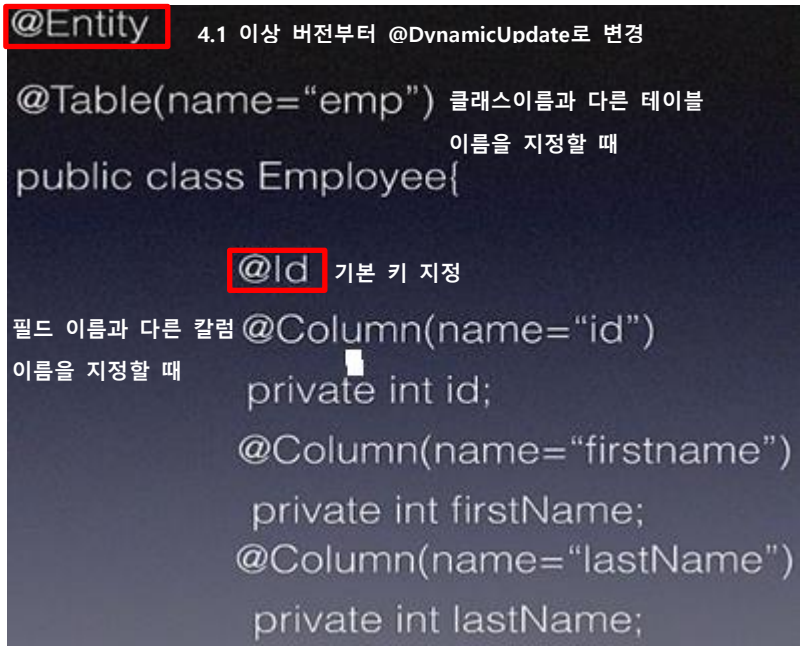
가. 개발자가 할 일: 매핑 정보 제공(클래스내의 필드와 데이터베이스 테이블의 칼럼 사이에)

나. 방법: XML과 애노테이션

다. JPA 애노테이션: @Entity, @Table, @Id, @Column

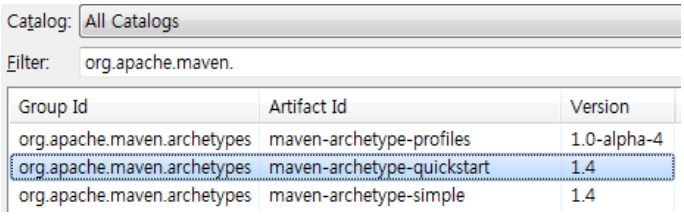


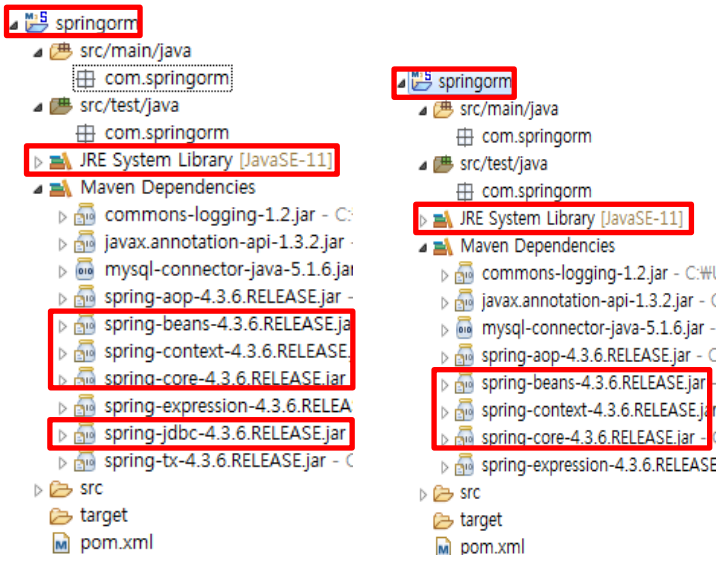
- 예제

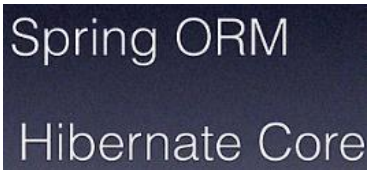


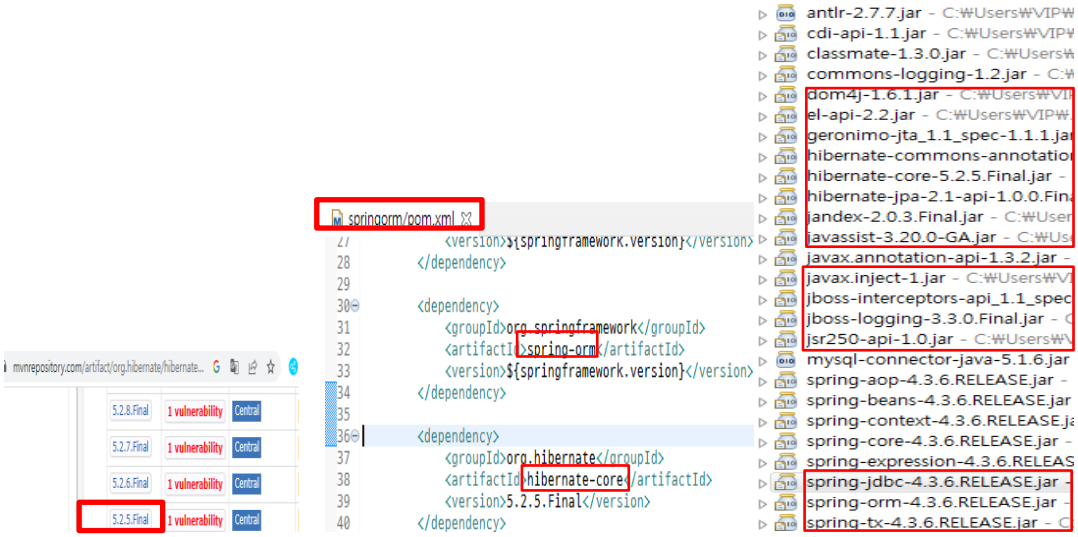
4. 테이블 생성

5. 메이븐 프로젝트 만들기



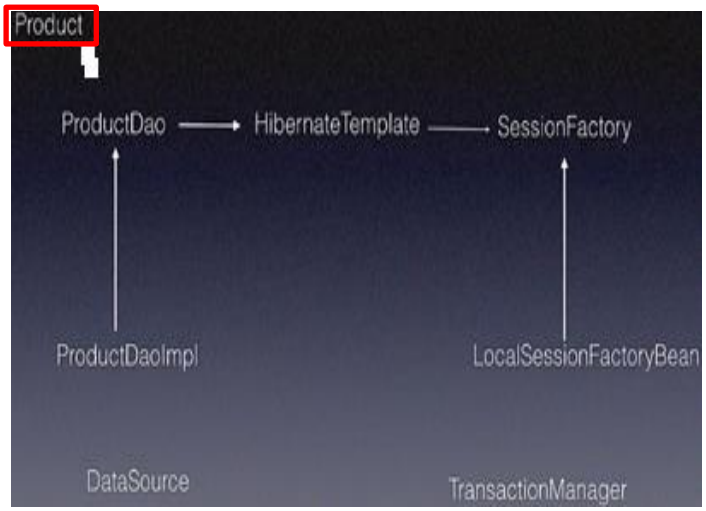






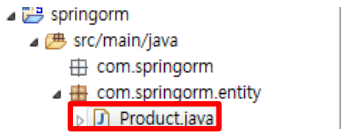
6. 엔터티 만들기

가. 개요



나. 엔터티(=모델) 만들기

- 클래스생성(Product, com.springorm.entity)
- getter/setter생성(Source>Generate Getters and Setters...)
- 소스 정렬(Ctrl+Shift+F)



Product.java

```java
package com.springorm.product.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "product")
public class Product {

    @Id
    private int id;
    @Column(name = "name")
    private String name;
    @Column(name = "description")
    private String desc;
    @Column(name = "price")
    private double price;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

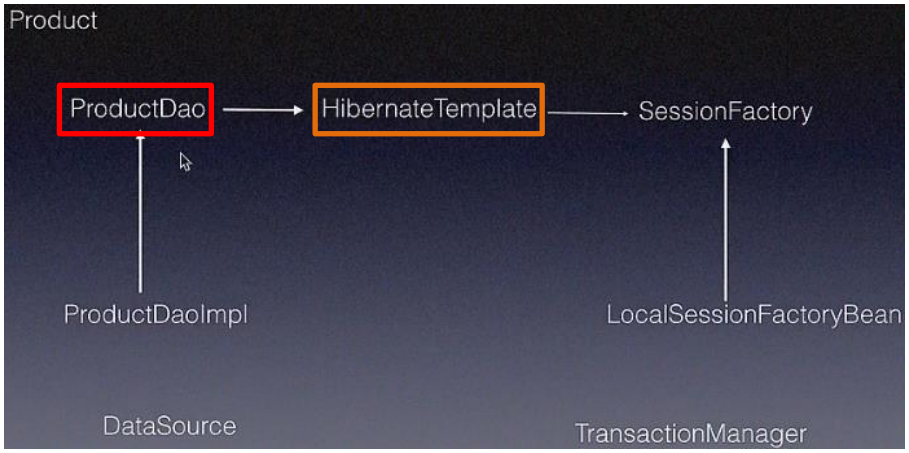
@Table은 옵션으로 클래스이름과 테이블이름 다를 경우 name으 재지정

@Id: 기본키 지정

@Column: 테이블 칼럼과 이름이 다를 때, name으로 재지정

@Entity 애노테이션으로 일반 자바클래스를 데이터베이스 테이블과 매핑되는 엔터티 클래스로 변경

7. DAO 만들기



Product.java | springorm/pom.xml

```
19⊖     <dependency>
20          <groupId>org.springframework</groupId>
21          <artifactId>spring-core</artifactId>
22          <version>${springframework.version}</version>
23      </dependency>
24⊖     <dependency>
25          <groupId>org.springframework</groupId>
26          <artifactId>spring-context</artifactId>
27          <version>${springframework.version}</version>
28      </dependency>
29
30⊖     <dependency>
31          <groupId>org.springframework</groupId>
32          <artifactId>spring-orm</artifactId>
33          <version>${springframework.version}</version>
34      </dependency>
35
36⊖     <dependency>
37          <groupId>org.hibernate</groupId>
38          <artifactId>hibernate-core</artifactId>
39          <version>5.4.33.Final</version>
40      </dependency>
```

ProductDao.java | ProductDaoImpl.java

```
1  package com.springorm.product.dao.impl;
2
3⊖ import org.springframework.orm.hibernate5.HibernateTemplate;
4
5  import com.springorm.product.dao.ProductDao;
6  import com.springorm.product.entity.Product;
7
8  public class ProductDaoImpl implements ProductDao {
9
10     HibernateTemplate hibernateTemplate;
11
12⊖    @Override
13     public int create(Product product) {
14
15         return 0;
16     }
17 }
```

!!!주의: hibernate-core를 pom에 설정할 때 설치버전에 주의

기존 JDK 1.8까지는 5.2버전도 가능했으나, JDK11에서는 **최소한**

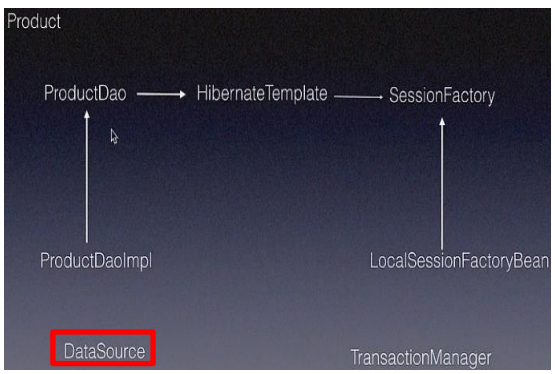
5.4이후 버전을 설치해야 호환이 가능하다.

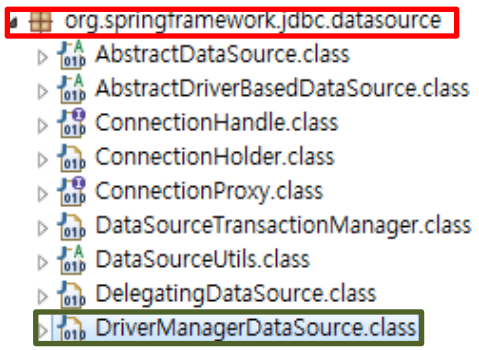
pom.xml에 Hibernate를 설치하면

springframework.jdbc가 의존관계로

설치되기 때문에, 기존의 springframework.jdbc 삭제

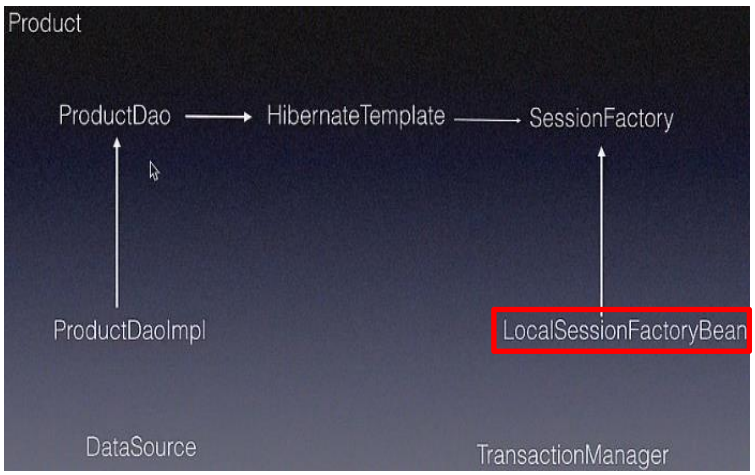
8. 설정파일 만들기

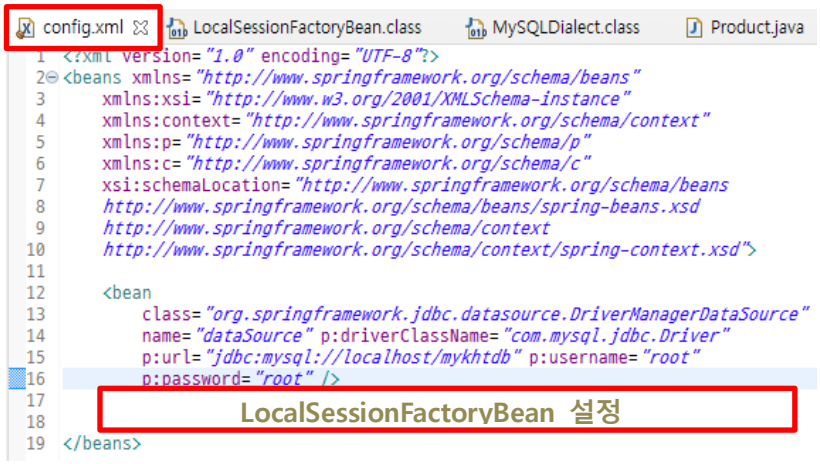


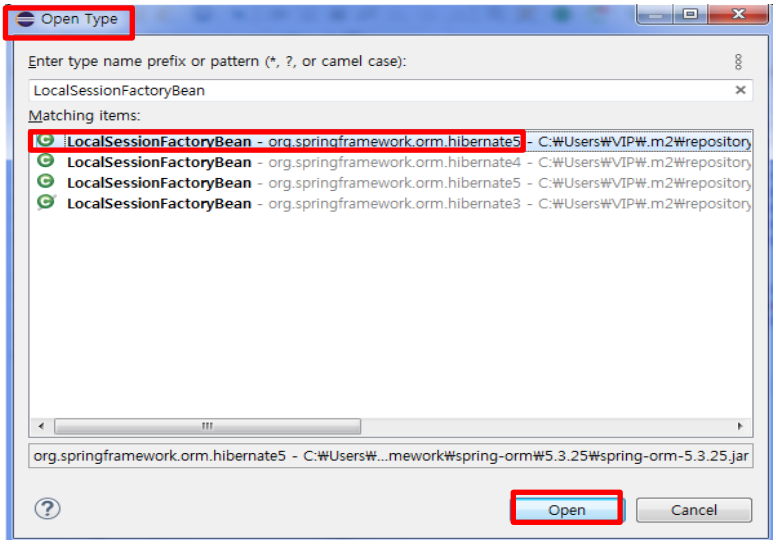


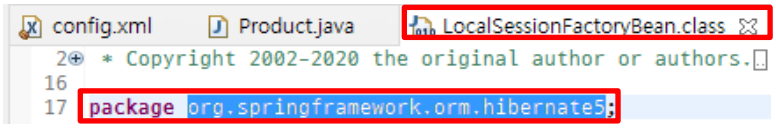

springorm
src/main/java
com.springorm.product.dao
ProductDao.java
com.springorm.product.dao.impl
ProductDaoImpl.java
com.springorm.product.entity
Product.java
com.springorm.product.test
config.xml
config.xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:c="http://www.springframework.org/schema/c"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<bean
class="org.springframework.jdbc.datasource.DriverManagerDataSource"
name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
p:password="root" />
</beans>

9. 세션팩토리 설정

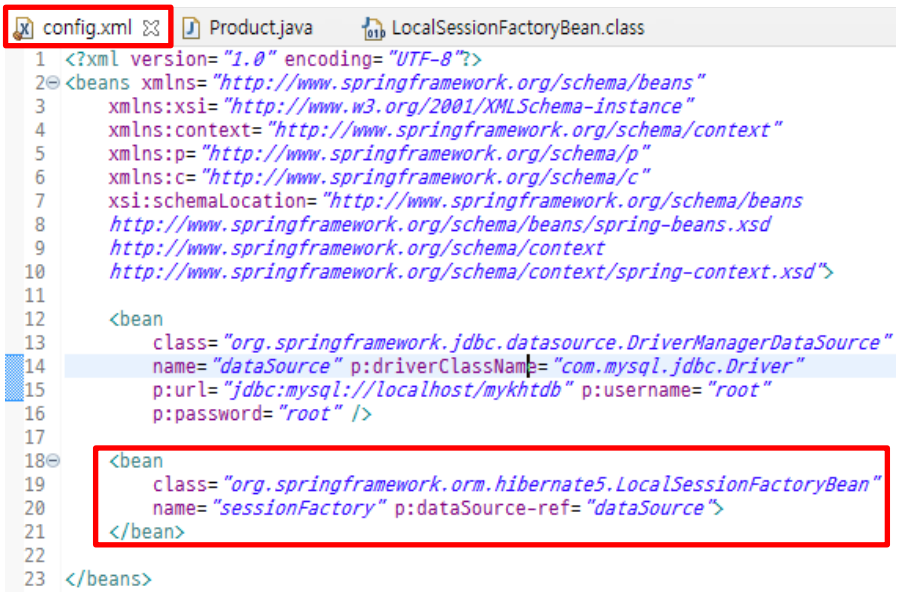

Product
ProductDao
HibernateTemplate
SessionFactory
ProductDaoImpl
LocalSessionFactoryBean
DataSource
TransactionManager


config.xml
LocalSessionFactoryBean.class
MySQLDialect.class
Product.java
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:c="http://www.springframework.org/schema/c"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<bean
class="org.springframework.jdbc.datasource.DriverManagerDataSource"
name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
p:password="root" />
LocalSessionFactoryBean 설정
</beans>



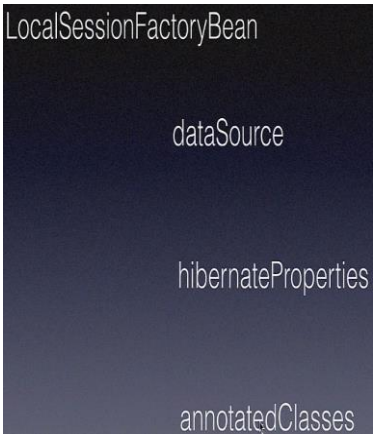



```
public class LocalSessionFactoryBean extends Hibe
        implements FactoryBean<SessionFactory>, R
```



config.xml | Product.java

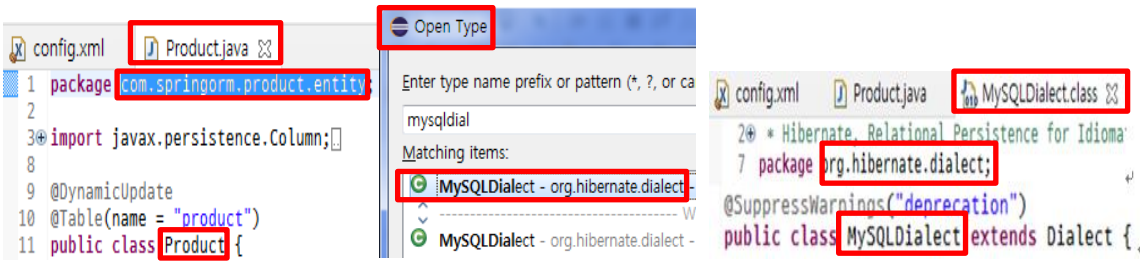
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <bean
        class="org.springframework.jdbc.datasource.DriverManagerDataSource"
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
        p:password="root" />

    <bean
        class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties"> ①
            <props>
                <prop key=""></prop>
                <prop key=""></prop> ②
            </props>
        </property>
        <property name="annotatedClasses"> ③
            <list>
                <value></value>
            </list>
        </property>
    </bean>

</beans>
```



① **데이터 소스**: 데이터베이스 접속 정보

② **하이버네이트 프로퍼티**: SQL 생성을 담당하는 클래스/ SQL display 출력여부

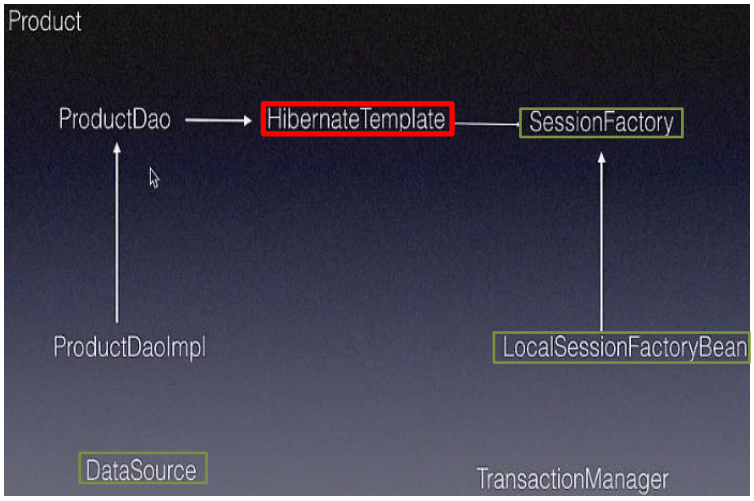
③ **애노테이트클래스**: 데이터베이스 테이블과 매핑될 엔터티 클래스 지정

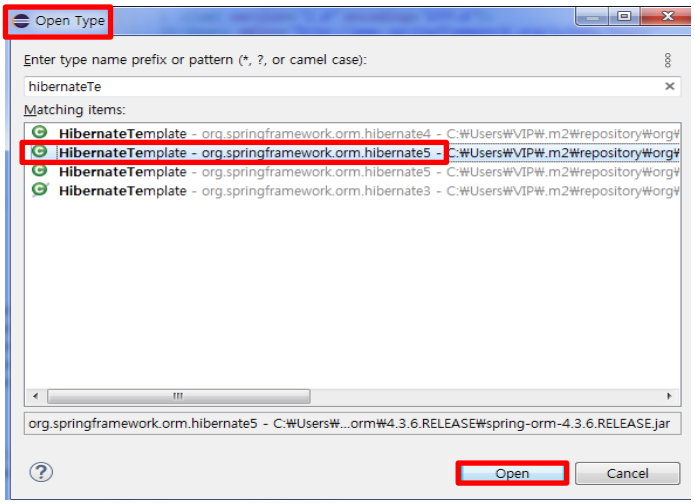


config.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <bean
        class="org.springframework.jdbc.datasource.DriverManagerDataSource"
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
        p:password="root" />

    <bean
        class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialet">
                    org.hibernate.dialect.MySQLDialect
                </prop>
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
        <property name="annotatedClasses">
            <list>
                <value>com.springorm.product.entity.Product</value>
            </list>
        </property>
    </bean>
</beans>
```

10. Hibernate Template 을 설정하고 사용하기





config.xml HibernateTemplate.class

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <bean
        class="org.springframework.jdbc.datasource.DriverManagerDataSource
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
        p:password="root" />

    <bean
        class="org.springframework.orm.hibernate5.LocalSessionFactoryBean
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialet">
                    org.hibernate.dialect.MySQLDialect
                </prop>
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
        <property name="annotatedClasses">
            <list>
                <value>com.springorm.product.entity.Product</value>
            </list>
        </property>
    </bean>

    <bean
        class="org.springframework.orm.hibernate5.HibernateTemplate"
        name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />

</beans>
```

config.xml ProductDaoImpl.java

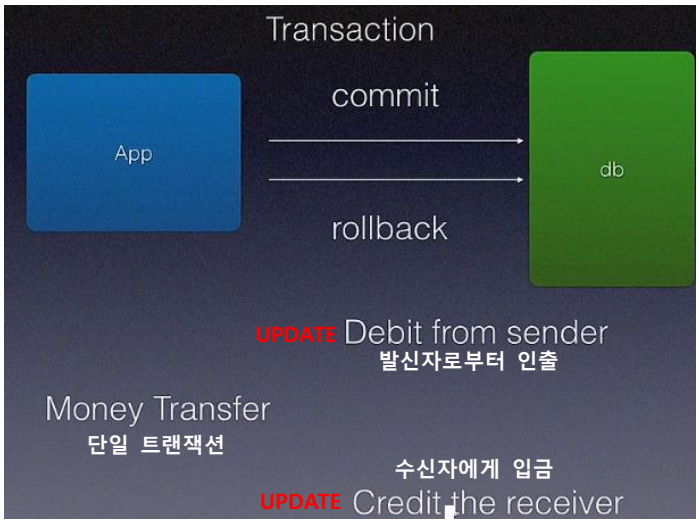
```java
package com.springorm.product.dao.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.orm.hibernate5.HibernateTemplate;
import org.springframework.stereotype.Component;

import com.springorm.product.dao.ProductDao;
import com.springorm.product.entity.Product;

@Component
public class ProductDaoImpl implements ProductDao {

    @Autowired
    HibernateTemplate hibernateTemplate;

    @Override
    public int create(Product product) {

        return 0;
    }
}
```

config.xml ProductDaoImpl.java

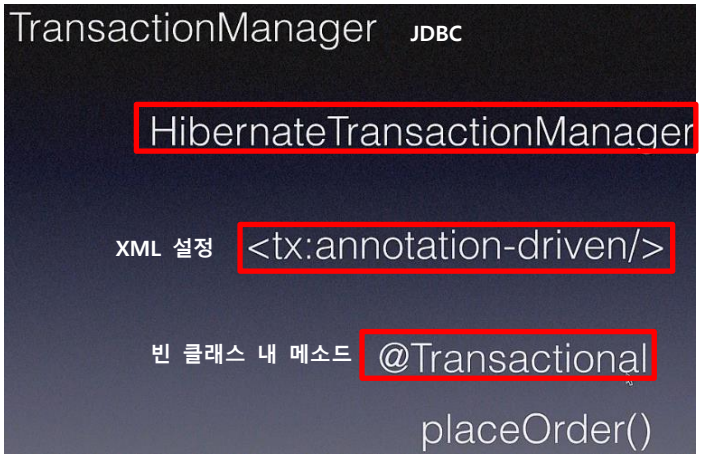
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="com.springorm.product.dao.impl" />

    <bean
        class="org.springframework.jdbc.datasource.DriverManagerDataSource"
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
```

11. 트랜잭션 매니저(Transaction Manager)

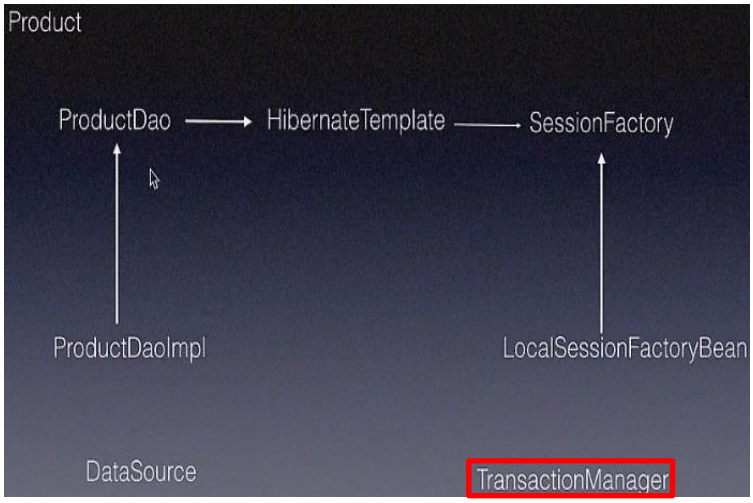
가. 개요

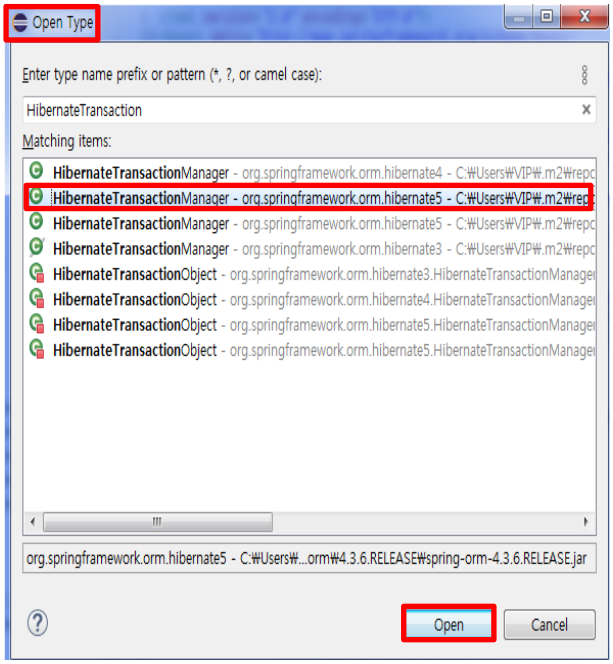


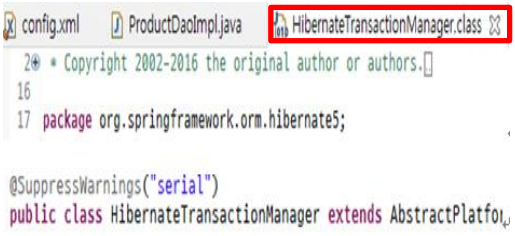
나. 트랜잭션 매니저(TransactionManager)



12. 트랜잭션매니저 설정







config.xml | ProductDaoImpl.java

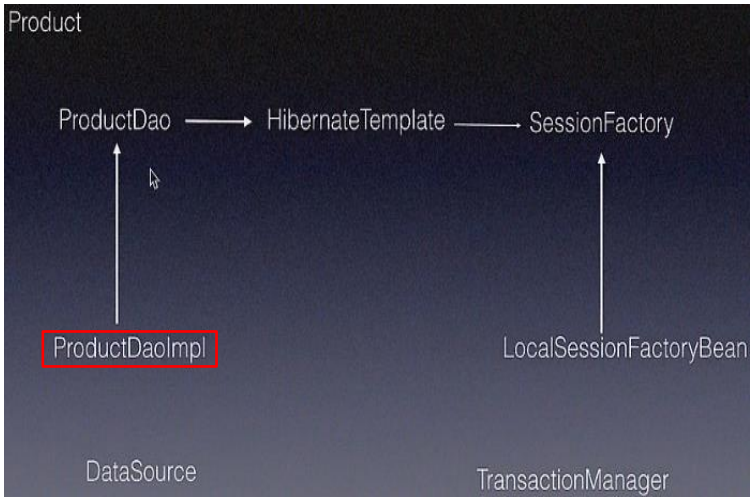
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">

    <tx:annotation-driven/> 트랜잭션 애노테이션 사용 활성화

    <context:component-scan base-package="com.springorm.product.dao.impl" />

    <bean
        class="org.springframework.jdbc.datasource.DriverManagerDataSource"
        name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
        p:password="root" />

    <bean
        class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
        name="sessionFactory" p:dataSource-ref="dataSource">
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialet">
                    org.hibernate.dialect.MySQLDialect
                </prop>
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
        <property name="annotatedClasses">
            <list>
                <value>com.springorm.product.entity.Product</value>
            </list>
        </property>
    </bean>

    <bean
        class="org.springframework.orm.hibernate5.HibernateTemplate"
        name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />

    <bean 트랜잭션 매니저 빈 생성(세션팩토리 참고)
        class="org.springframework.orm.hibernate5.HibernateTransactionManager"
        name="transactionManager" p:sessionFactory-ref="sessionFactory"/>
```

스프링 부트는 대부분 설정을 애노테이션을 사용하지만, 스프링 레거시의 경우 가급적 애노테이션을 사용하지만, 직접 만들지 않는 클래스는 직접 애노테이션을 사용할 없다는 점이 불편하다. 따라서, 애노테이션과 XML 설정을 혼용할 수 밖에 없다.

13. create 메소드 구현

가. 위치: **com.springorm.product.dao.impl.ProductDaoImpl**

나. 개념



다. 메소드 구현

14. 테스트 클래스 만들고 실행

15. 업데이트 메소드 구현

16. 삭제 메소드 구현

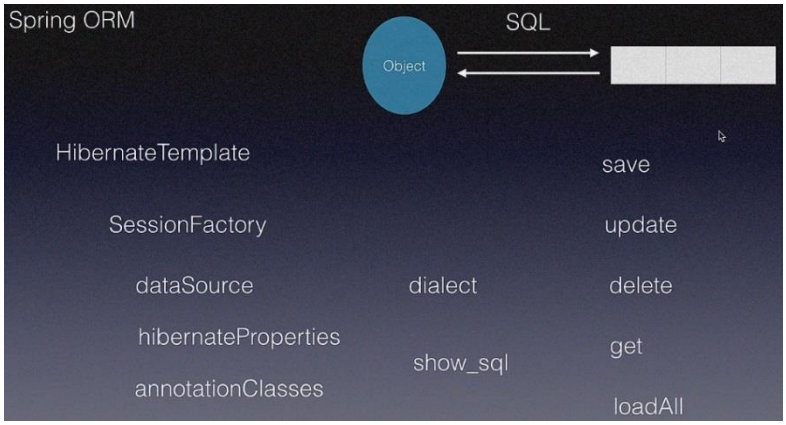
17. 단일 레코드 가져오기

가. 샘플 데이터 입력

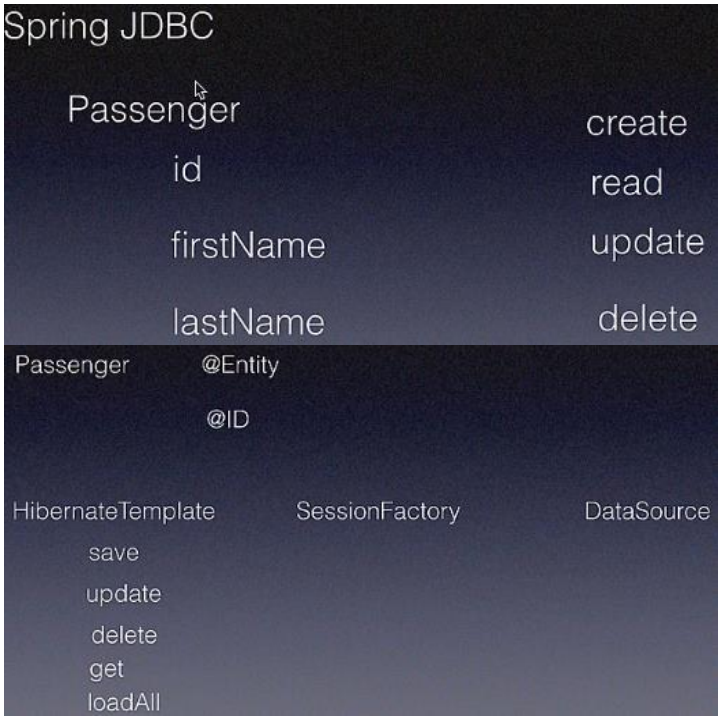
나. 샘플코드 구현

18. 다중 레코드 가져오기

19. 요약



20. 과제


Spring JDBC
Passenger
id
firstName
lastName
create
read
update
delete
Passenger
@Entity
@ID
HibernateTemplate
save
update
delete
get
loadAll
SessionFactory
DataSource