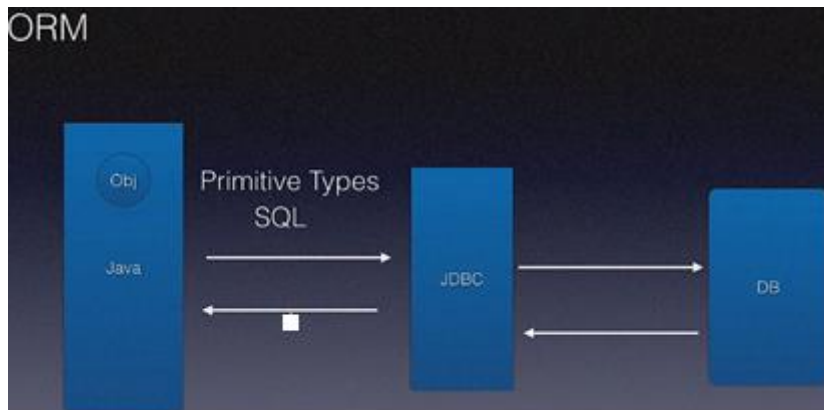


## 1. ORM(Object Relational Mapping) 소개

### 가. ORM 개념(스프링 프레임워크에서만이 아닌 일반적인 개념)

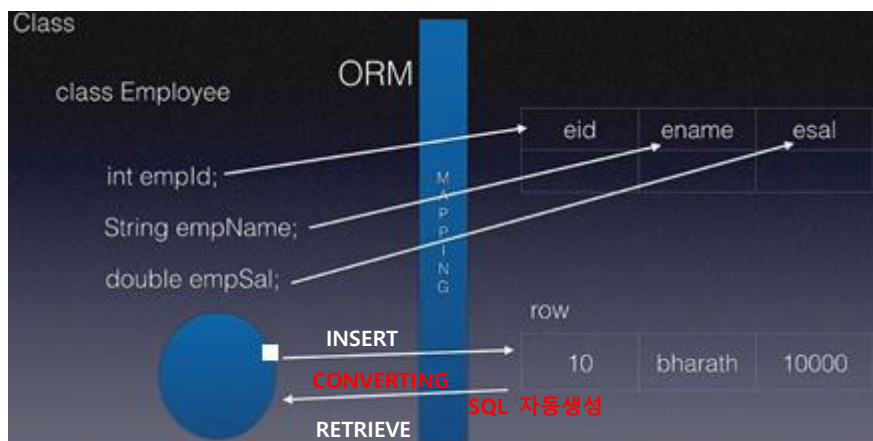
- JDBC, Spring JDBC 데이터베이스 오퍼레이션은
- 오브젝트안에 있는 원시타입을 가지고 개발자가 SQL 문장을 만들고 실행
- 실행결과 되돌아온 데이터를 자바오브젝트로 변경
- 개발자가 해야 하는 많은 일



나. ORM 툴의 등장: 개발자가 해야 하는 많은 일이 너무 많다

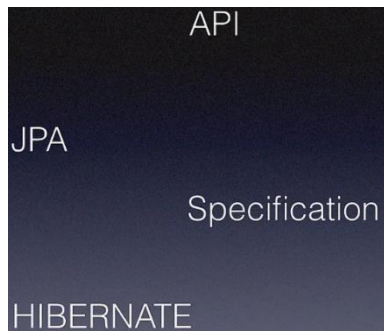
### 다. ORM 일반적인 개념

- 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

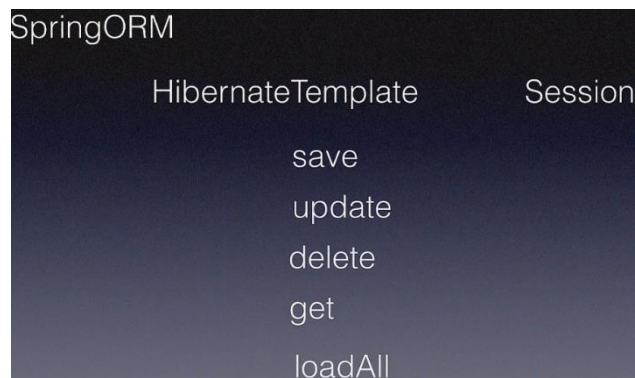
- JPA(JAVA EE의 ORM 표준
- Java Persistence API)
- JPA: 일부 애노테이션, 인터페이스 클래스들
- JPA 표준 구현체: 하이버네이트(HIBERNATE)



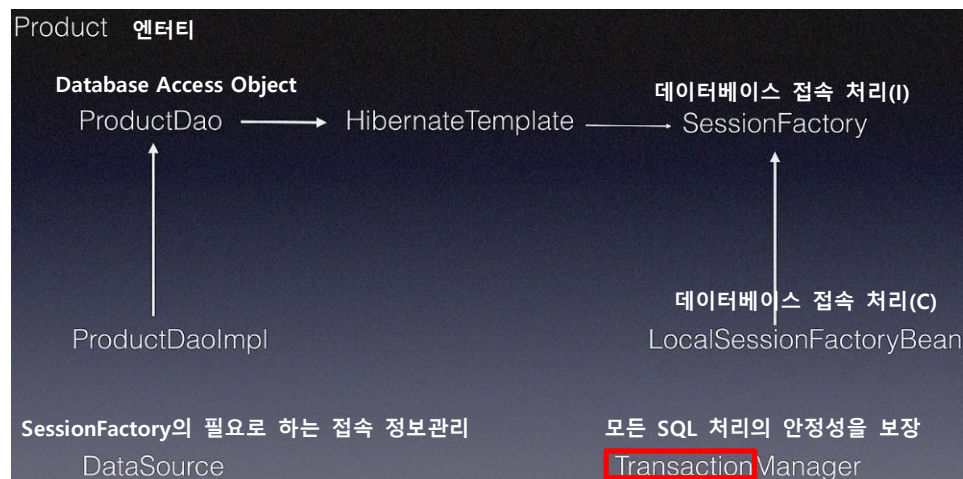
## 2. 스프링 프레임워크 ORM 소개: HibernateTemplate

### 가. 개요

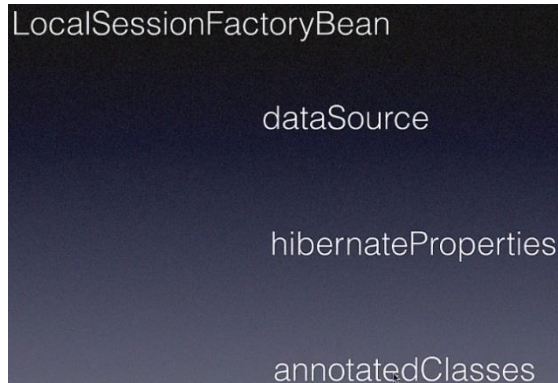
- HibernateTemplate
- 모든 복잡하고 지겨운 코드를 숨김: 하이버네트 세션을 생성해서
- 오퍼레이션 실행: SAVE, UPDATE, DELETE, GET
- HibernateTemplate 사용->메소드호출->내부적으로 필요한 SQL 문장을 생성하고 실행



### 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 책임을 지는 클래스: 예)save->insert, update->update 등등)
- Hibernate.show\_sql(디폴트는 보여주지 않음) = true

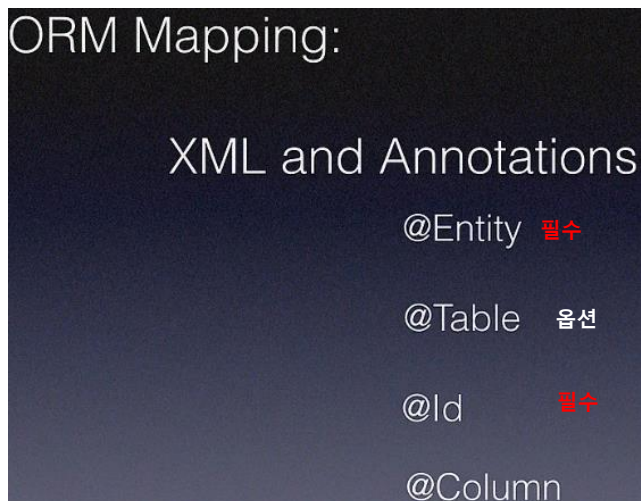


3. 엔티티를 데이터베이스 테이블에 매핑하기

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

**@Entity** 4.1 이상 버전부터 @DynamicUpdate로 변경

@Table(name="emp") 클래스이름과 다른 테이블 이름을 지정할 때

```
public class Employee{
```

**@Id** 기본 키 지정

필드 이름과 다른 컬럼 이름을 지정할 때

```
@Column(name="id")
private int id;
@Column(name="firstname")
private int firstName;
@Column(name="lastName")
private int lastName;
```

4. 테이블 생성

5. 메이븐 프로젝트 만들기

Catalog: All Catalogs

Filter: org.apache.maven.

Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.4
org.apache.maven.archetypes	maven-archetype-simple	1.4

**springorm**

- src/main/java
  - com.springorm
- src/test/java
  - com.springorm
- JRE System Library [JavaSE-11]
- Maven Dependencies
  - commons-logging-1.2.jar - C:\
  - javax.annotation-api-1.3.2.jar - C:\
  - mysql-connector-java-5.1.6.jar - C:\
  - spring-aop-4.3.6.RELEASE.jar - C:\
  - spring-beans-4.3.6.RELEASE.jar
  - spring-context-4.3.6.RELEASE.jar
  - spring-core-4.3.6.RELEASE.jar
  - spring-expression-4.3.6.RELEASE.jar
  - spring-jdbc-4.3.6.RELEASE.jar
  - spring-tx-4.3.6.RELEASE.jar - C:\
- src
- target
- pom.xml

**springorm**

- src/main/java
  - com.springorm
- src/test/java
  - com.springorm
- JRE System Library [JavaSE-11]
- Maven Dependencies
  - commons-logging-1.2.jar - C:\
  - javax.annotation-api-1.3.2.jar - C:\
  - mysql-connector-java-5.1.6.jar - C:\
  - spring-aop-4.3.6.RELEASE.jar - C:\
  - spring-beans-4.3.6.RELEASE.jar
  - spring-context-4.3.6.RELEASE.jar
  - spring-core-4.3.6.RELEASE.jar
  - spring-expression-4.3.6.RELEASE.jar
- src
- target
- pom.xml

Spring ORM  
Hibernate Core

springorm/pom.xml

```

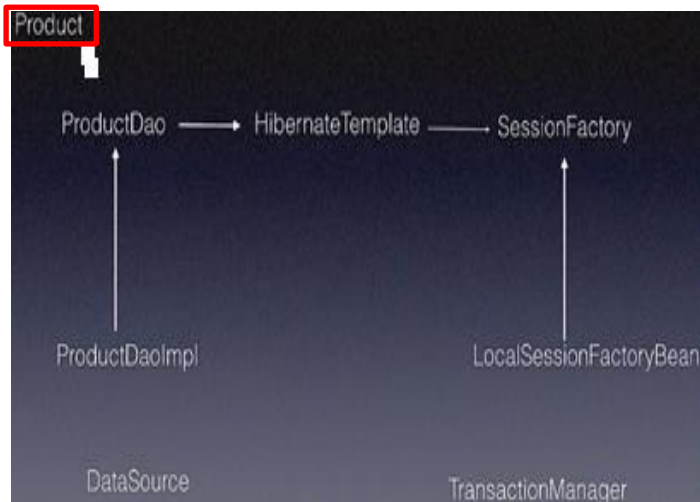
27 <version>${springframework.version}</version>
28 </dependency>
29
30 <dependency>
31 <groupId>org.springframework</groupId>
32 <artifactId>spring-orm</artifactId>
33 <version>${springframework.version}</version>
34 </dependency>
35
36 <dependency>
37 <groupId>org.hibernate</groupId>
38 <artifactId>hibernate-core</artifactId>
39 <version>5.2.5.Final</version>
40 </dependency>

```

antlr-2.7.7.jar - C:\Users\WVIPW  
cdi-api-1.1.jar - C:\Users\WVIPW  
classmate-1.3.0.jar - C:\Users\WVIPW  
commons-logging-1.2.jar - C:\Users\WVIPW  
dom4j-1.6.1.jar - C:\Users\WVIPW  
el-api-2.2.jar - C:\Users\WVIPW  
geronimo-jta\_1.1\_spec-1.1.1.jar - C:\Users\WVIPW  
hibernate-commons-annotation-5.2.5.Final.jar - C:\Users\WVIPW  
hibernate-core-5.2.5.Final.jar - C:\Users\WVIPW  
hibernate-jpa-2.1-api-1.0.0.Final.jar - C:\Users\WVIPW  
jandex-2.0.3.Final.jar - C:\Users\WVIPW  
javassist-3.20.0-GA.jar - C:\Users\WVIPW  
javax.annotation-api-1.3.2.jar - C:\Users\WVIPW  
javax.inject-1.jar - C:\Users\WVIPW  
jboss-interceptors-api-1.1\_spec-1.1.1.Final.jar - C:\Users\WVIPW  
jboss-logging-3.3.0.Final.jar - C:\Users\WVIPW  
jsr250-api-1.0.jar - C:\Users\WVIPW  
mysql-connector-java-5.1.6.jar - C:\Users\WVIPW  
spring-aop-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-beans-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-context-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-core-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-expression-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-jdbc-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-orm-4.3.6.RELEASE.jar - C:\Users\WVIPW  
spring-tx-4.3.6.RELEASE.jar - C:\Users\WVIPW

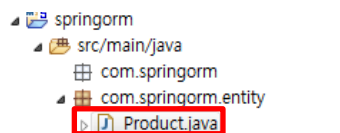
## 6. 엔터티 만들기

### 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
1 package com.springorm.product.entity;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.Id;
6 import javax.persistence.Table;
7
8 @Entity
9 @Table(name = "product") @Table은 옵션으로 클래스이름과 테이블이름 다를 경우 name으로 재지정
10 public class Product {
11
12     @Id
13     private int id;
14     @Column(name = "name")
15     private String name;
16     @Column(name = "description")
17     private String desc;
18     @Column(name = "price")
19     private double price;
20
21     public int getId() {
22         return id;
23     }
24
25     public void setId(int id) {
26         this.id = id;
27     }
28
29     public String getName() {
30         return name;
31     }
32
33     public void setName(String name) {
34         this.name = name;
35     }
36
37     public String getDesc() {
38         return desc;
39     }
40
41     public void setDesc(String desc) {
42         this.desc = desc;
43     }
44
45     public double getPrice() {
46         return price;
47     }
48
49     public void setPrice(double price) {
50         this.price = price;
51     }
52 }

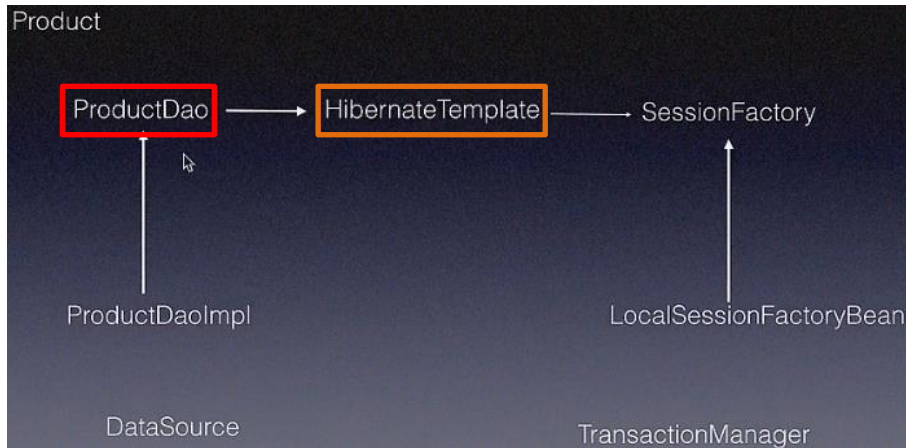
```

@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
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

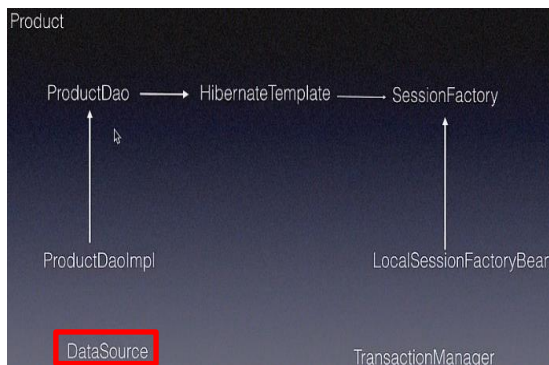
5.4이후 버전을 설치해야 호환이 가능하다.

pom.xml에 Hibernate를 설치하면

springframework.jdbc가 의존관계로

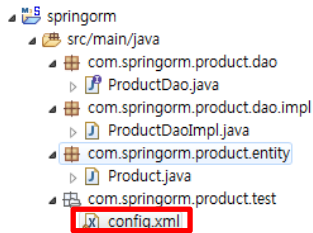
설치되기 때문에, 기존의 springframework.jdbc 삭제

## 8. 설정파일 만들기



```

org.springframework.jdbc.datasource
├── AbstractDataSource.class
├── AbstractDriverBasedDataSource.class
├── ConnectionHandle.class
├── ConnectionHolder.class
├── ConnectionProxy.class
├── DataSourceTransactionManager.class
├── DataSourceUtils.class
├── DelegatingDataSource.class
└── DriverManagerDataSource.class
  
```

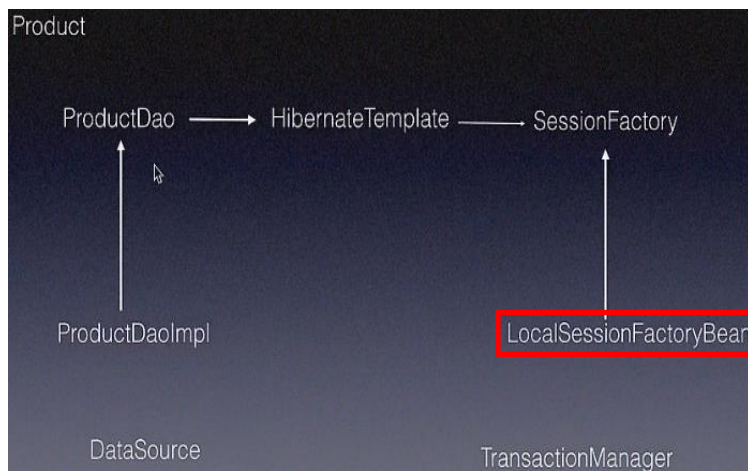


```

config.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xmlns:p="http://www.springframework.org/schema/p"
6     xmlns:c="http://www.springframework.org/schema/c"
7     xsi:schemaLocation="http://www.springframework.org/schema/beans
8         http://www.springframework.org/schema/beans/spring-beans.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context.xsd">
11
12     <bean
13         class="org.springframework.jdbc.datasource.DriverManagerDataSource"
14         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
15         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
16         p:password="root" />
17 </beans>

```

## 9. 세션팩토리 설정

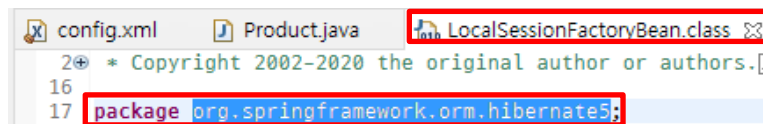
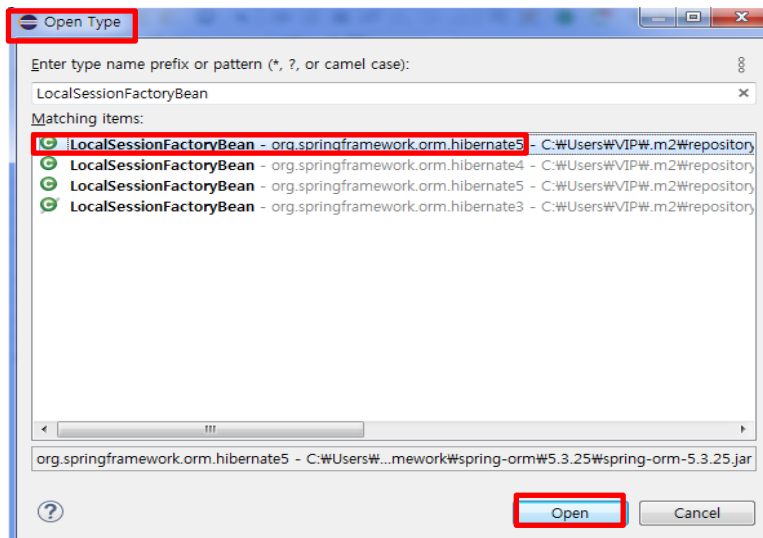


```

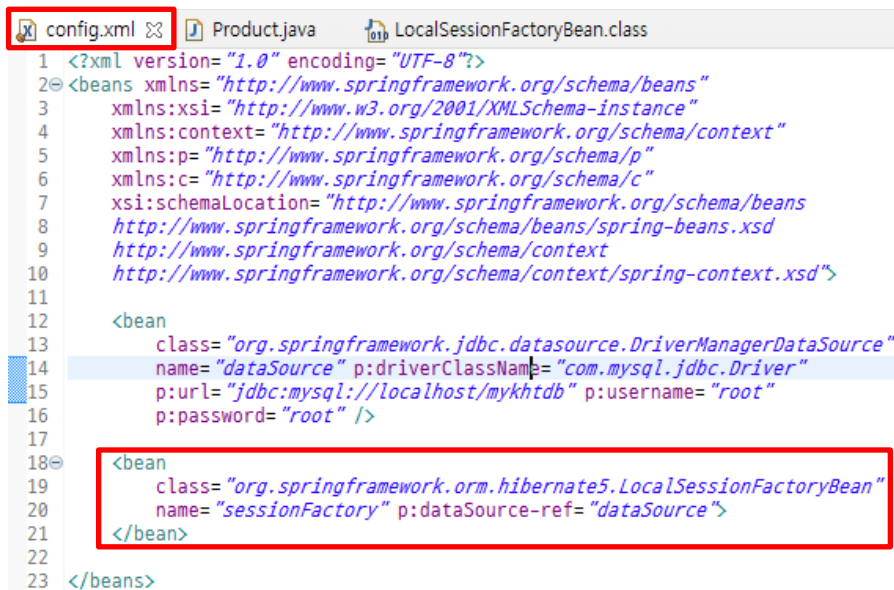
config.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xmlns:p="http://www.springframework.org/schema/p"
6     xmlns:c="http://www.springframework.org/schema/c"
7     xsi:schemaLocation="http://www.springframework.org/schema/beans
8         http://www.springframework.org/schema/beans/spring-beans.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context.xsd">
11
12     <bean
13         class="org.springframework.jdbc.datasource.DriverManagerDataSource"
14         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
15         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
16         p:password="root" />
17
18     <!-- LocalSessionFactoryBean 설정 -->
19 </beans>

```





public class LocalSessionFactoryBean extends HibernateLocalSessionFactoryBean implements FactoryBean<SessionFactory>,



config.xml Product.java

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:c="http://www.springframework.org/schema/c"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans
8         http://www.springframework.org/schema/beans/spring-beans.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context.xsd">
11
12   <bean
13     class="org.springframework.jdbc.datasource.DriverManagerDataSource"
14     name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
15     p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
16     p:password="root" />
17
18   <bean
19     class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
20     name="sessionFactory" p:dataSource-ref="dataSource">
21     <property name="hibernateProperties"> ①
22       <props>
23         <prop key="></prop> ②
24         <prop key="></prop> ②
25       </props>
26     </property>
27     <property name="annotatedClasses"> ③
28       <list>
29         <value></value>
30       </list>
31     </property>
32   </bean>
33
34 </beans>

```

LocalSessionFactoryBean

dataSource

hibernateProperties

annotatedClasses

- 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 Product.java MySQLDialect.class

```

1 package com.springorm.product.entity;
2
3 import javax.persistence.Column;
4
5 @DynamicUpdate
6 @Table(name = "product")
7 public class Product {

```

Open Type

Enter type name prefix or pattern (\*, ?, or ca

mysqldial

Matching items:

- MySQLDialect - org.hibernate.dialect -
- MySQLDialect - org.hibernate.dialect -

```

20 * Hibernate: Relational Persistence for Idioma
7 package org.hibernate.dialect;
@SuppressWarnings("deprecation")
public class MySQLDialect extends Dialect {

```

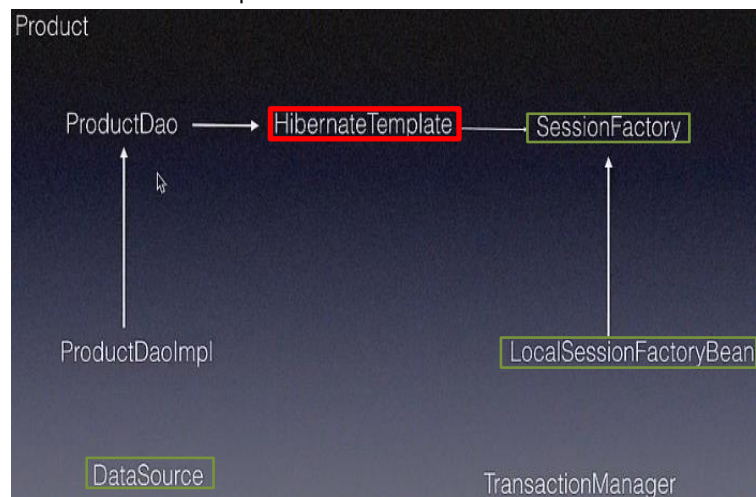
config.xml

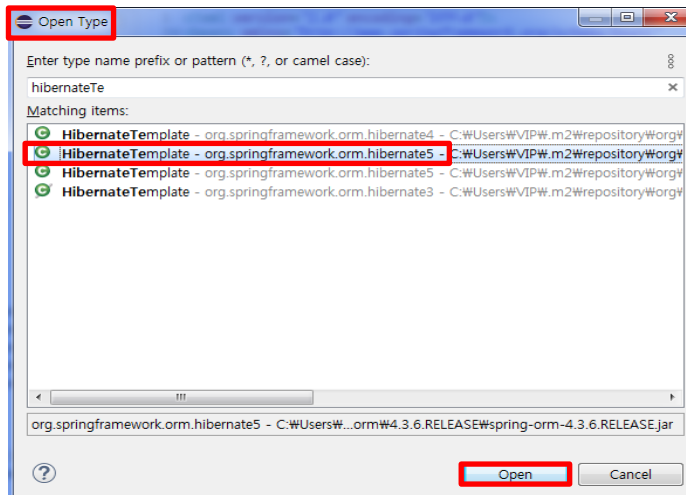
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:c="http://www.springframework.org/schema/c"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans
8         http://www.springframework.org/schema/beans/spring-beans.xsd
9         http://www.springframework.org/schema/context
10        http://www.springframework.org/schema/context/spring-context.xsd">
11
12     <bean
13         class="org.springframework.jdbc.datasource.DriverManagerDataSource"
14         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
15         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
16         p:password="root" />
17
18     <bean
19         class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
20         name="sessionFactory" p:dataSource-ref="dataSource">
21         <property name="hibernateProperties">
22             <props>
23                 <prop key="hibernate.dialect">
24                     org.hibernate.dialect.MySQLDialect
25                 </prop>
26                 <prop key="hibernate.show_sql">true</prop>
27             </props>
28         </property>
29         <property name="annotatedClasses">
30             <list>
31                 <value>com.springorm.product.entity.Product</value>
32             </list>
33         </property>
34     </bean>
35 </beans>

```

## 10. Hibernate Template 을 설정하고 사용하기





```
config.xml  HibernateTemplate.class
20 * Copyright 2002-2016 the original author or
16
17 package org.springframework.orm.hibernate5;
18
public class HibernateTemplate implements HibernateOperations,
```

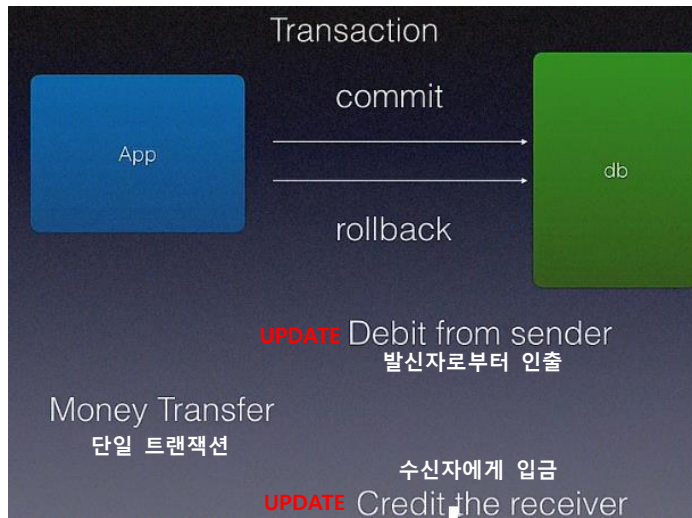
```
config.xml  HibernateTemplate.class
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:c="http://www.springframework.org/schema/c"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans
8       http://www.springframework.org/schema/beans/spring-beans.xsd
9       http://www.springframework.org/schema/context
10      http://www.springframework.org/schema/context/spring-context.xsd">
11
12     <bean
13         class="org.springframework.jdbc.datasource.DriverManagerDataSource"
14         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
15         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
16         p:password="root" />
17
18     <bean
19         class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
20         name="sessionFactory" p:dataSource-ref="dataSource">
21         <property name="hibernateProperties">
22             <props>
23                 <prop key="hibernate.dialect">
24                     org.hibernate.dialect.MySQLDialect
25                 </prop>
26                 <prop key="hibernate.show_sql">true</prop>
27             </props>
28         </property>
29         <property name="annotatedClasses">
30             <list>
31                 <value>com.springorm.product.entity.Product</value>
32             </list>
33         </property>
34     </bean>
35
36     <bean
37         class="org.springframework.orm.hibernate5.HibernateTemplate"
38         name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />
39
40 </beans>
```

```
config.xml  ProductDaoImpl.java  config.xml  ProductDaoImpl.java
1 package com.springorm.product.dao.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.orm.hibernate5.HibernateTemplate;
5 import org.springframework.stereotype.Component;
6
7 import com.springorm.product.dao.ProductDao;
8 import com.springorm.product.entity.Product;
9
10 @Component
11 public class ProductDaoImpl implements ProductDao {
12
13     @Autowired
14     HibernateTemplate hibernateTemplate;
15
16     @Override
17     public int create(Product product) {
18
19         return 0;
20     }
21 }
```

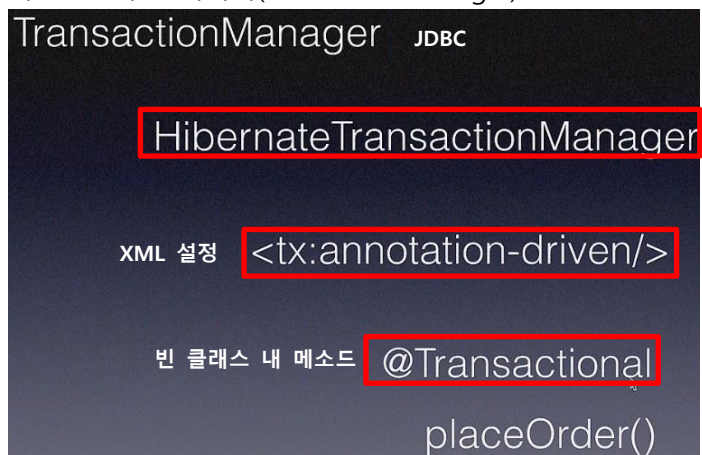
```
config.xml  ProductDaoImpl.java
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:c="http://www.springframework.org/schema/c"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans
8       http://www.springframework.org/schema/beans/spring-beans.xsd
9       http://www.springframework.org/schema/context
10      http://www.springframework.org/schema/context/spring-context.xsd">
11
12     <context:component-scan base-package="com.springorm.product.dao.impl">
13
14     <bean
15         class="org.springframework.jdbc.datasource.DriverManagerDataSource"
16         name="dataSource" p:driverClassName="com.mysql.jdbc.Driver"
17         p:url="jdbc:mysql://localhost/mykhtdb" p:username="root"
18         p:password="root" />
19
20     <bean
21         class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"
22         name="sessionFactory" p:dataSource-ref="dataSource">
23         <property name="hibernateProperties">
24             <props>
25                 <prop key="hibernate.dialect">
26                     org.hibernate.dialect.MySQLDialect
27                 </prop>
28                 <prop key="hibernate.show_sql">true</prop>
29             </props>
30         </property>
31         <property name="annotatedClasses">
32             <list>
33                 <value>com.springorm.product.entity.Product</value>
34             </list>
35         </property>
36     </bean>
37
38     <bean
39         class="org.springframework.orm.hibernate5.HibernateTemplate"
40         name="hibernateTemplate" p:sessionFactory-ref="sessionFactory" />
41
42 </beans>
```

## 11. 트랜잭션 매니저(Transaction Manager)

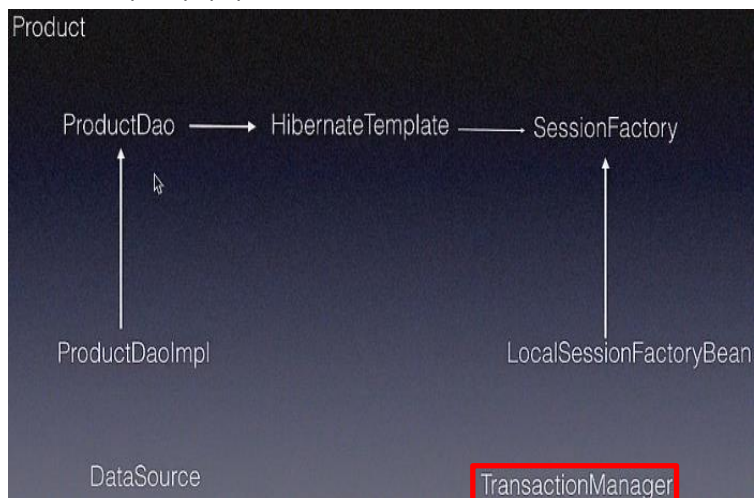
### 가. 개요



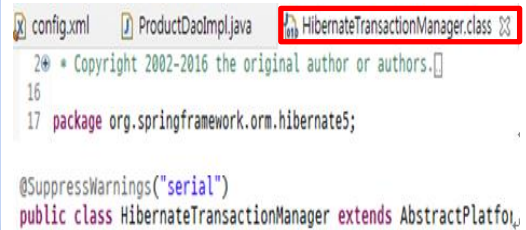
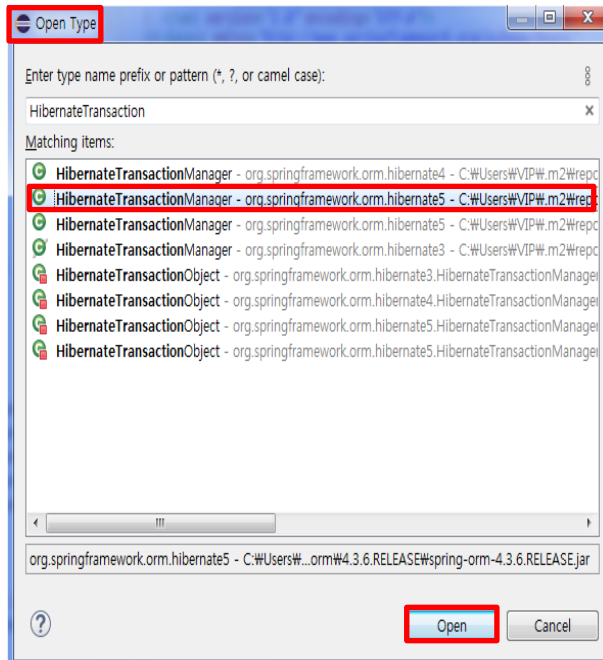
### 나. 트랜잭션 매니저(TransactionManager)



## 12. 트랜잭션매니저 설정





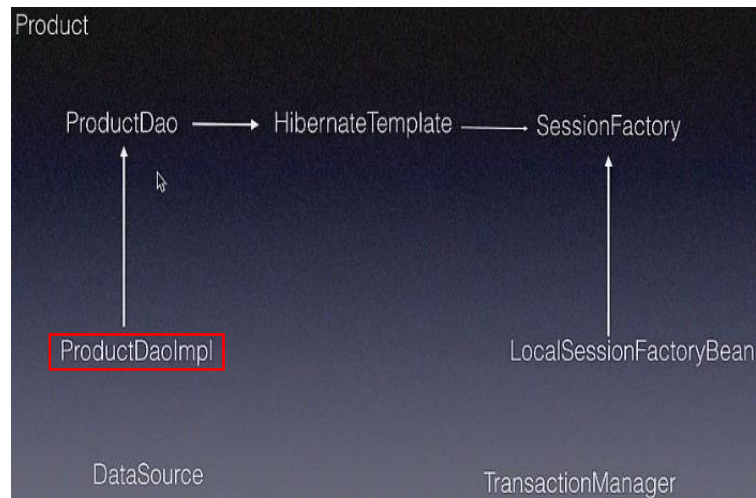


스프링 부트는 대부분 설정을 애노테이션을 사용하지만, 스프링 레거시의 경우 가급적 애노테이션을 사용하지만, 직접 만들지 않는 클래스는 직접 애노테이션을 사용할 없다는 점이 불편하다. 따라서, 애노테이션과 XML 설정을 혼용할 수 밖에 없다.

### 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

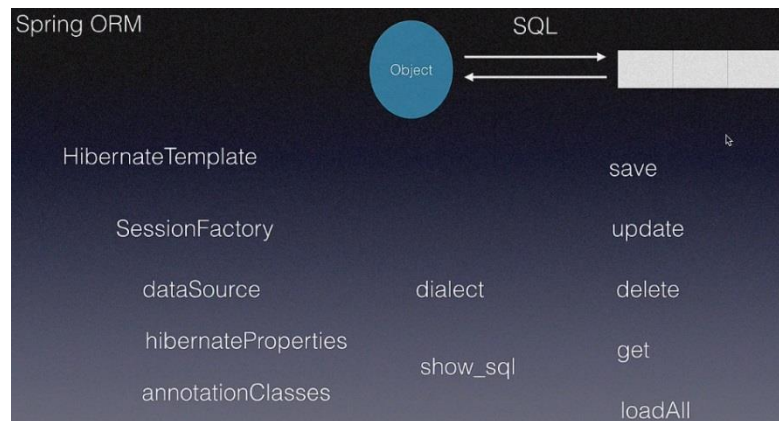
### 17. 단일 레코드 가져오기

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

SessionFactory

DataSource

save

update

delete

get

loadAll