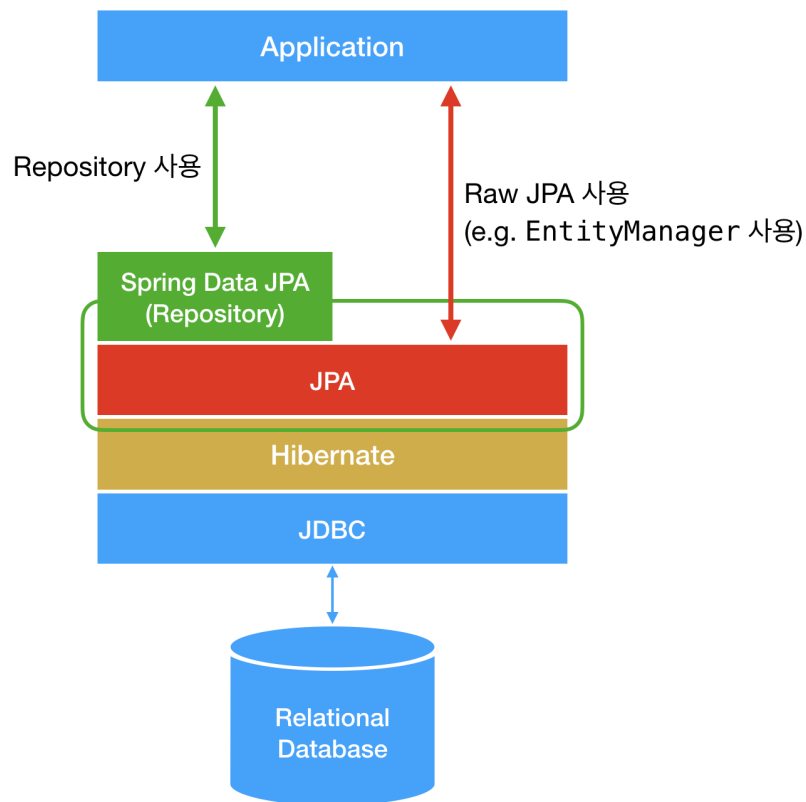


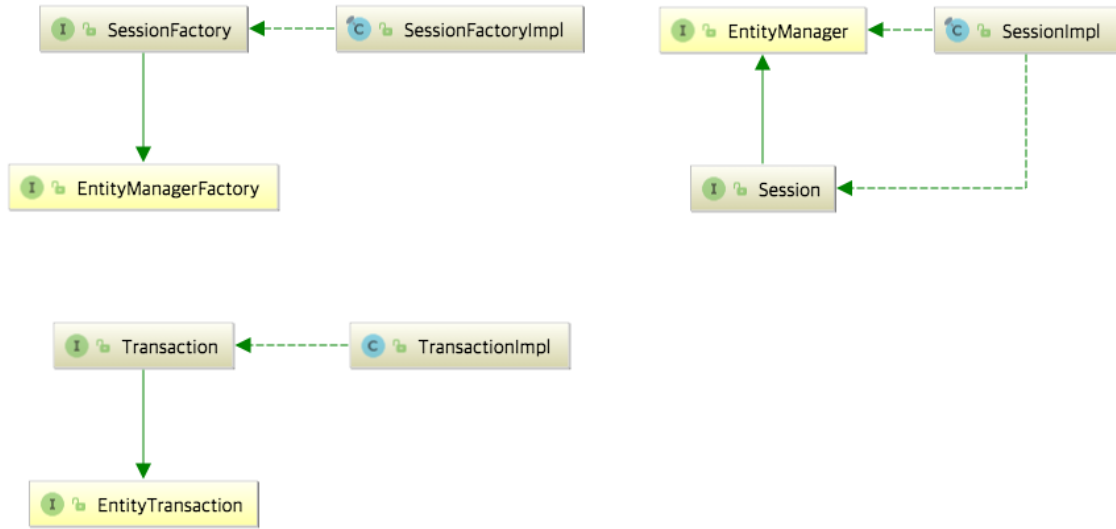
Working with SQL Databases

- 스프링 프레임워크는 SQL database 를 지원한다. `JdbcTemplate` 을 이용해 직접적으로 JDBC 에 접근하거나 `Hibernate` 같은 ORM 기술을 사용할 수 있다.
- Spring Data 는 추가적인 추상화를 지원해준다. `Repository` 인터페이스에서 직접 구현부를 만들어주고 메소드 이름으로 관습적인 쿼리를 만들어주기도 한다

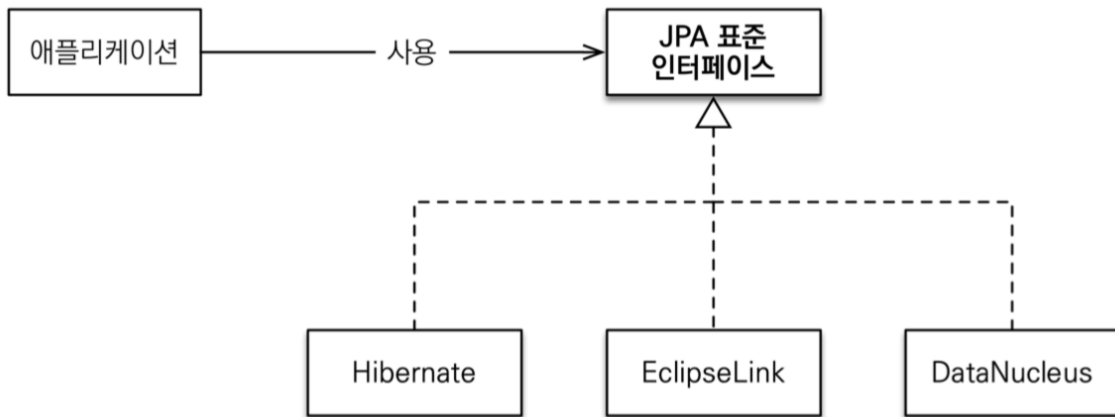
Hibernate, JPA, Spring Data JPA



- JPA
 - Java Persistence API
 - 기술 명세, 인터페이스!
 - `javax.persistence` 패키지에서 JPA 를 정의하고 있음
 - `EntityManagerFactory`, `EntityManager`, `EntityTransaction`
- Hibernate
 - JPA 명세의 구현체
 - `SessionFactory`, `Session`, `Transaction` 클래스가 JPA 인터페이스를 으로 상속받고 Impl 로 구현



- JPA 구현체는 Hibernate 말고도 EclipseLink, DataNucleus 등 여러개가 있다.



- Spring Data JPA
 - JPA 를 한단계 더 추상화 한 `Repository` 라는 인터페이스를 제공
 - `Repository` 인터페이스에 정해진 규칙대로 메서드를 입력하면 Spring 이 알아서 해당 메소드 이름에 적합한 쿼리를 날리는 구현체를 만들어 Bean 으로 등록한다
 - `SimpleJpaRepository` 코드

```

package org.springframework.data.jpa.repository.support;

import ...

public class SimpleJpaRepository<T, ID> implements JpaRepositoryImplementation<T, ID> {

    private final EntityManager em;
  
```

```

public Optional<T> findById(ID id) {

    Assert.notNull(id, ID_MUST_NOT_BE_NULL);

    Class<T> domainType = getDomainClass();

    if (metadata == null) {
        return Optional.ofNullable(em.find(domainType, id));
    }

    LockModeType type = metadata.getLockModeType();

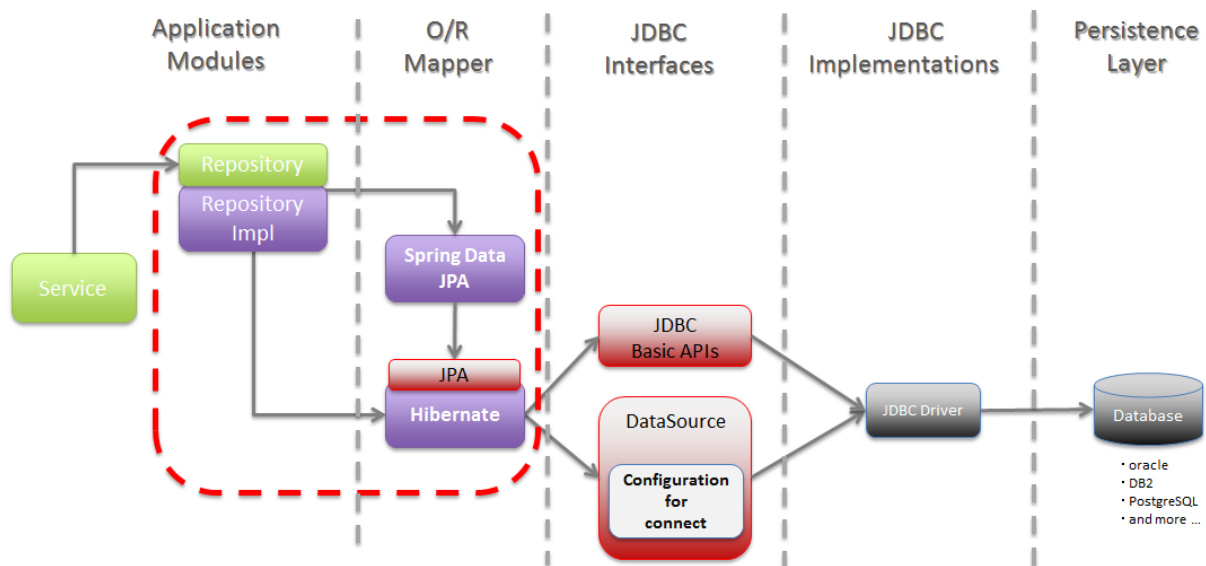
    Map<String, Object> hints = getQueryHints().withFetchGraphs(em).asMap();

    return Optional.ofNullable(type == null ? em.find(domainType, id, hints) : em.find(domainType, id, type, hints));
}

// Other methods...
}

```

DataSource, JDBC, JDBC Template



- DataSource
 - JDBC 명세의 일부분
 - DB 연결 팩토리
 - DB 서버와 연결, DB Connection Pooling, 트랜잭션 처리
 - url, driver, username, password 같은 프로퍼티를 사용해서 DB 서버와 연결한다
 - 데이터베이스에 연결하여 Connection 객체를 얻는 과정은 시간이 많이 걸리는 작업. 일정량의 Connection 을 미리 생성시켜 저장소에 저장했다가 프로그램 요청이 있을 때 저장소에서 Connection 을 꺼내 제공하면 시간을 절약할 수 있다. 이런 기법을 **Connection Pooling** 이라고 한다
- JDBC

- Java Database Connectivity
 - 모든 Java의 Data Access 기술의 근간으로 자바에서 제공하는 API
 - 데이터베이스에서 데이터를 쿼리하거나 업데이트 하는 방법을 제공
- JDBC Template
 - 스프링에서 제공하는 Plain JDBC API 의 문제점을 보완한 형태의 클래스

11.1 Configure a DataSource

- 자바의 `javax.sql.DataSource` 인터페이스는 DB connection 에 사용되는 표준 메소드를 제공한다
- 전통적으로 `DataSource` 는 `URL` 과 `credential` 을 사용하여 DB connection 을 맺는다

11.2 Using JdbcTemplate

- 스프링의 `JdbcTemplate` 과 `NamedParameterJdbcTemplate` 클래스는 자동설정된다.
- 이들을 빈에 주입하여 사용할 수 있다.

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

@Component
public class MyBean {

    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public MyBean(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    // ...

}
```

- `spring.jdbc.template.*` 를 사용해서 프로퍼티를 커스터마이징 할 수 있다.
- 여러개의 DataSource 빈을 생성할 수도 있다.



`NamedParameterJdbcTemplate` 은 `JdbcTemplate` 인스턴스를 재활용한다. 만약 하나 이상의 `JdbcTemplate` 이 정의되어 있고 주요 후보가 없다면 `NamedParameterJdbcTemplate` 은 자동으로 설정되지 않는다

11.3 JPA and Spring Data JPA

- 자바 영속성 API 는 표준 기술로써 이를 이용해 객체를 관계형 데이터베이스와 매핑할 수 있다.
- `spring-boot-starter-data-jpa` 를 사용해 손쉽게 시작해 볼 수 있다.

- Hibernate: 가장 널리 사용되는 JPA 구현체 중 하나
- Spring Data JPA: JPA 를 추상화한 레파지토리의 구현체를 만들어준다
- Spring ORM: 스프링 프레임워크가 제공하는 core ORM 기술

11.4 Spring Data JDBC

- Spring Data 는 JDBC 를 내부적으로 사용하는 repository 를 포함하고 있으며 `CrudRepository` 에 있는 메서드들로부터 SQL 문을 자동으로 생성해준다. `@Query` 어노테이션을 사용할 수도 있다.
- 스프링 부트는 어플리케이션 클래스패스에 필요한 의존성이 추가되어 있다면 Spring Data JDBC repository 를 자동설정 한다. 간단히 `spring-boot-starter-data-jdbc` 를 추가하여 사용할 수 있다.
- 필요하다면 Spring Data JDBC 설정을 커스터마이징 할 수 있다.
 - `@EnableJdbcRepositories`
 - `JdbcConfiguration` 의 자식 클래스 구현하기

11.5 Using H2's Web Console

- H2 데이터베이스는 웹 기반 콘솔을 제공한다
- Spring Boot 는 다음 조건이 만족되면 H2 데이터베이스를 자동설정 한다
 - 서블릿 기반의 웹 어플리케이션
 - `com.h2database:h2` 가 클래스패스에 위치
 - 스프링 부트 developer tool 사용중일 때


만약 사용중이 아니라도 `spring.h2.console.enabled` 프로퍼티를 true 로 하면 자동설정을 사용할 수 있다. 단 H2 콘솔은 개발 단계에서만 사용하도록 되어져 있어서 production 레벨에서는 해당 프로퍼티가 true 로 설정 되어 있지 않도록 주의해야 한다

11.6 Using jOOQ

- Java Object Oriented Querying
- 자바 코드로 쿼리를 작성하게 해주는 데이터베이스 인터페이스
- type-safe한 SQL 쿼리를 작성할 수 있다.
- jOOQ 의 commercial 버전과 open source 버전 둘 다 스프링 부트에서 사용할 수 있다.
- DB 스키마에서 클래스를 생성하고, 그로부터 생성된 클래스의 쿼리를 작성할 수 있는 별도의 내부 도메인 특정 언어 (DSL) 을 제공한다
- `jOOQ vs Hibernate` (→ ORM vs SQL)

jOOQ vs. Hibernate: When to Choose Which

Hibernate has become a de-facto standard in the Java ecosystem, and after the fact, also an actual JavaEE standard implementation if standards matter to you, and if you put the JCP on the same level with ISO, ANSI, IEEE, etc. This article does not intend to discuss standards,

 <https://blog.jooq.org/2015/03/24/jooq-vs-hibernate-when-to-choose-which/>

vin King

entlich geteilt - 10.12.2013

you're using Hibernate, doesn't mean you h
A point I've been making for about ten yea

- `jOOQ vs QueryDSL`

QueryDSL vs. jOOQ. Feature Completeness vs. Now More Than Ever

This week, Timo Westkämper from QueryDSL has announced feature completeness on the QueryDSL user group, along with his call for contributions and increased focus on bugfixes and documentation. Timo and us, we have always been in close contact, observing each other's progress. <https://blog.jooq.org/2014/05/29/querydsl-vs-jooq-feature-completeness-vs-now-more-than-ever/>

Way to Wri

11.6.1 Code Generation

- jOOQ 는 type-safe 한 쿼리를 사용할 수 있도록 데이터베이스 스키마를 가져와 자바 클래스를 생성한다 (QueryDSL이나 JPA를 사용할때처럼 도메인과 DB간의 매핑을 시켜주면서 하는 작업이 아닌 역으로 DB 테이블 스키마를 도메인으로 만든다)
- DB 를 스캔하여 tables, records, sequences, POJOs, DAOs, sorted procedures, user-defined types 등의 class 를 생성할 수 있다.

11.6.2 Using DSLContext

- jOOQ 는 `org.jooq.DSLContext` 인터페이스를 통해 API를 제공한다
- 스프링 부트는 `DSLContext` 를 스프링 빈으로 자동설정하여 application 의 `DataSource` 와 연결한다
- 아래처럼 `DSLContext` 를 주입받아 사용할 수 있다.

```
@Component
public class JooqExample implements CommandLineRunner {

    private final DSLContext create;

    @Autowired
    public JooqExample(DSLContext dslContext) {
        this.create = dslContext;
    }

}
```

- 그러면 쿼리를 만들 때 아래처럼 사용할 수 있다.

```
public List<GregorianCalendar> authorsBornAfter1980() {
    return this.create.selectFrom(AUTHOR)
        .where(AUTHOR.DATE_OF_BIRTH.greaterThan(new GregorianCalendar(1980, 0, 1)))
        .fetch(AUTHOR.DATE_OF_BIRTH);
}
```


11.6.3 jOOQ SQL Dialect

- `spring.jooq.sql-dialect` 프로퍼티가 설정되지 않으면 스프링 부트가 사용중인 데이터베이스에 맞는 SQL 방언을 결정한다
- 만약 적당한 방언을 찾지 못하면 DEFAULT 를 사용한다
- 스프링 부트는 jOOQ 오픈 소스 버전에서 지원하는 방언만을 자동설정 해 줄 수 있다.

참고 자료

jOOQ !! (Java Object Oriented Querying)

8 min to read 사내에서 Query Repository로 사용하게 된 jOOQ 를 간단하게 정리해보겠습니다. Java Object Oriented Querying jOOQ는 자바 코드로 쿼리를 작성할 수 있는 데이터베이스 인터페이스 입니다. 데이터베이스 스키마에서 생성 된 클래스의 쿼리를 작성하는 내부 도메인 특정 언어를 제공하며 내부 도메인 특정

 <https://zuminternet.github.io/jOOQ/>

