



스프링 부트와 JPA

학습내용

- JPA(Java Persistence API) 개념
- JPA(Java Persistence API) 쿼크 스타트

학습목표

- JPA의 개념과 동작 원리를 설명할 수 있다.
- JPA를 이용하여 간단한 CRUD 기능을 구현할 수 있다.

JPA (Java Persistence API) 개념

1 JPA (Java Persistence API)

① JPA의 등장

- 데이터베이스 연동 기술

1 전통적인 JDBC

2 데이터 매퍼(Mapper) 기술 MyBATIS, Spring JDBC

3 대표적 ORM(Object Relational Mapping) 기술 Hibernate

1 JPA (Java Persistence API)

① JPA의 등장

ORM

- iBATIS, MyBATIS 등의 데이터 매퍼와 달리 데이터베이스 연동에 필요한 JAVA 코드, 실질적인 SQL 구문까지 제공
- 개발 및 유지보수성을 극적으로 향상시킴

JPA (Java Persistence API) 개념

1 JPA (Java Persistence API)

① JPA의 등장



1 JPA (Java Persistence API)

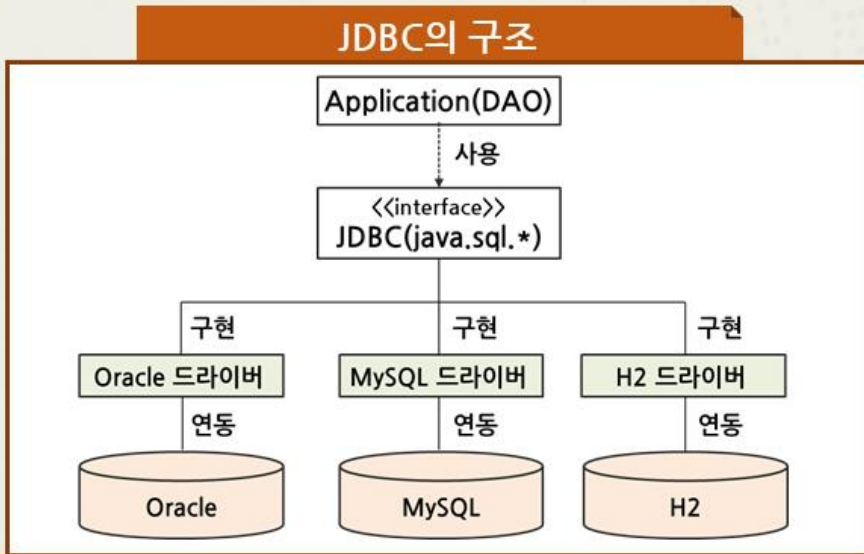
② JPA 개념

- JPA는 다형성을 기반으로 제공되는 **JDBC API와 동일한 개념**으로 이해할 수 있음

JPA (Java Persistence API) 개념

1 JPA (Java Persistence API)

② JPA 개념



1 JPA (Java Persistence API)

③ JPA 구현체

JPA를 이용한 데이터베이스 연동 처리

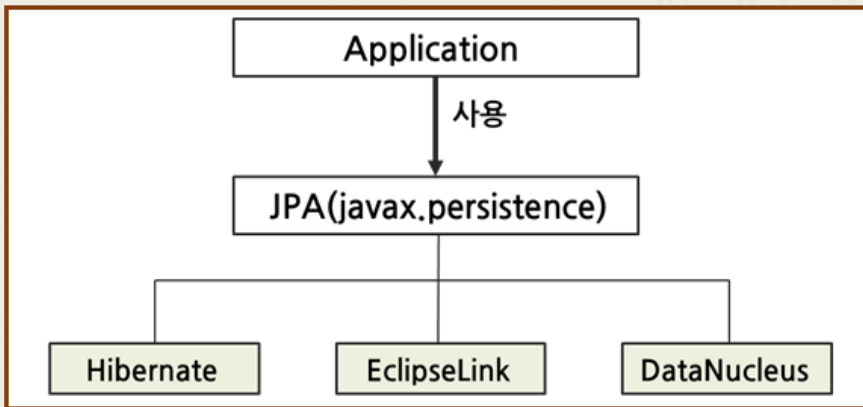
- 실제로는 JPA를 구현한 **구현체가 동작함**
- JPA는 모든 ORM 프레임워크를 일관된 방법으로 사용할 수 있도록 제공된 **확장 API**인 것

JPA (Java Persistence API) 개념

1 JPA (Java Persistence API)

③ JPA 구현체

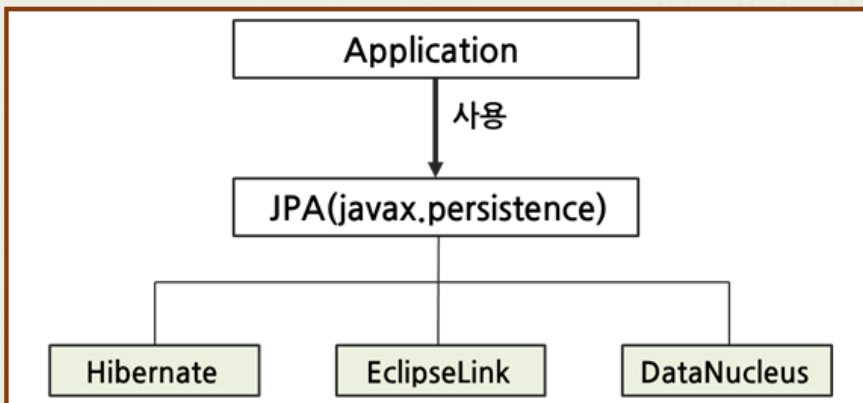
1 JPA를 구현한 구현체는 Hibernate, EclipseLink, DataNucleus 등 다양하게 존재함



1 JPA (Java Persistence API)

③ JPA 구현체

2 JPA를 이용하여 애플리케이션 개발 시 유지보수 과정에서 JPA 구현체를 쉽게 변경할 수 있음



JPA (Java Persistence API) 개념

1 JPA (Java Persistence API)

③ JPA 구현체

- 별도의 설정이 없으면 JPA는 Hibernate를 기본 구현체로 사용함

2 JPA 동작 원리

① JPA와 JDBC

JPA

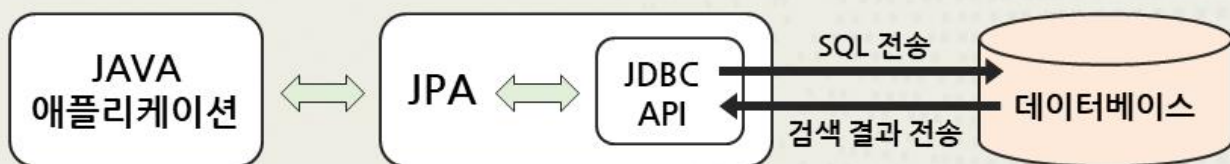
- JAVA 애플리케이션과 JDBC 사이에 존재하면서 JDBC의 복잡한 절차를 대신 처리함

JPA (Java Persistence API) 개념

2 JPA 동작 원리

① JPA와 JDBC

- JPA 내부적으로는 JDBC API를 이용하여 데이터베이스 연동을 처리함
- JPA가 일종의 프록시(대리자) 역할을 수행하는 것



2 JPA 동작 원리

② JPA와 SQL

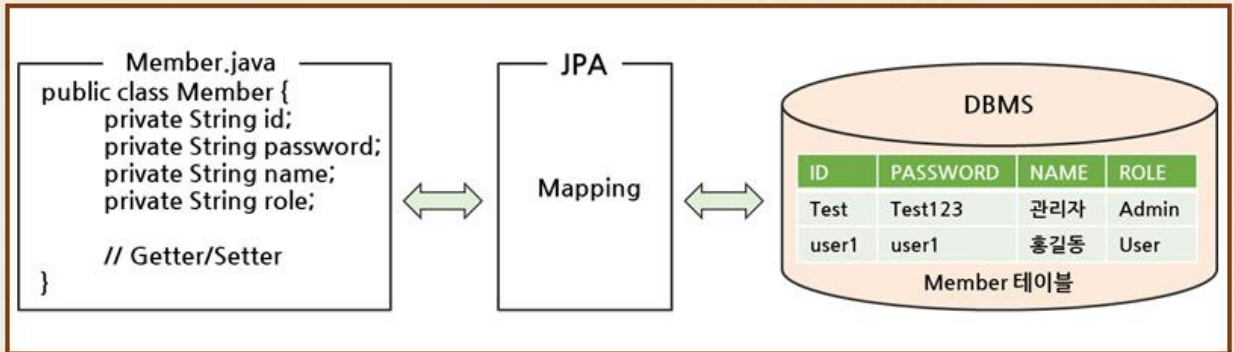
JPA

- 애플리케이션에서 생성한 JAVA 객체(Value Object)를 관계형 데이터베이스 테이블과 **자동으로 매핑**함
- JAVA 객체의 클래스 이름을 테이블과 매핑하고 멤버 변수는 컬럼과 매핑하면 **일관성 있는 SQL을 생성**할 수 있음

JPA (Java Persistence API) 개념

2 JPA 동작 원리

② JPA와 SQL



3 데이터베이스 설치

① H2 데이터베이스

H2

- 순수 JAVA로 만든 관계형 데이터베이스
- 용량이 작고 실행 속도가 빠른 오픈 소스 데이터베이스

JPA (Java Persistence API) 개념

3 데이터베이스 설치

① H2 데이터베이스



일반적인 JDBC 지원



인메모리(In-memory), 서버(Server) 모드까지 지원

3 데이터베이스 설치

① H2 데이터베이스

H2

- 브라우저 기반의 관리 콘솔 제공

스프링 부트

- 기본적으로 H2 지원

...> 테스트용 데이터베이스로 적합함

JPA (Java Persistence API) 개념

3 데이터베이스 설치

② H2 데이터베이스 다운로드

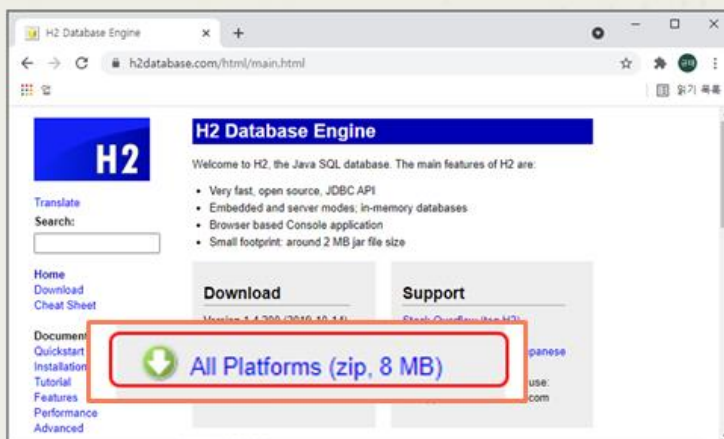
H2 홈페이지(<https://www.h2database.com>)

- 모든 플랫폼에서 사용할 수 있는 H2 데이터베이스를 다운로드할 수 있음

3 데이터베이스 설치

② H2 데이터베이스 다운로드

H2 홈페이지(<https://www.h2database.com>)

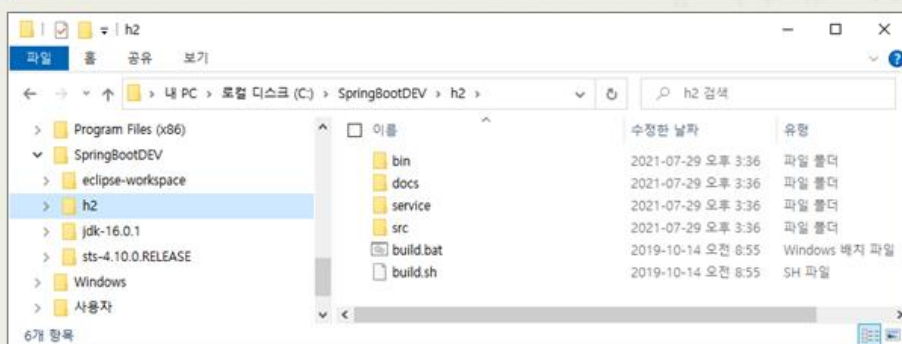


JPA (Java Persistence API) 개념

3 데이터베이스 설치

③ H2 데이터베이스 설치

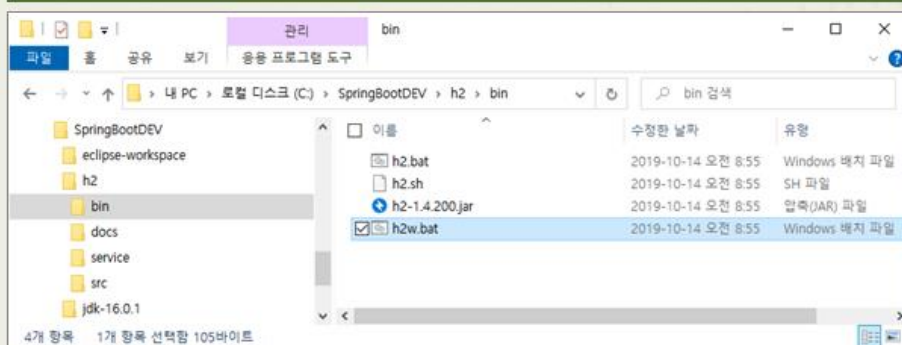
- 다운로드한 압축파일(h2-2019-10-14.zip)을 C:\WSpringBootDEV 폴더에 압축만 해제하면 설치가 완료됨



3 데이터베이스 설치

④ H2 데이터베이스 구동

- h2/bin 폴더에 있는 h2w.bat 파일을 실행하면 데이터베이스가 구동됨



JPA (Java Persistence API) 개념

3 데이터베이스 설치

⑤ H2 데이터베이스 연결

- H2 콘솔에서 드라이버 클래스, JDBC URL, 사용자명, 비밀번호를 설정하고 [연결] 버튼을 누르면 데이터베이스에 연결됨

로그인

저장한 설정: Generic H2 (Embedded)

설정 이름: Generic H2 (Embedded) 저장 삭제

드라이버 클래스: org.h2.Driver

JDBC URL: jdbc:h2:tcp://localhost/~:/test

사용자명: sa

비밀번호:

연결 연결 시험

3 데이터베이스 설치

⑤ H2 데이터베이스 연결

- H2 콘솔에서 드라이버 클래스, JDBC URL, 사용자명, 비밀번호를 설정하고 [연결] 버튼을 누르면 데이터베이스에 연결됨

로그인

저장한 설정: Generic H2 (Embedded)

설정 이름: Generic H2 (Embedded) 저장 삭제

드라이버 클래스: org.h2.Driver

JDBC URL: jdbc:h2:tcp://localhost/~:/test

사용자명: sa

비밀번호:

연결 연결 시험

JPA (Java Persistence API) 개념

3 데이터베이스 설치

⑤ H2 데이터베이스 연결

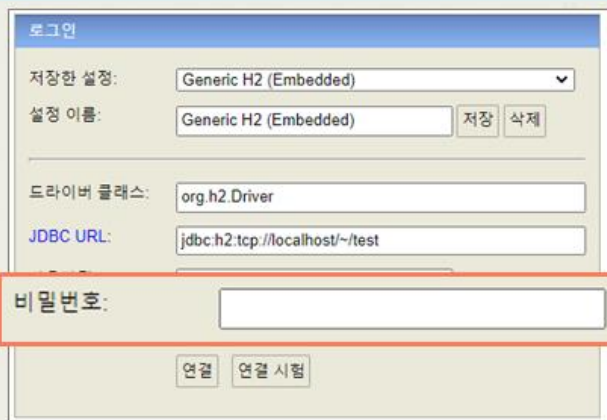
- H2 콘솔에서 드라이버 클래스, JDBC URL, 사용자명, 비밀번호를 설정하고 [연결] 버튼을 누르면 데이터베이스에 연결됨



3 데이터베이스 설치

⑤ H2 데이터베이스 연결

- H2 콘솔에서 드라이버 클래스, JDBC URL, 사용자명, 비밀번호를 설정하고 [연결] 버튼을 누르면 데이터베이스에 연결됨

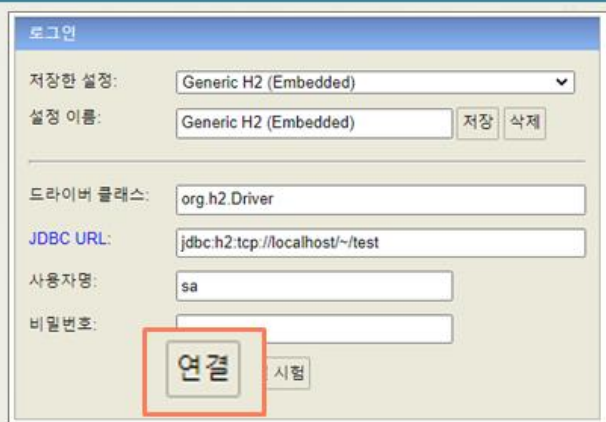


JPA (Java Persistence API) 개념

3 데이터베이스 설치

5 H2 데이터베이스 연결

- H2 콘솔에서 드라이버 클래스, JDBC URL, 사용자명, 비밀번호를 설정하고 [연결] 버튼을 누르면 데이터베이스에 연결됨



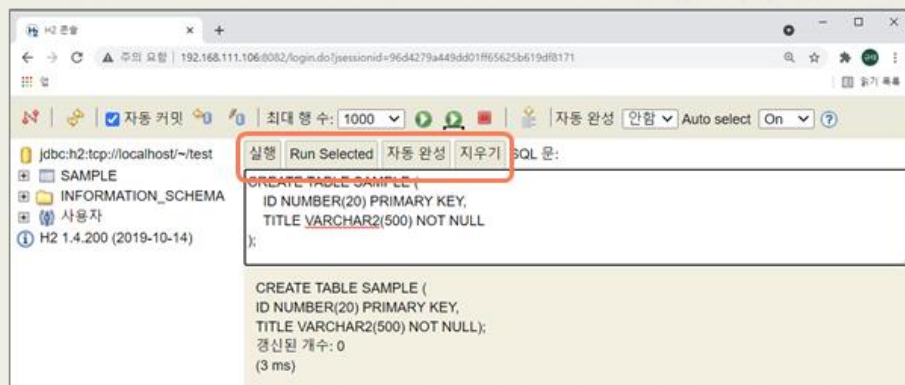
The image shows the '로그인' (Login) dialog box in the H2 Console. It contains the following fields and buttons:

- 저장한 설정:** A dropdown menu showing 'Generic H2 (Embedded)'.
- 설정 이름:** A text field containing 'Generic H2 (Embedded)', with '저장' (Save) and '삭제' (Delete) buttons next to it.
- 드라이버 클래스:** A text field containing 'org.h2.Driver'.
- JDBC URL:** A text field containing 'jdbc:h2:tcp://localhost/~test'.
- 사용자명:** A text field containing 'sa'.
- 비밀번호:** A text field for the password.
- Buttons:** '연결' (Connect) and '시험' (Test) buttons at the bottom. The '연결' button is highlighted with a red box.

3 데이터베이스 설치

6 SQL 작성 및 실행

- H2 콘솔에서 SQL을 작성하고 실행하기 위해서는 SQL 입력화면 위에 있는 버튼을 이용함



The image shows the H2 Console interface with a SQL query entered and executed. The '실행' (Execute) button is highlighted with a red box.

SQL 문:

```
CREATE TABLE SAMPLE (
  ID NUMBER(20) PRIMARY KEY,
  TITLE VARCHAR2(500) NOT NULL
);
```

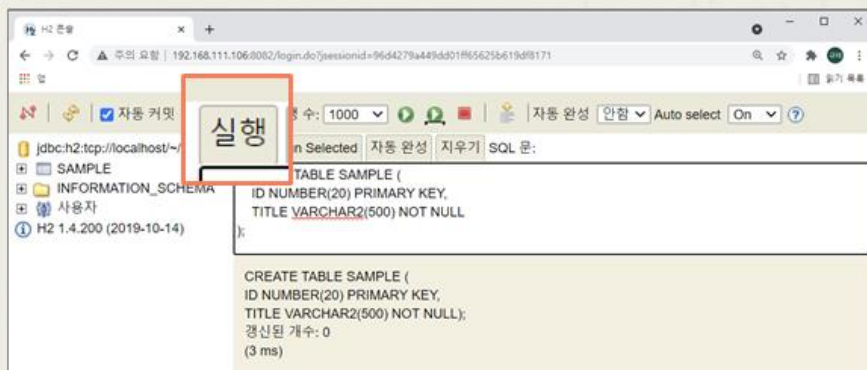
실행 결과:

```
CREATE TABLE SAMPLE (
  ID NUMBER(20) PRIMARY KEY,
  TITLE VARCHAR2(500) NOT NULL);
경산된 개수: 0
(3 ms)
```

JPA (Java Persistence API) 개념

3 데이터베이스 설치

⑥ SQL 작성 및 실행

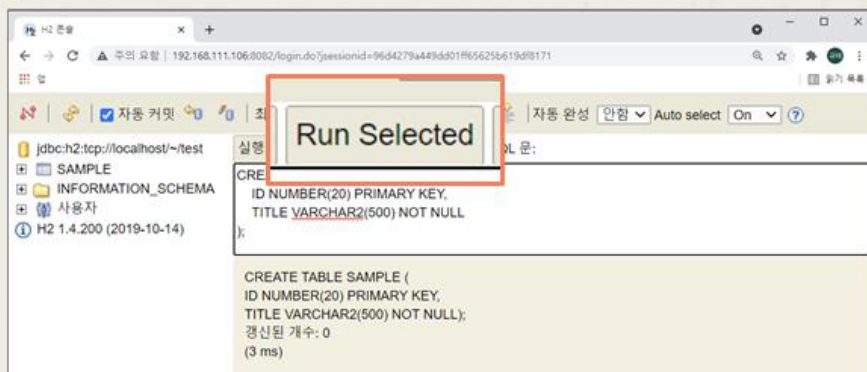


실행

• SQL 입력창에 작성된 SQL을 모두 실행함

3 데이터베이스 설치

⑥ SQL 작성 및 실행



Run Selected

• 선택된 SQL만 실행함

JPA (Java Persistence API) 개념

3 데이터베이스 설치

6 SQL 작성 및 실행

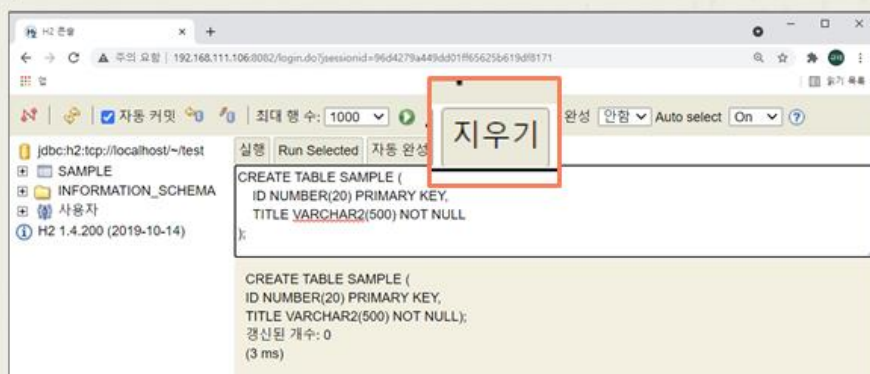


자동
완성

- SQL을 자동완성 시킴 (Ctrl + Space와 동일)

3 데이터베이스 설치

6 SQL 작성 및 실행



지우기

- SQL 입력창을 초기화함



JPA (Java Persistence API) 퀵 스타트

1 실습 프로젝트 생성 및 로그 설정

① 프로젝트 생성

스프링 부트 프로젝트 생성 시

- JAPQuickStart
- JPA 실습에 필요한 스타터와 라이브러리를 추가함

1 실습 프로젝트 생성 및 로그 설정

① 프로젝트 생성

Available:	Selected:
Type to search dependencies	
<ul style="list-style-type: none"> ▶ Developer Tools ▶ I/O ▶ Messaging ▶ Microsoft Azure ▶ NoSQL ▶ Observability ▶ Ops ▶ SQL ▶ Security 	<ul style="list-style-type: none"> X Spring Boot DevTools X Lombok X Spring Data JPA X H2 Database

H2
Database

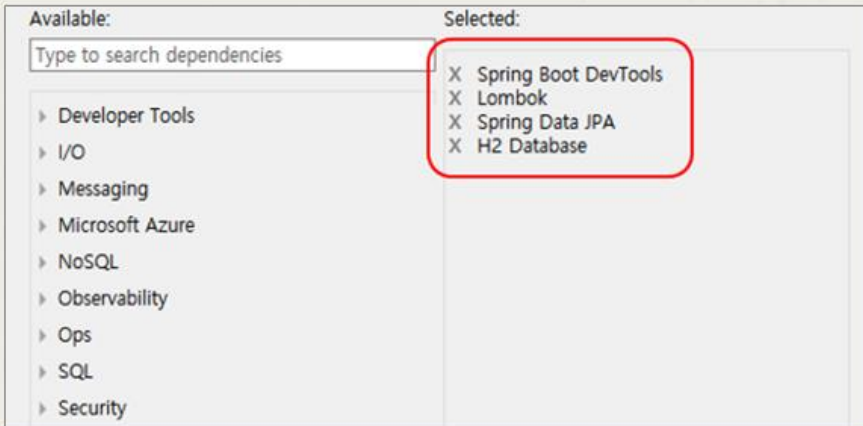
- H2 데이터베이스 JDBC 드라이버



JPA (Java Persistence API) 퀵 스타트

1 실습 프로젝트 생성 및 로그 설정

① 프로젝트 생성



Spring
Data JPA

• JPA 및 Hibernate(JPA 구현체)
라이브러리

1 실습 프로젝트 생성 및 로그 설정

② 기본 로그 설정 변경

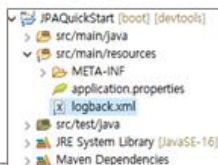
• src/main/resources 소스 폴더에 logback.xml 파일을 작성함

```
<?xml version="1.0" encoding="UTF-8"?>

<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
      <pattern>
        %d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n
      </pattern>
    </encoder>
  </appender>

  <logger name="org.hibernate" level="info" additivity="false">
    <appender-ref ref="STDOUT"/>
  </logger>

  <root level="warn">
    <appender-ref ref="STDOUT"/>
  </root>
</configuration>
```





JPA (Java Persistence API) 퀵 스타트

2 엔티티 클래스 작성 및 테이블 매핑

① 엔티티 클래스 작성

- 테이블과 매핑되는 클래스를 엔티티 클래스라 함
- JPA는 엔티티와 매핑된 테이블이 존재하지 않을 때 엔티티 클래스를 기준으로 테이블을 생성함

```
package com.mycompany.entity;

import java.util.Date;

@Entity
@Table(name = "BOARD")
@Data
public class Board {
    @Id
    @GeneratedValue
    private Long seq;
    private String title;
    private String writer;
    private String content;
    private Date createDate;
    private Long cnt;
}
```

2 엔티티 클래스 작성 및 테이블 매핑

① 엔티티 클래스 작성

- 엔티티 클래스의 멤버 변수는 private으로 선언함
- @Data를 설정하여 public Getter/Setter 메소드를 제너레이션 함

```
package com.mycompany.entity;

import java.util.Date;

@Entity
@Table(name = "BOARD")
@Data
public class Board {
    @Id
    @GeneratedValue
    private Long seq;
    private String title;
    private String writer;
    private String content;
    private Date createDate;
    private Long cnt;
}
```




JPA (Java Persistence API) 퀵 스타트

2 엔티티 클래스 작성 및 테이블 매핑

② 엔티티 매핑

- JPA가 제공하는 어노테이션을 이용하여 엔티티 클래스와 테이블을 매핑하며 각 어노테이션의 의미는 다음과 같음

어노테이션	의미
@Entity	Board 클래스를 엔티티 클래스로 설정함(※필수※)
@Table	엔티티 클래스와 테이블 이름이 다른 경우, name 속성으로 매핑함
@Id	테이블의 기본 키(PK)를 매핑함(※필수※)
@GeneratedValue	@Id가 선언된 필드에 자동으로 증가된 값을 할당함

3 Persistence 설정

① persistence.xml 설정

- JPA는 클래스 패스에 존재하는 persistence.xml 파일을 로딩하여 데이터베이스 연동에 필요한 정보들을 획득함



JPA (Java Persistence API) 퀵 스타트

3 Persistence 설정

① persistence.xml 설정

- persistence.xml 파일 영속성 유닛(Persistence Unit)을 설정하며, 영속성 유닛 설정에 JPA가 사용할 엔티티 클래스나 데이터베이스에 대한 정보들이 포함됨

3 Persistence 설정

① persistence.xml 설정

- src/main/resources 소스 폴더에 META-INF 폴더를 생성하고 META-INF 폴더에 다음과 같은 persistence.xml 파일을 작성함

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">

  <persistence-unit name="JPAUnit">
    <class>com.mycompany.entity.Board</class>

    <properties>
      <!-- 필수 속성 -->
      <property name="javax.persistence.jdbc.driver" value="org.h2.Driver" />
      <property name="javax.persistence.jdbc.user" value="sa" />
      <property name="javax.persistence.jdbc.password" value="" />
      <property name="javax.persistence.jdbc.url" value="jdbc:h2:tcp://localhost/~test" />
      <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect" />

      <!-- 옵션 -->
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.use_sql_comments" value="false" />
      <property name="hibernate.id.new_generator_mappings" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="create" />
    </properties>
  </persistence-unit>
</persistence>
```



JPA (Java Persistence API) 쿼크 스타트

3 Persistence 설정

1 persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">

  <persistence-unit name="JPATest">
    <class>com.mycompany.entity.Board</class>

    <properties>
      <!-- 필수 속성 -->
      <property name="javax.persistence.jdbc.driver" value="org.h2.Driver" />
      <property name="javax.persistence.jdbc.user" value="sa" />
      <property name="javax.persistence.jdbc.password" value="" />
      <property name="javax.persistence.jdbc.url" value="jdbc:h2:tcp://localhost/~test" />
      <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect" />

      <!-- 옵션 -->
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
      <property name="hibernate.use_sql_comments" value="false" />
      <property name="hibernate.id.new_generator_mappings" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="create" />
    </properties>
  </persistence-unit>
</persistence>
```

3 Persistence 설정

2 persistence.xml 설정의 의미

<persistence-unit name="JPATest">

- 영속성 유닛 설정의 루트 엘리먼트로서 다른 영속성 유닛과 식별하기 위한 유일한 이름을 name 속성 값으로 지정함



JPA (Java Persistence API) 퀵 스타트

3 Persistence 설정

② persistence.xml 설정의 의미

`<class>com.mycompany.entity.Board</class>`

- 테이블과 매핑된 엔티티 클래스 목록을 등록함

`<properties>`

- 데이터 소스, Dialect 클래스, JPA 구현체(Hibernate)와 관련된 설정들을 등록함

4 클라이언트 작성

① 등록 기능 실행

1

JPA 이용을 위해 가장 먼저 생성해야 하는 객체
→ EntityManager

- EntityManager는 EntityManagerFactory로부터 획득함

2

EntityManager.persist 메소드를 이용하여
등록 작업을 수행함



JPA (Java Persistence API) 퀵 스타트

4 클라이언트 작성

① 등록 기능 실행

3

데이터베이스 연동이 마무리되었으면
EntityManager와 EntityManagerFactory
객체를 순차적으로 종료함

4

글 등록 기능의 JPAClient 프로그램을 작성함

4 클라이언트 작성

① 등록 기능 실행

```
package com.mycompany;

import java.util.Date;

public class JPAClient {
    public static void main(String[] args) {
        // EntityManager 생성
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPATest");
        EntityManager em = emf.createEntityManager();
        try {
            Board board = new Board();
            board.setTitle("JPA 제목");
            board.setWriter("관리자");
            board.setContent("JPA 글 등록 잘 되네요.");
            board.setCreateDate(new Date());
            board.setCnt(0L);
            // 글 등록
            em.persist(board);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            em.close();
            emf.close();
        }
    }
}
```



JPA (Java Persistence API) 퀵 스타트

4 클라이언트 작성

② 트랜잭션 처리

- 등록/수정/삭제에 해당하는 SQL이 실제로 처리하기 위해서는 해당 작업이 반드시 트랜잭션 안에서 수행되어야 함

```
public class JPAClient {
    EntityManager em = emf.createEntityManager();
    EntityTransaction tx = em.getTransaction();

    try {
        tx.begin();
        Board board =
            board.setContent("JPA 글 등록 잘 되네요.");
        board.setCreateDate(new Date());
        board.setEnt(0L);
        // 글 등록
        em.persist(board);
    } catch (Exception e) {
        tx.rollback();
    } finally {
        tx.close();
    }
}
```

4 클라이언트 작성

③ 데이터 누적

- 현재 상태에서는 클라이언트 프로그램(JPAClient)을 실행할 때마다 매번 테이블이 새롭게 생성되어 데이터가 누적되지 않음



JPA (Java Persistence API) 퀵 스타트

4 클라이언트 작성

③ 데이터 누적

- persistence.xml 파일에서 hibernate.hbm2ddl.auto 설정을 create에서 update로 수정함

```
<!-- 옵션 -->
<property name="hibernate.show_sql" value="true" />
<property name="hibernate.format_sql" value="true" />
<property name="hibernate.cache.new_generator_mappings" value="true" />
<property name="hibernate.hbm2ddl.auto" value="update" />
</properties>
```

4 클라이언트 작성

④ 데이터 검색

- 데이터를 상세 조회할 때는 EntityManager.find 메소드를 사용함

```
package com.mycompany;

import javax.persistence.EntityManager;

public class JPAClient {
    public static void main(String[] args) {
        // EntityManager 생성
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPATest");
        EntityManager em = emf.createEntityManager();
        try {
            // 글 상세 조회
            Board searchBoard = em.find(Board.class, 1L);
            System.out.println("---> " + searchBoard.toString());
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            em.close();
            emf.close();
        }
    }
}
```



적용 스프링 부트

스프링 부트와 JPA

적용 스프링 부트

Spring Boot Application

H2

Translate

Search:

Home

Download

Cheat Sheet

Documentation

Quickstart

Installation

Tutorial

Features

Performance

Advanced

Reference

Commands

Functions

Aggregate • Window

Data Types

SQL Grammar

System Tables

Javadoc

PDF (1.5 MB)

Support

FAQ

Error Analyzer

H2 Database Engine

Welcome to H2, the Java SQL database. The main features of H2 are:

- Very fast, open source, JDBC API
- Embedded and server modes; in-memory databases
- Browser based Console application
- Small footprint: around 2 MB jar file size

Download

Version 1.4.200 (2019-10-14)

Windows Installer (5 MB)

All Platforms (zip, 8 MB)

All Downloads

Support

Stack Overflow (tag H2)

Google Group English, Japanese

For non-technical issues, use: dbsupport@h2database.com

Features

	H2	Derby	HSQLDB	MySQL	PostgreSQL
Pure Java	Yes	Yes	Yes	No	No
Memory Mode	Yes	Yes	Yes	No	No
Encrypted Database	Yes	Yes	Yes	No	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	No	No	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Footprint (embedded)	~2 MB	~3 MB	~1.5 MB	—	—
Footprint (client)	~500 KB	~600 KB	~1.5 MB	~1 MB	~700 KB

1. JPA 관련 설정하기

- Zulu(OpenJDK) 16 사용
- Spring Tools 4 for Eclipse 4.0.10 사용
- H2 Database 2.0.206 사용

실습단계

데이터베이스 설치, 순수 자바로 구현된 H2 데이터베이스

JAVA로 만들어진 H2 데이터베이스를 다운로드 받을 수 있음

압축만 해제하면 설치가 됨

h2w.bat 파일 더블 클릭 시 데이터베이스 구동

커넥션에 필요한 네 가지 정보를 적절히 설정

연결 버튼을 누르면 커넥션 연결 성공

JPA를 이용해 DB를 연동하기 때문에 SQL을 직접 작성하는 일은 없음

연결 끊기

중요! JDBC URL 변경, jdbc:h2:tcp://localhost/~ /test

DBMS와 연결된 상태에서 데이터베이스 이용



적용 스프링 부트

2. JPA를 이용하여 기능 테스트하기



JPA를 이용하여 등록 기능과 상세 조회 기능을 테스트함

실습단계
이클립스로 돌아와서 프로젝트 생성, New → Project
File → New → Spring Starter Project 선택
프로젝트 이름 : BoardWeb
Package는 그룹 ID로 똑같이 설정
Lombok, Spring Data JPA, H2 Database 선택
src/main/java → New → Class
Package : 끝에 domain 추가, Name : Board
컬럼으로 사용할 여섯 개의 변수 설정
Date : java.util
lombok 어노테이션 추가
식별자 필드임을 지정
유니크한 숫자 값이 자동으로 세팅 될 수 있도록 함
엔티티 클래스 작성 완료, 환경설정 파일 작성
src/main/resources → New → Other...



적용 스프링 부트

실습단계
General → Folder, META-INF
XML 파일에 persistence 관련 설정 추가
Tip! 화면을 잠깐 멈추고 소스를 타이핑하며 진행할 것
유닛 정보를 이용해 H2 데이터베이스와 연결
H2 데이터베이스에 최적화된 SQL을 제너레이션
Board 엔티티 클래스를 기준으로 매핑된 Board table이 없을 경우, 생성함
무조건 Board table을 만들어서, 새로 Creat 한 후에 사용할 것
src/main/resources → New → Other...
BoardServiceRunner 클래스 생성
Run 메소드 오버라이딩
EntityManagerFactory가 만들어짐
영속 컨테스트 또는 영속 컨테이너라고 불림
이를 활용해 글등록 작업 처리
메인 클래스 실행
@Service 어노테이션 작성
애플리케이션 실행
중요! BOARD라는 테이블이 없으면 엔티티 클래스를 기준으로 만들어 줌
Insert가 실행되지 않는 이유는 트랜잭션 안에서 insert 되어야 하기 때문
트랜잭션 획득
트랜잭션 start
트랜잭션 종료
데이터가 없음
중요! Insert SQL이 JPA에서 제너레이션 되는지 확인
있는 테이블을 재사용
상세정보 조회 시 트랜잭션 관련 코드는 필요 없음
화면의 코드 작성