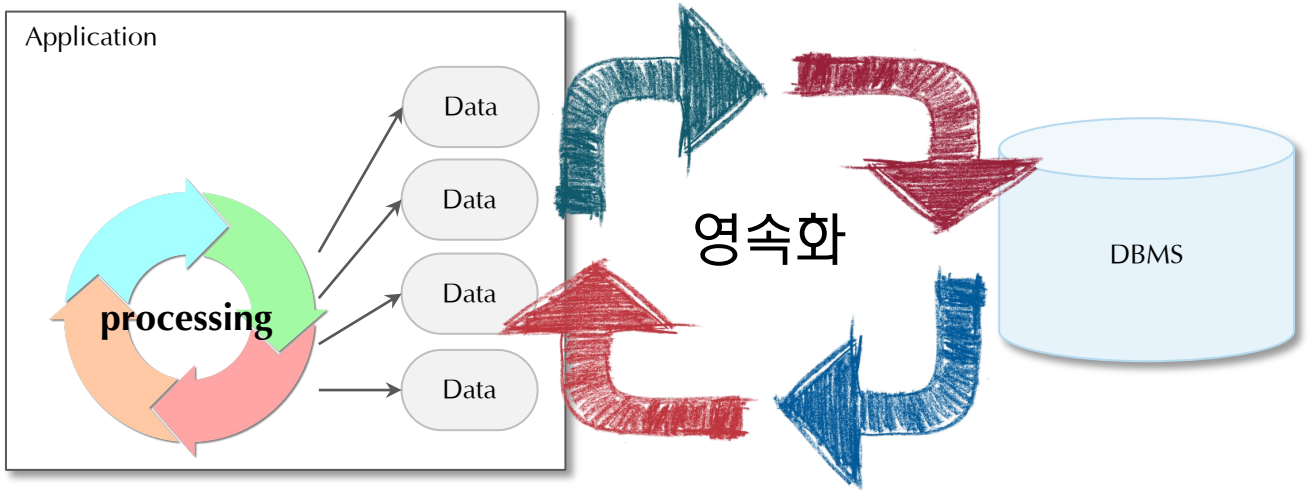
1. JPA란?

1.1 영속성에 대한 이해
1.2 자원(resource) 접근 레이어
1.3 객체의 세상과 테이블의 세상
1.4 JPA(Java Persistence API)의 이해
1.5 JPA 기초 실습

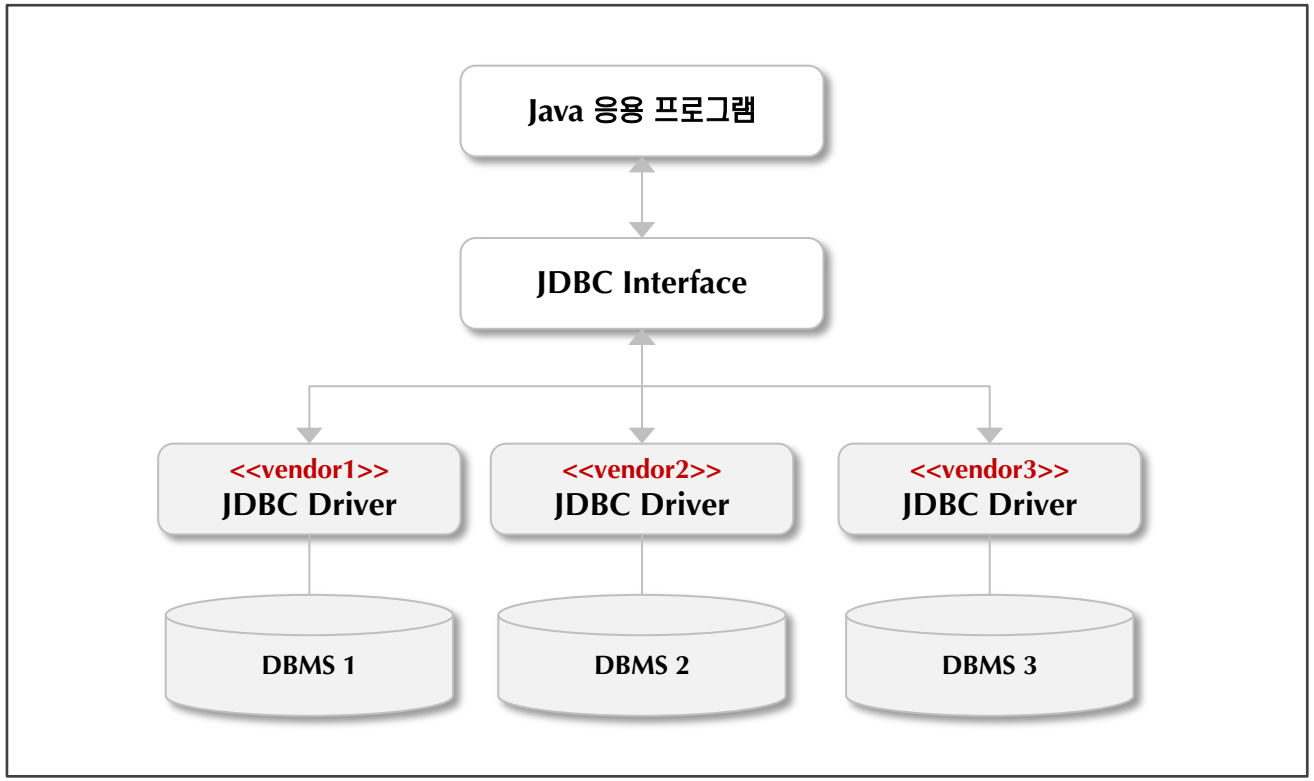
1.1 영속성에 대한 이해(1/3) - 개요

✓ 영속성은 사전적인 의미로 영원히 계속되는 성질이나 능력을 뜻합니다.
✓ 어플리케이션의 상태와 상관 없도록 물리적인 저장소를 이용해 데이터를 저장하는 행위를 영속화라 할 수 있습니다.
✓ 데이터를 어떤 공간에 어떤 형태로 저장할 것인지에 따라 영속화 방식은 달라질 수 있습니다.
✓ 보편적으로 적용되는 RDBMS에 데이터를 저장하기 위해서는 SQL을 이용해 데이터를 영속화 해야 합니다.



1.1 영속성에 대한 이해(2/3) - JDBC API(1/2)

- ✓ 자바 어플리케이션에서 데이터베이스에 접근 하는 방법은 기본적으로 JDBC 인터페이스를 통한 방법입니다.
- ✓ JDBC 인터페이스는 자바 어플리케이션과 데이터베이스의 소통을 위한 기능들을 정의하고 있습니다.
- ✓ 각 데이터베이스 제조사들은 JDBC 인터페이스를 구현하는 클래스들을 제공하며 이를 드라이버라고 합니다.
- ✓ 순수 JDBC 기반의 영속적 데이터 관리는 개발 과정에서 다소 많은 시간과 비용을 발생하게 합니다.



JDBC 인터페이스와 DBMS의 관계

1.1 영속성에 대한 이해(3/3) - JDBC API(2/2)

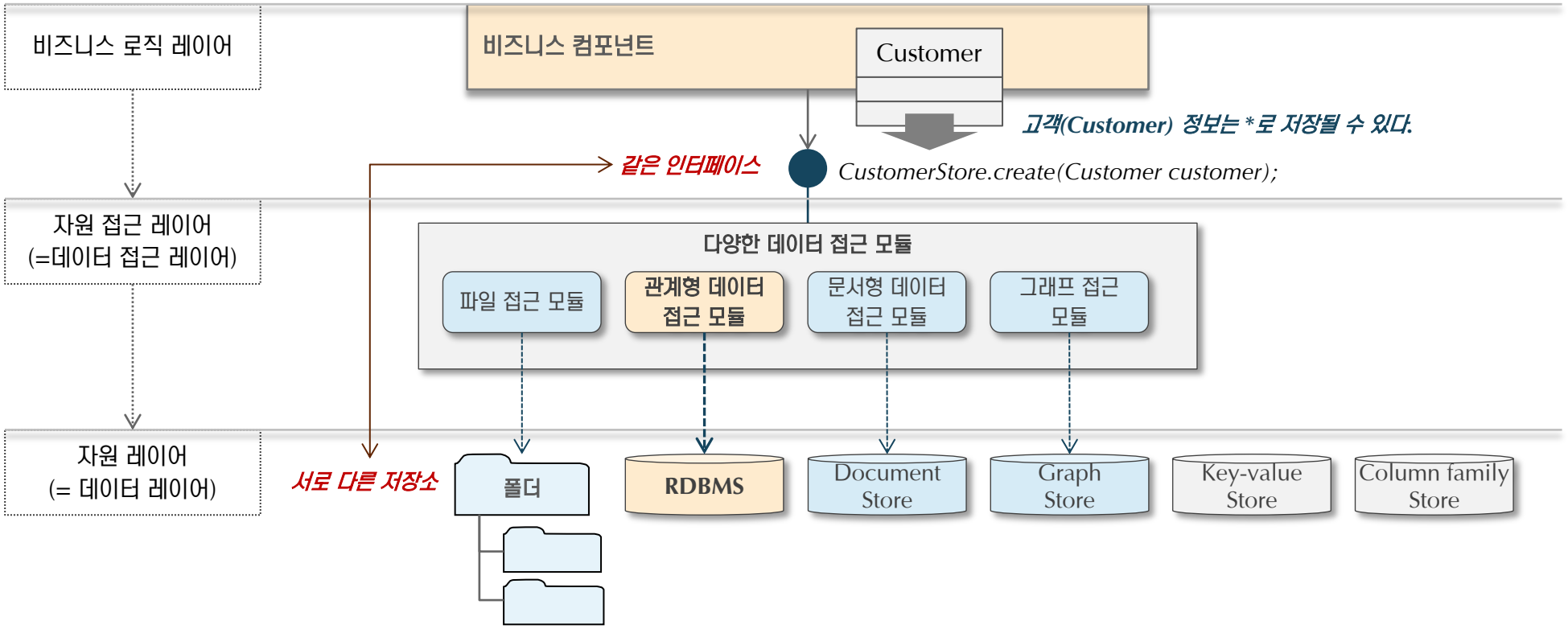
- ✓ DBMS의 데이터 관리는 SQL(Structured Query Language)을 통해 이루어집니다.
- ✓ 데이터베이스에 접속부터 질의(Query)를 보내고, 결과를 받는 것까지 전체 과정에서 자바와 데이터베이스간의 변환 절차가 필요합니다.
- ✓ 순수 JDBC를 적용하여 데이터를 관리할 경우 자바 코드와 SQL 코드를 동일한 파일에서 관리하게 됩니다.

```
public class UserJDBCProviderImpl implements UserProvider {
private final String DB_URL = "jdbc:hsqldb:mem:test";
private final String USER_ID = "sa";
private final String PASSWORD = "";

    public UserJDBCProviderImpl() {
        try { Class.forName("org.hsqldb.jdbcDriver"); }
        catch (ClassNotFoundException e) {
        throw new RuntimeException("JDBC 드라이버 로드 중 에러 발생"); } }

    public List<User> findAllUsers() {
        List<User> users = new ArrayList<User>();
        String sql = "SELECT USERID, NAME, EMAIL, TYPE "
               + "FROM USER_TB " + "ORDER BY USERID ASC";
        Connection conn = null;
        PreparedStatement psmt = null;
        ResultSet rs = null;
        try {
            conn = DriverManager.getConnection(DB_URL, USER_ID,  PASSWORD);
            psmt = conn.prepareStatement(sql);
            rs = psmt.executeQuery();
```

```
            while (rs.next()) {
                User user = new User();
                user.setUserId(rs.getString(1));
                user.setName(rs.getString(2));
                user.setEmail(rs.getString(3));
                user.setType(rs.getString(4));
                users.add(user);
            }
        } catch (SQLException e) {
            throw new RuntimeException("전체 사용자 조회중 오류 발생", e);
        } finally {
            try {
                if (rs != null) rs.close();
                if (psmt != null) psmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
            }
        }
        return users;
    }
}
```

1.2 자원(resource) 접근 레이어

✓ 비즈니스 컴포넌트는 비즈니스 로직 레이어에, 자원 접근 모듈은 자원 접근 레이어에 놓여 있습니다.
✓ 이들 간의 관계를 생각할 때, 좌측에 표현한 레이어 관점에서 볼 수 있습니다.
✓ 어떤 객체의 정보(예, Customer)가 자원 접근 레이어를 통해서 저장될 때는 하나의 인터페이스를 타고 들어갑니다.
✓ 하지만, 실제로 그 데이터가 저장되는 경로는 자원 접근 레이어에서 어떤 모듈이 처리하는가에 따라 다릅니다.



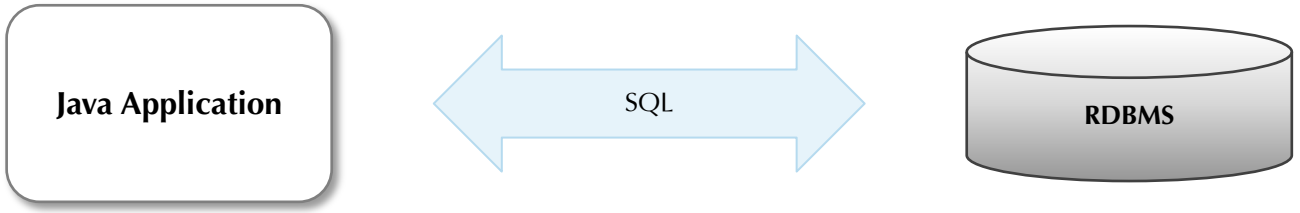
1.3 객체의 세상과 테이블의 세상(1/3)

✓ 객체 중심의 객체지향 어플리케이션과 테이블 중심의 관계형 데이터베이스는 서로의 목표가 다릅니다.
✓ 이를 패러다임이 일치하지 않는다고 표현하며 이는 개발 과정에서 많은 비용을 발생하게 합니다.
✓ 영속적으로 데이터를 저장하기 위해 객체의 세상과 테이블의 세상을 서로 맞추는 과정이 필요합니다.

객체지향 모델	관계형 모델
객체, 클래스	테이블, 로우
속성(attribute, property)	컬럼
Identity	Primary Key
Relationship/다른 엔티티 참조	Foreign Key
상속/다형성	없음
메소드	SQL 로직, SP, 트리거
코드의 이식성	벤더 종속적임

1.3 객체의 세상과 테이블의 세상(2/3)

- ✓ 자바 어플리케이션에서 관계형 데이터베이스의 사용을 돕는 프레임워크를 Persistence Framework라 합니다.
- ✓ Persistence Framework는 SQL Mapping과 OR Mapping으로 구분합니다.
- ✓ SQL Mapping은 자바 코드와 SQL을 분리하며 개발자가 작성한 SQL의 수행 결과를 객체로 매핑합니다.
- ✓ OR Mapping은 객체와 관계형 데이터베이스 사이에서 매핑을 담당하며 SQL을 생성하여 패러다임의 불일치를 해결합니다.



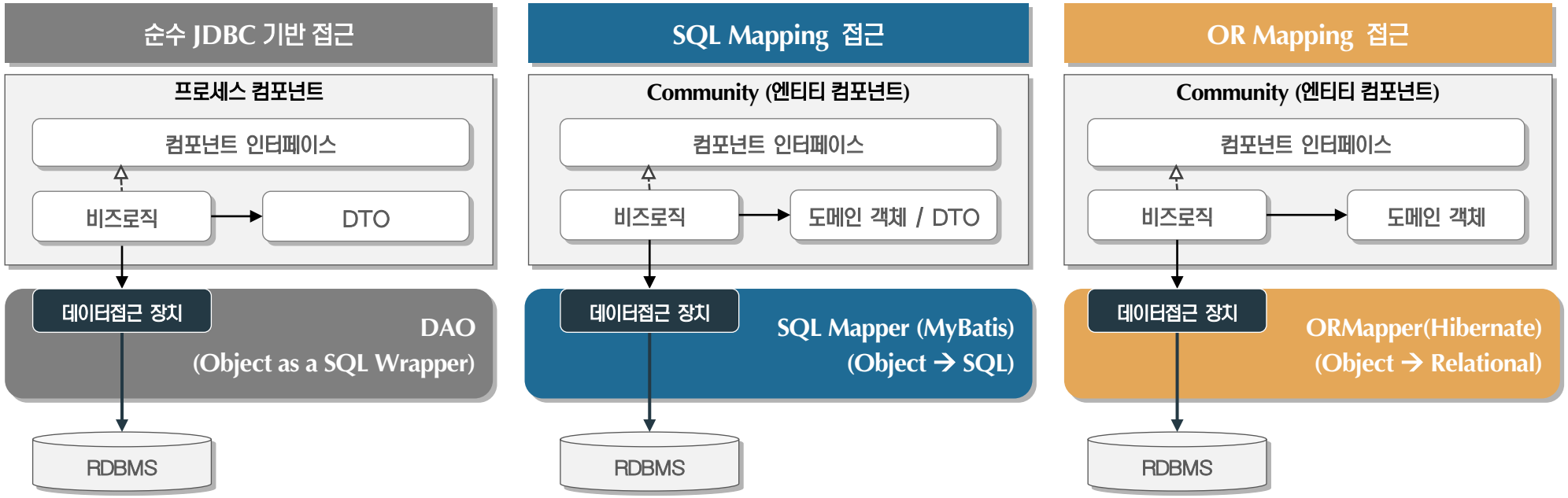
1.3 객체의 세상과 테이블의 세상(3/3)

✓ **Persistence Framework는 로직에서 DB연결 설정을 분리하여, 개발자가 비즈로직에 집중할 수 있도록 도와줍니다.**

- 순수 JDBC를 적용하면, DB자원 연결 및 사용에 관련된 코드가 메소드마다 중복됩니다.
- 상황에 맞는 Persistence Framework를 적용하면, 개발 편의성 뿐 아니라 성능 및 유지보수에도 큰 이점이 됩니다.

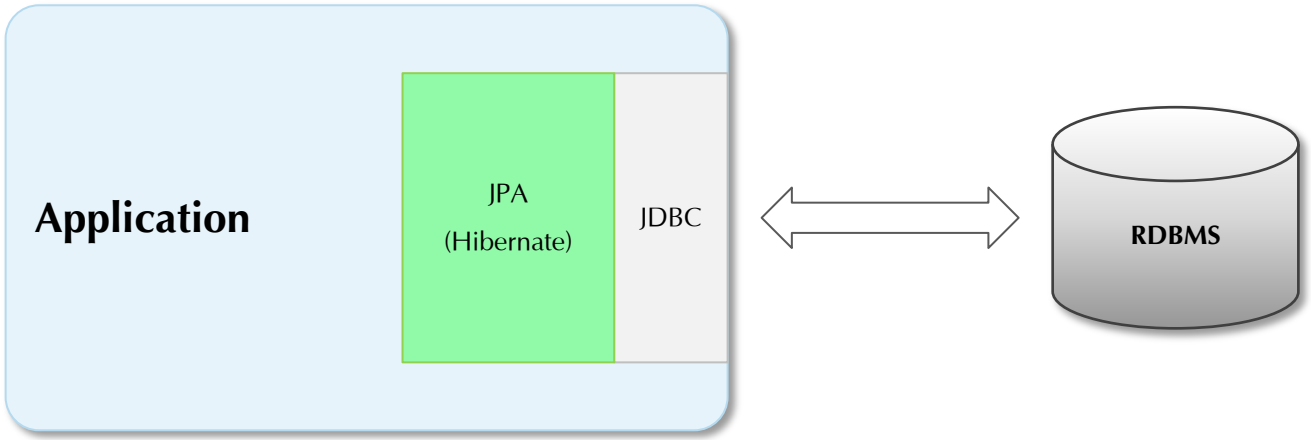
✓ **관계형 데이터 접근 프레임워크는 크게 SQL Mapping과 OR Mapping 접근 기반 프레임워크로 나눠집니다.**

- SQL Mapping 프레임워크는 자바 객체와 쿼리 결과를 매핑합니다.
- OR Mapping 프레임워크는 자바 객체와 데이터베이스 릴레이션(테이블)을 매핑합니다.



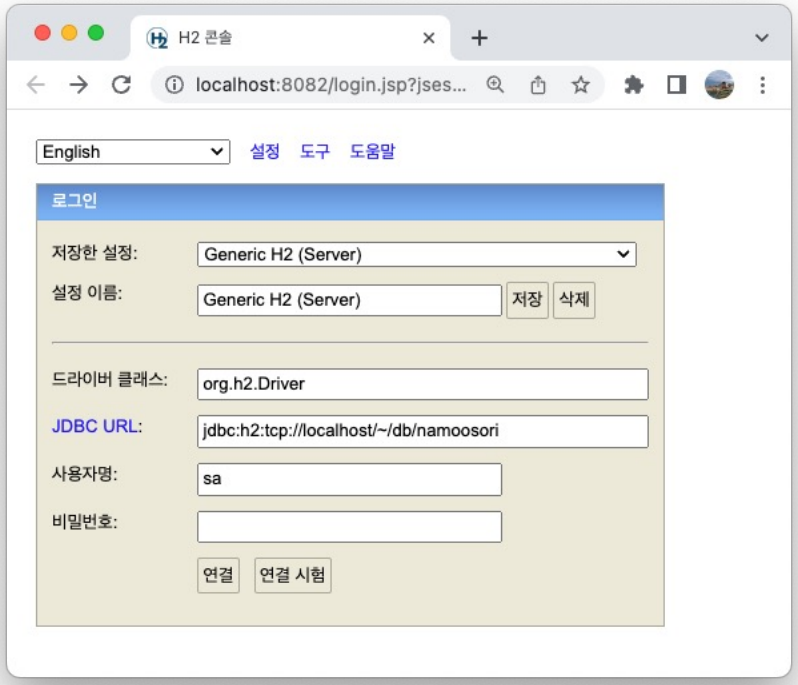
1.4 JPA(Java Persistence API)의 이해

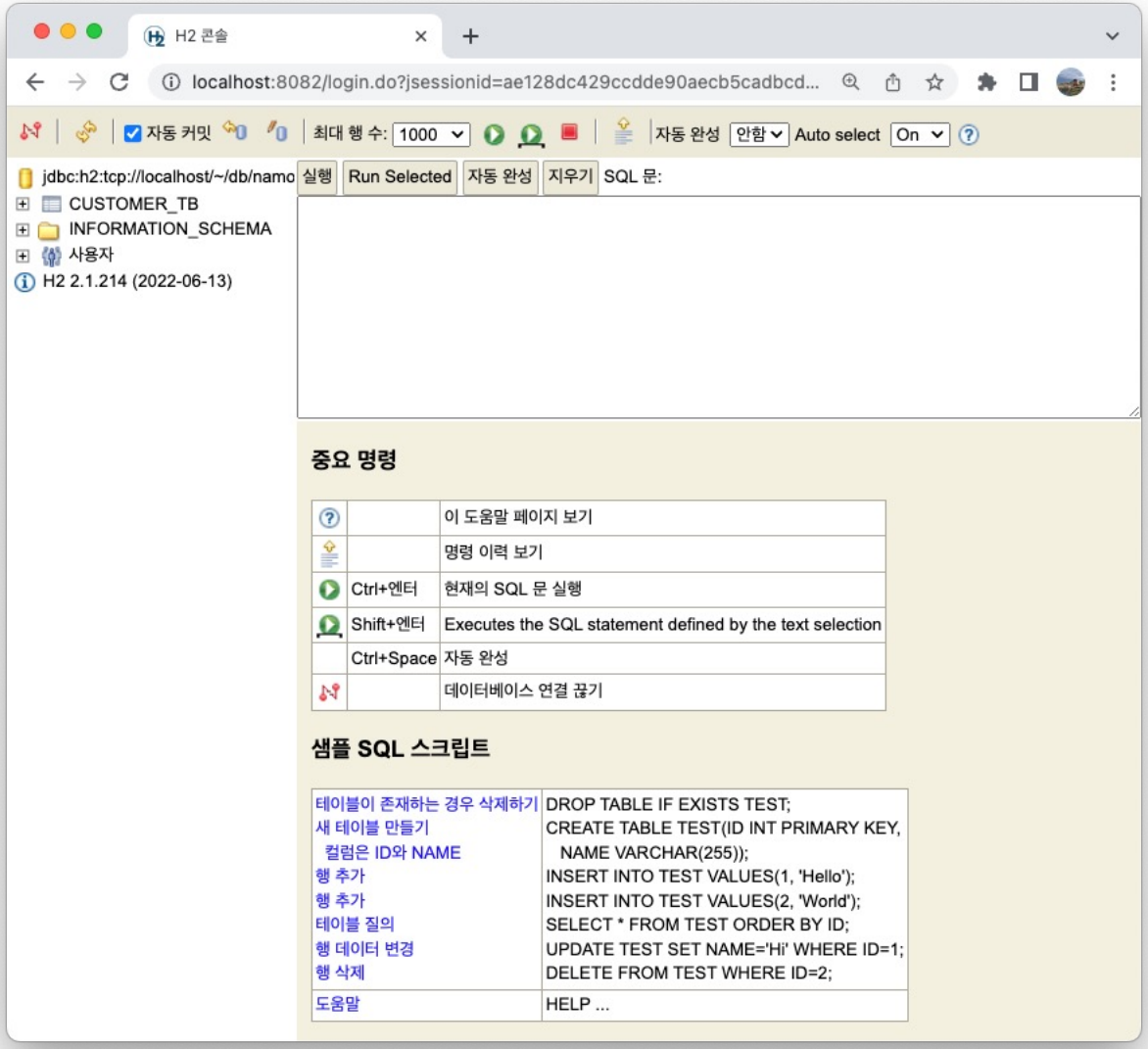
- ✓ JPA(Java Persistence API)은 자바 프로그램에서 관계형 데이터베이스에 접근하는 방식을 명세화한 인터페이스입니다.
- ✓ JPA는 자바 진영의 ORM(Object-Relational Mapping) 기술 표준입니다.
- ✓ JPA는 자바 애플리케이션과 JDBC 사이에서 동작하며, 일반적으로 구현체는 Hibernate 라이브러리를 사용합니다.
- ✓ JPA를 적용할 경우 도메인 객체는 기술에 의존적이지 않으며 재사용을 높일 수 있습니다.
- ✓ JPA를 사용해서 객체를 영속화 하기 위해서는 객체에 annotation을 추가하거나 별도의 메타 데이터 구성이 필요합니다.



1.5 JPA 기초 실습(1/4) - database

- ✓ **Database 설치 : h2 database**
  - http://www.h2database.com/html/main.html
- ✓ **h2 Database의 3가지 모드**
  - Server Mode
  - Embedded Mode
  - In-Memory Mode





1.5 JPA 기초 실습(2/4) – 프로젝트 생성 및 Dependency 구성

✓ Maven Project 생성 및 구성(pom.xml)

pom.xml

```
...
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.1.7.Final</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>2.1.214</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.26</version>
    </dependency>
</dependencies>
...
```

build.gradle

```
dependencies {
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'

    implementation group: 'org.hibernate', name: 'hibernate-core', version: '6.1.7.Final'
    implementation group: 'com.h2database', name: 'h2', version: '2.1.214'
    implementation group: 'org.projectlombok', name: 'lombok', version: '1.18.26'
}
```

1.5 JPA 기초 실습(3/4) – persistence.xml 설정

✓ resources/META-INF/persistence.xml

```
<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="customer-exam">
    <properties>
      <!-- H2 DB          -->
      <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/~/db/namoosori"/>
      <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>

      <!-- Option          -->
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="true"/>
      <property name="hibernate.use_sql_comments" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

1.5 JPA 기초 실습(4/4) – Customer Class 정의

✓ 간단한 데이터로 구성한 Customer Class

Customer
- id : String - name : String - registerDate : long

✓ JPA annotation

- @Entity
- @Table
- @Id

End of Document

✓토론

감사합니다...

- 나무소리
- https://www.youtube.com/namoosori