03 JPA 기초 지식

persistence.xml

persistence.xml

```
<persistence-unit name="playground" transaction-type="RESOURCE_LOCAL">
```

- a box holding all the needed information for creating an EntityManagerFactory instance
- JPA설정을 구별하는 구별자
- 일반적으로 연결할 데이터베이스당 하나의 영속성 유닛을 등록
- 실제 개발에서는 여러 개의 DB를 사용할 수 있음. ex) 테스트 DB와 운영DB

```
<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
<property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/jpa_playground?serverTimezone=UTC" />
<property name="javax.persistence.jdbc.user" value="kim" />
<property name="javax.persistence.jdbc.password" value="sungryulKim12" />
```

- JPA표준 속성: 데이터베이스 연동 및 접속에 필요한 정보 입력
  - ✓ driver → JDBC드라이버
  - ✓ url → 데이터베이스 접속 ID
  - ✓ user → 계정
  - ✓ password → 비밀번호

persistence.xml

```
<property name="hibernate.dialect" value="org.hibernate.dialect.MySQL8Dialect"
/>
<property name="hibernate.show_sql" value="true" />
<property name="hibernate.format_sql" value="true" />
<property name="hibernate.use_sql_comments" value="true" />
```

- Hibernate 전용 속성
  - ✓ dialect → 사용할 방언
  - ✓ show_sql → JPA가 실행하는 sql 보기
  - ✓ format_sql → sql을 formatting하여 보기 좋게
  - ✓ use_sql_comments → 실행되는 sql 설명 보기

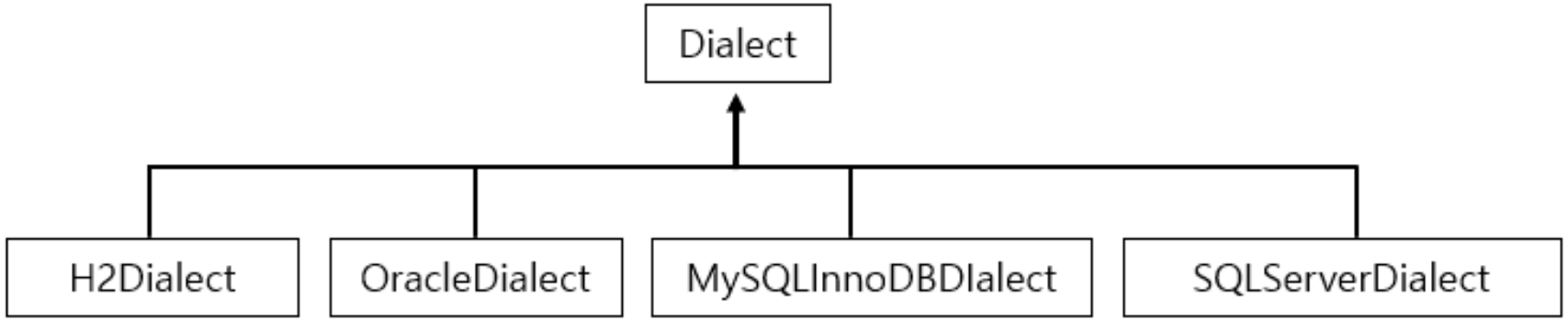
persistence.xml

- 방언

구분	MySQL	오라클
데이터 타입	VARCHAR	VARCHAR2
다른 함수명	SUBSTRING()	SUBSTR()
페이징 처리	LIMIT	ROWNUM

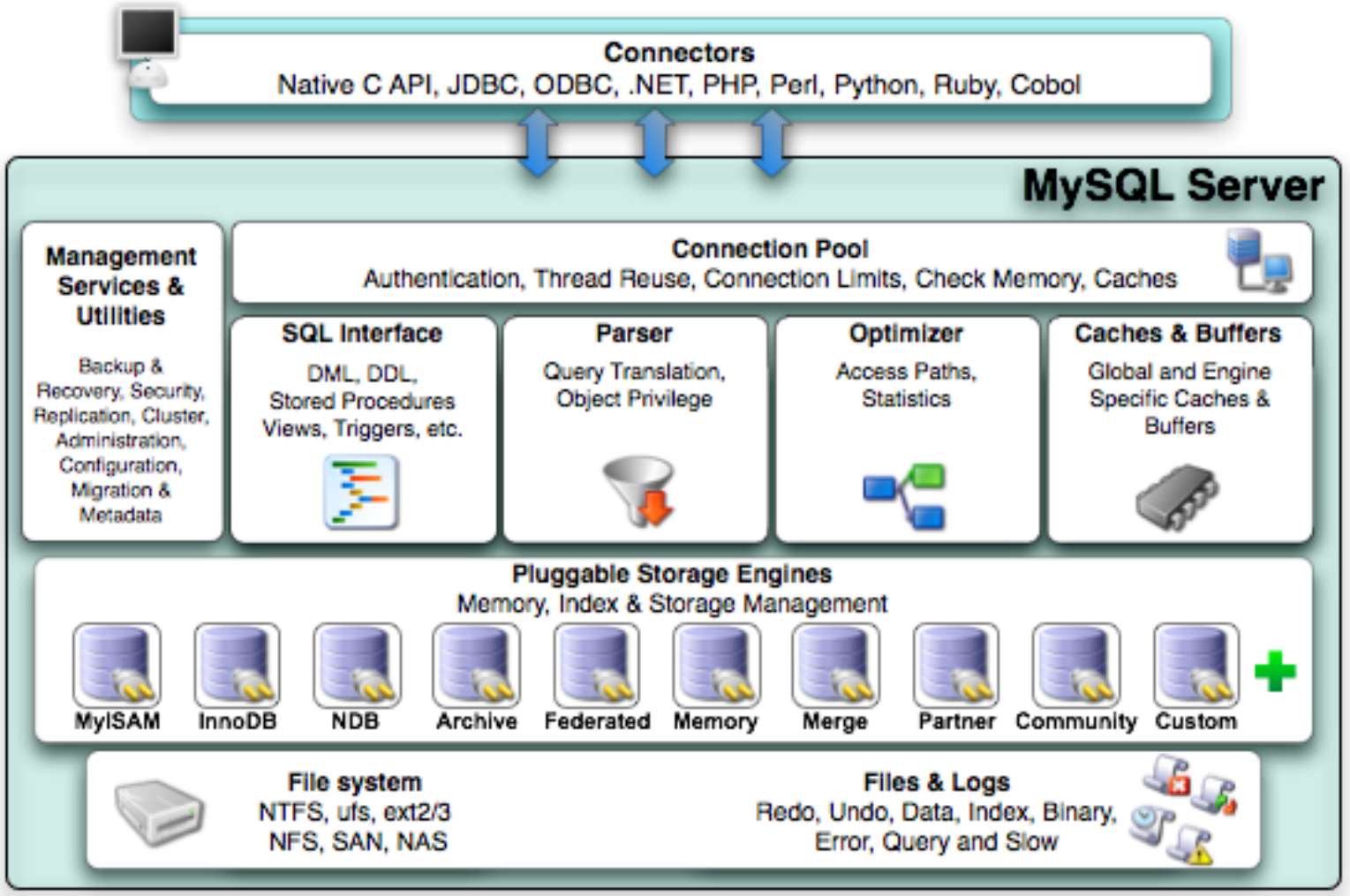
- H2: org.hibernate.dialect.H2Dialect
- 오라클 10g: org.hibernate.dialect.Oracle10gDialect
- MySQL: org.hibernate.dialect.MySQL5InnoDBDialect
  (MySQLDialect)

→ ANSI SQL을 사용하는 이유



persistence.xml

- 서버 엔진과 저장 엔진



[참고] https://mysqldba.tistory.com/2

어노테이션

어노테이션

- **Hibernate JPA Annotations**

Annotation	Package Detail/Import statement
@Entity	import javax.persistence.Entity;
@Table	import javax.persistence.Table;
@Column	import javax.persistence.Column;
@Id	import javax.persistence.Id;
@GeneratedValue	import javax.persistence.GeneratedValue;
@Version	import javax.persistence.Version;
@OrderBy	import javax.persistence.OrderBy;
@Transient	import javax.persistence.Transient;
@Lob	import javax.persistence.Lob;

  - 연관관계와 관련된 어노테이션은 추후에 설명

[참고] https://www.techferry.com/articles/hibernate-jpa-annotations.html

어노테이션

- **@Entity**
  - 정의한 클래스가 영속성 컨텍스트에서 관리될 엔티티임을 나타냄
- **@Table**
  - 엔티티와 매핑되는 테이블을 지정
  - 특히, 엔티티명과 테이블명이 다를 경우 이름을 지정
- **@OrderBy**
  - 정렬 수행

```
@OrderBy("firstName asc")
 private Set contacts;
```

- **@Transient**
  - 테이블의 컬럼에 매핑되지는 않지만 비즈니스 로직 수행 등의 이유로 유지해야 하는 상태(값)가 있을 경우
- **@Temporal**
  - Date, LocalDateTime과 같은 시간 관련 데이터형에 적용

[참고] https://www.techferry.com/articles/hibernate-jpa-annotations.html

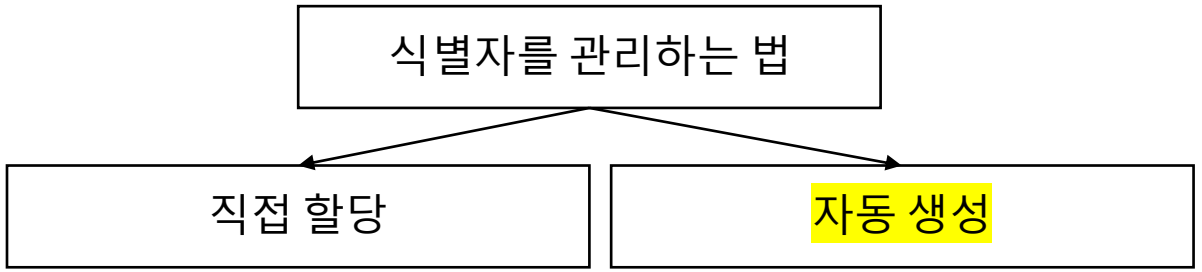
@Column

- **@Column**
  - 컬럼의 세부 속성을 정의
- **@Column의 세부 속성(DDL의 constraint를 걸어주는 것과 유사)**
  - name: 컬럼명
  - length: 문자열에 대하여 길이 지정
  - nullable: null 허용 여부
  - unique: 유일성

[참고] https://www.techferry.com/articles/hibernate-jpa-annotations.html

@Id

- **@Id**
  - 해당 필드를 식별자(PK) 컬럼에 매핑



  - @GeneratedValue
    - 식별자를 직접 할당하지 않고 자동으로 생성
    - 값을 생성하는 방식
      - Identity
      - Sequence
      - Table
      - Custom
      - Auto: 방언에 따라

[참고] https://www.baeldung.com/hibernate-identifiers

@Id

- **IDENTITY Generation**
  - MySQL의 Auto Increment에 해당

```
@Entity
public class Movie {
   @Id
   @GeneratedValue (strategy = GenerationType.IDENTITY)
   private long movieId;
}
```

- **SEQUENCE Generation**
  - 오라클의 Sequence 사용에 대응
  - 데이터베이스의 Sequence 객체 사용

[참고] https://www.baeldung.com/hibernate-identifiers

예시

- **AUTO-DDL**
  - none: No change to the database structure
  - update: Hibernate changes the database according to the given Entity structures
  - validate: If schema is changed, terminating application
  - create: Creates the database every time, but don't drop it when close
  - create-drop: Creates the database then drops it when the SessionFactory closes

[참고] https://www.baeldung.com/hibernate-identifiers

엔티티 사용시 주의사항

- **주의사항**
  - JPA가 엔티티 객체를 생성할 때, 기본 생성자를 사용 → 기본 생성자 필수
    - (Caused by: org.hibernate.InstantiationException: No default constructor for entity)
  - 자바에서는 기본 생성자가 없더라도 자동 생성되지만 사용자가 argument 생성자를 만들면 기본 생성자를 명시해야 함
  - public 또는 protected 생성자 필요
  - final 클래스, enum, interface, inner 클래스에는 사용할 수 없음
  - 저장할 필드에 final을 사용하면 안됨(상속이 가능한 형태로 클래스를 정의해라)

https://javarevisited.blogspot.com/2014/01/why-default-or-no-argument-constructor-java-class.html

- When Hibernate creates an instance of entities using reflection it uses the Class.newInstance() or Constructor.newInstance() method, which requires a no-argument constructor
- Many of open source framework, uses reflection to create instance or Object at runtime, based upon the name of the class.
- Why JPA requires Entity classes to be non-final & fields non-final
  - JPA implementations use proxies in front of your entities to manage for example: Lazy loading. As a final class cannot be extended, a proxy cannot be built.(상속)
- 기본 생성자가 없더라도 엔티티가 동작할 수 있지만 표준 방식은 아님

GeneratedValue 사용시 주의사항

- **주의사항**
  - If you're working with a MySQL database, you should always use *GenerationType.IDENTITY*
  - most efficient approach available
  - Problems with *GenerationType.AUTO* in Hibernate 5
    - In older versions → GenerationType.IDENTITY
    - in Hibernate 5 → GenerationType.TABLE(This approach requires a lot of database queries and pessimistic locks to generate unique values)

[참고] https://thorben-janssen.com/5-things-you-need-to-know-when-using-hibernate-with-mysql/