둘 다 DB와의 연동, 저장을 위한 기술이며, SQL Mapper는 ‘**개발자가 작성한 SQL 실행 결과를 객체에 매핑**’시켜주는 프레임워크이며, ORM은 객체와 DB의 데이터를 ‘**자동으로 매핑**’시켜주는 프레임워크를 말합니다.

**MyBatis의 특징**

[**Java**](https://www.elancer.co.kr/blog/view?seq=47)**코드와 SQL 매핑**

MyBatis를 사용하면, MyBatis 내부에서 그러한 Boilerplate 코드가 구현되어 있고, **MyBatis에서 Java 메소드와 SQL 간에 매핑을 시켜서**[**개발자**](https://www.elancer.co.kr/blog/view?seq=36)**는 Java 메소드 선언과 SQL 문만 만들면 MyBatis가 자동으로 그 둘 간을 연결시켜 주게 됩니다.**

SQL 문장이 Java 코드 내에 들어가 있어서 수정하기도 힘들고, 보기도 힘들었는데, SQL 문을 **별도로 Java 코드에서 분리해두어서 관리가 편하게 하였으며, 분리된 SQL 문을 MyBatis가 찾아서 실행해 주는 기능**을 합니다.

**동적 SQL 생성 기능**

**MyBatis는** Boilerplate 코드 제거 및 SQL 문 분리 외에도 **동적인 SQL(Dynamic SQL) 생성 기능을 제공하여 프로그램 실행 중에 입력되는 파라미터에 따라 서로 다른 SQL 문을 동적으로 생성해 내는 기능을 제공해 줍니다.**

동적으로 SQL 문이 생성된다는 의미는 검색 조건같이 사용자가 입력하는 값에 따라 서로 다른 SQL 문장이 생성되어 실행된다는 의미입니다. 예를 들어, 게시판 테이블을 검색할 때, 사용자가 작성자만 입력하여 검색하면 작성자가 WHERE 절에 있는 다음의 SQL이 생성되고,

* **SELECT \* FROM BOARD WHERE AUTHOR = ‘파라미터’**

제목을 입력하여 검색하면 다음과 같이 제목을 검색하는 SQL이 만들어진다는 의미입니다.

* **SELECT \* FROM BOARD WHERE TITLE LIKE ‘%파라미터%’**

MyBatis 내에 if 문, choose, when, otherwise, foreach 등의 문법을 지원해서 동적인 SQL 문 생성이 가능합니다. 아래 예시에서는 title 파라미터를 입력받게 되면 SQL 문이 아래와 같이 생성되고,

* **SELECT \* FROM BLOG WHERE state = ‘active’ AND title** **like #{title}**

title 파라미터 없이 호출이 되는 경우엔, 아래와 같이 title 조건이 없는 SQL 문이 생성됩니다.

* **SELECT \* FROM BLOG WHERE state = ‘active’**



**MyBatis는**[**개발자**](https://www.elancer.co.kr/blog/view?seq=36)**가 작성한 SQL 문을 Java 객체로 자동으로 매핑 시켜주는  프레임워크입니다.** 그렇다면 JPA는 어떤 기술일까요?

**JPA는 이보다 한발 더 나아가서 SQL 문까지 자동으로 생성해 주고, DB 데이터와 Java 객체를 매핑 시켜주는 기술입니다.** SQL 문장 작성이 필요 없으니 MyBatis보다 한 단계 더 자동화되어 더 편리함과 반복작업을 없애줍니다.

## JPA 특징

MyBatis와 다르게 SQL 문의 작성이 불필요하며 ORM 내부적으로 java 메소드에 적합한 SQL 문이 자동으로 생성이 되어 실행되게 됩니다. Java 개발자는 클래스만 만들어서 사용하고, ORM 프레임워크가 자동으로 관련된 SQL을 만들어 줍니다.

즉, Java 개발자는 jpa.persist(member)라고 사용하면, Insert SQL 문이 자동으로 생성되어 DB에 저장이 됩니다.

또한,  jpa.find(memberId)라고 사용하면, 자동으로 Select SQL 문이 자동으로 생성되어 DB에 조회가 됩니다.

**MyBatis와 JPA 모두 DB를 사용할 때 번거로운 반복작업을 없애줍니다.**MyBatis는 SQL 문을 Java와 분리하여 별도 파일로 관리할 수 있어서, SQL 개발과 유지 보수에 용이하고, JPA는 SQL 문을 아예 만들 필요가 없기 때문에 더욱 자동화되고 반복작업을 더욱 줄여줍니다.

### ****MyBatis 장점****

#### ****SQL 직접 제어****

MyBatis는 SQL에 대한 직접적인 제어를 제공하므로, 개발자가 SQL을 직접 작성하고 최적화할 수 있습니다. 또한 **복잡한 쿼리 또는 특정 데이터베이스에 최적화된 쿼리 작성이 필요한 경우에 적합**합니다.

#### ****학습 용이성****

**SQL을 잘 아는 경우에, JPA에 비해 학습이 용이하고 쉽게 사용이 가능합니다.** SQL을 직접 제어할 수 있다는 것은 DB에 적합한 성능 좋은 SQL문을 만들어서 사용할 수도 있고, 또, 통계/분석용 데이터 추출을 위해 복잡한 쿼리를 만들어야 하는 경우에 자유롭게 테이블 간 JOIN 등을 통해서 원하는 데이터를 추출할 수 있어서 좋습니다.

### ****MyBatis 단점****

#### ****CRUD 단순 작업에 반복 수작업 필요****

SQL을 직접 만들어야 하므로 **CRUD 같은 기본적인 코드에 대해서 단순한 코드를 계속 만들어야 하는 불편한 점이 있습니다.** 단순한 코드를 반복해서 만들면 개발 생산성이 떨어지고, 코드에 대한 [유지보수](https://www.elancer.co.kr/blog/view?seq=203)도 힘들어집니다.

#### ****데이터베이스에 종속적****

특정 DB를 기준으로 SQL 문이 작성되어 있어서, DB를 변경하려면 SQL 문을 전체 다 확인 후 수정해야 합니다.

### ****JPA 장점****

#### ****표준화된 인터페이스****

**JPA는 자바에서의 ORM을 위한 표준 인터페이스를 제공합니다.** Hibernate, EclipseLink, Apache OpenJPA 등의 구현체가 JPA 표준을 따릅니다.

EclipseLink, Apache OpenJPA 등의 구현체가 JPA 표준을 따를 경우 어떤 것이 좋은 건가요? EclipseLink, Apache OpenJPA가 JPA 표준을 따르기 때문에 좋은 것이 아니고 JPA가 자바 표준 기술이기 때문에 다른 프레임웍들이 표준을 따라 만들어진 것입니다.

**Java의 표준을 이용하므로 특정 제품에 종속되지 않고, 개발자 간에**[**협업**](https://www.elancer.co.kr/blog/view?seq=204)**이 용이하고, 일관된 방식으로 개발을 할 수 있습니다.**

#### ****자동 매핑으로 개발 및****[****유지보수****](https://www.elancer.co.kr/blog/view?seq=203)****성 향상****

**JPA는 객체와 데이터베이스 간의 자동 매핑을 지원합니다.** 개발자가 별도의 SQL 작성 없이도 데이터베이스와 객체 간의 매핑이 이루어집니다.

#### ****객체 지향적인 접근 지원****

**JPA는 객체 지향적인 개발에 중점을 둡니다**. 테이블 간의 연관 관계와 객체 간의 연관 관계를 쉽게 다룰 수 있습니다.

#### ****DBMS에 독립적****

사용하는 **DB 종류에 관계없이 JPA에서 자동으로 적합한 SQL Dialect(방언)를 만들어 주기때문에,** **DB가 변경되어도 SQL 문을 다시 작성할 필요가 없습니다.**DB 마이그레이션을 해야 할 경우에 MyBatis로 개발된 시스템과 비교하면 손쉽게 설정 변경만으로 DB 변경이 가능합니다.

### ****JPA 단점****

#### ****높은 학습곡선****

JPA 관련 다양한 스펙과 작성법(@Entitity, @Table, @Column, @Id, @OneToMany, @ManyToOn)을 학습해야 하고, 또 JPA 적용으로 생기는 다양한 이슈, 즉시 로딩(EAGER LOADING), 지연 로딩(LASY LOADING), 영속성 전이(CascadeType), 복합키 매핑(@EmbededId, @IdClass) 등에 대한 해결 방법을 익혀야 합니다. **MyBatis에 비해 배우기가 어렵습니다.**

#### ****복잡한****[****SQL****](https://www.elancer.co.kr/blog/view?seq=156)****생성의 어려움****

일반적으로 시스템을 개발하면 단순한 CRUD와 같은 기능도 많이 있지만, 통계 또는 분석과 같은 화면과 기능도 개발이 필요합니다. 이때는 복잡한 쿼리를 만들어야 하는데 여러 테이블을 Join 해서 데이터 결과를 가져와야 하는 경우에, **JPA로는 복잡한 쿼리를 만드는 데 용이하지 않습니다.**

직접적인 SQL 작성을 통해서 복잡한 쿼리를 만들어야 하는 경우, JPA와 같이 자동으로 만들어지는 SQL로는 원하는 결과를 정확히 얻기가 힘든 경우가 많습니다.

#### ****성능에 대한 고려 필요****

 JPA에 의해 자동으로 SQL이 만들어지다 보니, DB의 특성(index, join 등)을 이해하고 DB에 맞는 SQL을 직접 튜닝해서 만들면 성능이 훨씬 뛰어날 수 있으나, 자동화된 SQL 문으로 인해서 데이터 조회 성능이 떨어질 가능성이 있습니다.

일반적인 간단한 CRUD에는 큰 문제가 없으나, 데이터가 많아지고, 테이블 간 Join이 많아지는 경우, [**SQL**](https://www.elancer.co.kr/blog/view?seq=156)**문을 어떻게 튜닝하는 가로 인한 성능이 크게 차이가 날 여지가 있으므로, JPA 사용 시 이러한 부분을 주의해서 고려할 필요가 있습니다.**

정리

그러면 둘 사이의 선택 기준은 어떻게 하는 것이 좋을까요? **MyBatis와 JPA는 DB와 Java 프로그램간의 데이터 저장, 조회를 위해 사용되지만, 각각의 특징과 사용 시나리오에 따라 선택해야 합니다.**

**복잡한 쿼리와 SQL 제어가 필요한 경우는 MyBatis로 개발할 수 있습니다.**기존 SQL 경험이 풍부하거나, 특정 데이터베이스에 최적화된 SQL 작성이 필요한 경우는 MyBatis를 선택할 수 있습니다.

**간단한 매핑 및 객체 지향적인 접근이 필요한 경우는 JPA가**[**SQL**](https://www.elancer.co.kr/blog/view?seq=156)**작성 등의 반복적인 부분을 해결해 줍니다.** 표준화된 인터페이스와 자동 매핑이 중요하며, 다양한 데이터베이스에 대응해야 하는 경우는 JPA를 선택하면 DB가 변경되더라도 DB에 종속되지 않고 손쉽게 Migration이 가능합니다.

물론, 실제 선택은 [프로젝트](http://www.elancer.co.kr/)의 특성, 개발자의 선호도, 팀의 스킬셋 등을 고려하여 이루어져야 합니다. **MyBatis 또는 JPA 중 하나를 선택하거나 때로는 두 기술을 혼용하여 사용하는 것도 가능합니다.**