**iBATIS의 개념**

iBATIS는 간단한 XML서술자를 사용해서 간단하게 자바빈즈를 SQL statement에 맵핑시킨다. 간단함(Simplicity)이란 다른 프레임워크와 객체관계맵핑툴에 비해 iBATIS의 가장 큰 장점이다. iBATIS Data Mapper를 사용하기 위해서 당신은 자바빈즈와 XML 그리고 SQL에 친숙할 필요가 있다. 여기엔 배워야 할것도 거의 없고 테이블을 조인하거나 복잡한 쿼리문을 수행하기 위해 필요한 복잡한 스키마도 없다. Data Mapper를 사용하면 당신은 실제 SQL문의 모든 기능을 가질수 있다. JDBC 로만 프로그래밍 할 때의 번거로움을 줄여주기 위해 재사용 모듈로 개발된 것인데 그 주요한 어려움을 ibatis 개발자 가이드에서 다음과 같이 정리하고 있다.

 **iBATIS의 주요기능**

**Data Mapper** (com.ibatis.sqlmap.\*)

**개념**  
iBATIS Data Mapper API는 프로그래머에게 자바빈즈 객체를 PreparedStatement파라미터와 ResultSets으로 쉽게 맵핑할 수 있도록 한다. Data Mapper의 기본적인 생각은 간단함(simple)이다. 이는 자바코드의 20%를 사용하여 JDBC기능의 80%를 제공하는 간단한 프레임워크라는 뜻이다.

**작동원리**  
Data Mapper는 자바빈즈, Map구현, 원시래퍼타입(String, Integer…) 그리고 SQL문을 위한 XML문서를 맵핑하기 위한 XML서술자를 사용하는 매우 간단한 프레임워크를 제공한다.

다음은 생명주기에 대한 높은 레벨의 서술이다.

1) 파라미터(자바빈즈, Map 또는 원시래퍼)로써 객체를 제공한다. 파라미터 객체는 update문내에 입력값을 셋팅하기 위해 사용되거나 쿼리문의 where절을 셋팅하기 위해서 사용된다.

2) 맵핑된 statement을 실행한다. 이 단계는 마법이 일어나는곳이다. Data Mapper프레임워크는  
PreparedStatement 인스턴스를 생성할것이고 제공된 파라미터객체를 사용해서 파라미터를 셋팅한다. 그리고 statement를 실행하고 ResultSet으로부터 결과 객체를 생성한다.

3) update의 경우에 영향을 미친 rows의 숫자를 반환한다. 조회문일경우에 한 개(single)의 객체 또는 컬렉션 객체를 반환한다. 파라미터처럼 결과 객체는 자바빈즈, Map 원시타입래퍼또는 XML이 될수 있다.

출처 : <http://blog.empas.com/ahnyounghoe/11535652>

========================================================================================================

가장 간단히 설명하면, JAVA에서 DB관련 작업을 편하게 해주는 **프레임웍**정도라고 할까?  
  
iBATIS in action에서 iBATIS는 "SQL 실행 결과를 자바빈즈 혹은 Map 객체에 매핑해주는 퍼시스턴스 솔루션으로 SQL을 소스 코드가 아닌 XML로 따로 분리해 관리하여 지겨운 SQL 문자열 더하기에서 해방시켜 줍니다. 또한 XML에서 동적 SQL 요소를 사용하여 쿼리 문장을 프로그래밍 코딩 없이 자유롭게 변환할 수 있게 해줍니다. 이러한 접근방식으로 인해 iBATIS를 사용하면 JDBC를 사용할 때보다 약 60% 정도의 코드만으로 프로그램 작성이 가능하다" 라고 한다.  
  
말로만 하면 뭔소리인지 모르겠으니 간단한 예제 정도를 들어보자.

- 일반적인 JDBC 예제

**import** javax.naming.\*;  
**import** javax.sql.\*;  
**import** java.sql.\*;  
  
**public** **class** Employee {  
  **public** Account getAccount(**int** id) **throws** SQLException, NamingException{  
    Account account = **null**;  
      
    String sql = "select \* from employee where id= ?";  
      
    Connection conn      = **null**;  
    PreparedStatement ps = **null**;  
    ResultSet rs       = **null**;  
      
    **try** {        
      Context ctx = **new** InitialContext();  
      DataSource ds =  
              (DataSource)ctx.lookup(  
                 "java:comp/env/jdbc/TestDB");   
      conn = ds.getConnection();  
      ps = conn.prepareStatement(sql);  
      ps.setInt(1, id);  
      rs = ps.executeQuery();  
        
      **while**( rs.next()){  
        account = **new** Account();  
        account.setId(rs.getInt("ID"));          
      }  
    } **finally** {  
      **try** {  
        **if** ( rs != **null** ) rs.close();  
      } **finally** {  
        **try** {  
          **if** (ps != **null**) ps.close();  
        } **finally** {  
          **if** (conn != **null**) ps.close();  
        }  
      }   
    }   
    **return** account;    
  }  
}

뭐다들 아시겠지만 간단히 쿼리를 날려서 Acount 객체에 담아가지고 오는 소스이다. 대충봐도 무척이나 길다,

이걸 iBATIS를 이용해서 처리하는 예를 보자,

- iBATIS 를 이용한 예

**acount.xml**  
<select id="getAcount" resultClass="Acount" parameterClass="java.lang.Integer">  
    select \* from employee where id= #id#  
</select>  
  
**java**  
Acount act = (Acount) sqlMap.queryForObject("getAcount",new Integer(5));

보면 알겠지만 상단에 쿼리를 닮고있는 xml과 아래 간단히 크 쿼리를 실행시키는 java 한줄정도?이다.

사실 iBATIS를 설정하는 config파일과 sqlMap객체를 불러오는 부분이 있긴하지만, 무척이나 좋아보이도록,  
이것만 쓰겠다. -\_-;;  
  
iBATIS 의 목표와 특징은 몇마디로 짧게정의하다면,

쉽고, 간단하고, 의존성이 적은 프레임웍이라는 것이다.   
sql문과 java코드와의 분리만으로도 java개발자는 쿼리문을 신경쓰지 않아도 된다. sql문이 변경되더라도,

파라미터 값만 변경되지 않는다면, java소스에서는 수정할 부분이 없다.

출처: <https://jonghoit.tistory.com/73> [당구치는 개발자]