Introibatis

iBATIS SQL Maps 2.0에 대한 소개

번역 허태명

iBatis SQL Maps는 여러분의 자바 객체와 관계형 데이터베이스 사이에서 이동하는데 매우 간단하고 유연한 수단을 제공합니다. 단 한 줄의 JDBC 코드 없이 진정한 SQL의 완전한 능력을 사용하세요!

SQL Maps 프레임워크는 일반적으로 관계형 데이터베이스에 접근하기 위해 필요로 하는 자바 코드의 양을 현저하게 줄여주는데 도움을 줍니다. 이 프레임워크는 매우 간단한 형식의 XML을 사용하여 자바 Bean을 SQL 문장에 매핑합니다. 단순성은 다른 프레임워크와 ORM 툴들에 비해 SQL Maps의 가장 큰 장점입니다. SQL Maps를 사용하기 위해 여러분은 단지 자바 Bean과 XML, SQL에만 친숙하면 됩니다. 새로이 배워야 할 것은 거의 없습니다. 테이블을 조인하거나 복잡한 쿼리를 실행하기 위해 필요로 하는 복잡한 설계는 없습니다. SQL Maps를 사용함으로써 여러분은 진정한 SQL의 완전한 능력을 당장 사용할 수 있습니다. SQL Maps 프레임워크는 대부분의 데이터베이스를 어떤 객체 모델로도 매핑할 수 있고 레거시 설계, 심지어 그것이 나쁜 설계라도 매우 유연하게 대처할 수 있습니다. 이것은 특정한 데이터베이스 테이블의 생성이나 객체, 코드 생성없이 모두 이루어 집니다.

궁금한가요? 여기에 왜 여러분이 iBATIS SQL Maps를 좋아할 수 있는지 10가지 이유가 있습니다.

왜 iBATIS SQL Maps를 사용하나요?

10. JDBC 드라이버를 가지고 있는 어떤 데이터베이스에서도 사용할 수 있습니다.(플러그-인이 필요없습니다.)
9. 설정가능한 캐쉬 (의존성 포함)
8. Local 과 Global 트랜잭션 지원과 관리(JTA)
7. 간단한 XML 매핑 문서 구조
6. Map, Collection, List와 기본형의 래퍼(Integer, String 등)을 지원합니다.
5. 자바 Bean 스타일의 클래스를 지원합니다.(get/set 메소드)
4. 복잡한 객체 매핑을 지원합니다. (list와 복잡한 객체 모델 등의 생성)
3. 객체 모델은 결코 완벽하지 않습니다. (변경이 필요없습니다!)
2. 데이터베이스 디자인은 결코 완벽하지 않습니다 (변경이 필요없습니다!)
1. 여러분은 SQL을 이미 알고 있습니다. 왜 다른 것을 배우는데 시간을 낭비합니까?

무엇보다 가장 좋은 것은, 100% Open Source 프리웨어입니다!

예제는 어떤가요? 실제 코드에서 MappedStatement가 어떻게 사용되는지 확인해보세요.

```
<!-- 다음은 간단한 SQL Map의 예제입니다. 이 쿼리는 DB 컬럼의
알리어스를 자동적으로 자바 Bean 프로퍼티에 매핑해주는 SQL Map
프레임워크의 기능을 사용합니다. Address 클래스가 다음과 같은
프로퍼티를 가지고 있다고 생각해 보세요: id (int), description (String),
street (String), city (String), province (String), and postalCode (String).
SQL 문장(표준 SQL 기능)에서 컬럼 알리어스를 사용하고 result-class
어트리뷰트에서 Address 클래스를 명명함으로써, 컬럼의 값들은 자동적으로
자바 Bean에 매핑됩니다!
이것은 SQL Map 프레임워크를 사용하여 SQL 쿼리에서 객체(자바 Bean)을
얻는 단지 여러가지 방법 중의 하나입니다. -->

<select id="getAddress" parameterClass="int" resultClass="examples.domain.Address">
SELECT ADR_ID as id, ADR_DESCRIPTION as description, ADR_STREET as street, ADR_CITY as city, \
ADR_PROVINCE as province, ADR_POSTAL_CODE as postalCode
FROM ADDRESS
WHERE ADR_ID = #value#
</select>
```

어떻게 자바에서 이것을 실행하나요?

```
/* 다음 간단한 몇 줄의 코드는 위의 문장의 #value# 파라미터에 전달할 키로써
 * integer 5를 사용하여 위의 매핑된 문장을 실행합니다.
 */

Integer pk = new Integer(5);
Address address = (Address)sqlMap.queryForObject("getAddress", pk);
```

아래에 더 많은 예제가 있습니다.

```
<!-- SQL Maps은 단지 select 쿼리에서만 사용되지 않습니다, 여기에 어떻게
새로운 Address 클래스가 insert되는지 보여주고 있습니다. -->

<insert id="insertAddress" parameterClass="examples.domain.Address">
insert into ADDRESS(ADR_ID, ADR_DESCRIPTION, ADR_STREET, ADR_CITY, ADR_PROVINCE, ADR_POSTAL_CO
values (#id#, #description#, #street#, #city#, #province#, #postalCode#)
</insert>
```

// 이 insert 문장을 실행하는 것은 select 쿼리와 마찬가지로 쉽습니다. // executeUpdate 메소드는 insert, update, delete를 위해 사용됩니다.

```
Address address = new Address();
address.setId(15);
address.setDescription("Bob's Comic Book Store");
address.setStreet ("16 Somestreet");
...
sqlMap.insert ("insertAddress", address);
```

```
<!-- 여기에 하나 이상의 결과를 리턴하는 예제가 있습니다. -->

 <select id="getProductByCategory" parameterClass="string"
                                   resultClass="examples.domain.Product">
  select
    PRD_ID           as id,
    PRD_CATEGORY     as category,
    PRD_DESCRIPTION  as description,
    PRD_RETAIL       as retail,
    PRD_QUANTITY     as quantity
  from PRODUCT
  where PRD_CATEGORY = #value#
</select>
```

```
// "개" 카테고리의 제품 리스트를 위해 위의 문장을 실행하는
// 것은 다음과 같이 간단합니다.

String category = "dog";
List productList = sqlMap.queryForList("getProductByCategory", category);

// 결과가 너무 많은가요? 자바 Bean에 호환되고 JSP에서 쉽게
// 사용할 수 있는, 완전히 탐색가능하고 lazy-loading 기능과
// 페이징이 가능한 제품의 list 는 어떤가요?
// 페이지당 10개면 괜찮을까요?

PaginatedList productList = sqlMap.queryForList("getProductByCategory",
                                                category,
                                                10);

productList.nextPage();
productList.previousPage();
productList.isNextPageAvailable();
// etc.

// 동일한 쿼리에서 단지 Product 클래스의 ID와 Description의 Map만 필요하나요?

Map descriptionMap = sqlMap.queryForMap ("getProductByCategory",
                                         category,
                                         "id",
                                         "description");
```

```
<!-- 레거시 시스템의 복잡한 전용 쿼리? 예를 들어, 아마 여러분은 오라클을
사용하여 복잡한 parent/child의 트리를 구할 때? -->

<select id="getEmployeesByManagerRecursively" parameterClass="int"
                                              resultClass="examples.domain.Employee">
    SELECT
```

```
        emp_id           as id,
        emp_number       as employeeNumber,
        emp_first_name   as firstName,
        emp_last_name    as lastName,
        emp_dept_code    as departmentCode
    FROM employee
    START WITH emp_id = #value#
    CONNECT BY PRIOR emp_manager_id = emp_id;
</select>
```

```
Integer managerId = new Integer(15);
List employeeList = sqlMap.queryForList("getEmployeesByManagerRecursively",
                                        managerId);
```

더 많은 예제를 보고 싶으세요?

JPetStore 4를 살펴보세요. JPetStore4는 최신의 기능을 많이 사용하는 완전한 어플리케이션의 훌륭한 예입니다. 그리고 또한 포괄적인 완전한 문서의 Developer Guide 가 있습니다.