QueryForMapExample

여기서 사용된 소스는 iBATIS SQLMaps 2.3.0 배포판에 포함된 샘플코드를 사용하였다.

SqlMapConfig.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMapConfig
    PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>

  <!-- Configure a built-in transaction manager.  If you're using an
       app server, you probably want to use its transaction manager
       and a managed datasource -->
  <transactionManager type="JDBC" commitRequired="false">
    <dataSource type="SIMPLE">
      <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>
      <property name="JDBC.ConnectionURL" value="jdbc:mysql://localhost/ibatis_sample"/>
      <property name="JDBC.Username" value="root"/>
      <property name="JDBC.Password" value="root"/>
    </dataSource>
  </transactionManager>
  <!-- List the SQL Map XML files. They can be loaded from the
       classpath, as they are here (com.domain.data...) -->
  <sqlMap resource="com/mydomain/data/Account.xml"/>
</sqlMapConfig>
```

SimpleExample

```
package com.mydomain.data;

import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;
import com.ibatis.common.resources.Resources;
import com.mydomain.domain.Account;

import java.io.Reader;
import java.io.IOException;
import java.util.List;
import java.util.Map;
import java.sql.SQLException;

public class SimpleExample {

  private static SqlMapClient sqlMapper;

  static {
    try {
      Reader reader = Resources.getResourceAsReader("com/mydomain/data/SqlMapConfig.xml");
      sqlMapper = SqlMapClientBuilder.buildSqlMapClient(reader);
      reader.close();
    } catch (IOException e) {
      // Fail fast.
      throw new RuntimeException("Something bad happened while building the SqlMapClient instance." + e, e);
    }
  }

.............

  public static Map selectAccountMap() throws SQLException {
    return sqlMapper.queryForMap("selectAllAccounts", null, "id");
  }

  public static Map selectAccountMapValue() throws SQLException {
    return sqlMapper.queryForMap("selectAllAccounts", null, "id", "emailAddress");
  }
}
```

Account.java

```
package com.mydomain.domain;
```

```
public class Account {

  private int id;
  private String firstName;
  private String lastName;
  private String emailAddress;

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  public String getEmailAddress() {
    return emailAddress;
  }

  public void setEmailAddress(String emailAddress) {
    this.emailAddress = emailAddress;
  }

}
```

Account.xml

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMap
    PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Account">

  <!-- Use type aliases to avoid typing the full classname every time. -->
  <typeAlias alias="Account" type="com.mydomain.domain.Account"/>

  <!-- Result maps describe the mapping between the columns returned
       from a query, and the class properties.  A result map isn't
       necessary if the columns (or aliases) match to the properties
       exactly. -->
  <resultMap id="AccountResult" class="Account">
    <result property="id" column="ACC_ID"/>
    <result property="firstName" column="ACC_FIRST_NAME"/>
    <result property="lastName" column="ACC_LAST_NAME"/>
    <result property="emailAddress" column="ACC_EMAIL"/>
  </resultMap>

  <!-- Select with no parameters using the result map for Account class. -->
  <select id="selectAllAccounts" resultMap="AccountResult">
    select * from ACCOUNT
  </select>
</sqlMap>
```

설명

queryForMap() 메소드를 메소드 인자의 형태에 따라 두가지가 제공된다.
- queryForMap(String id, Object parameterObject, String keyProp)
- queryForMap(String id, Object parameterObject, String keyProp, String valueProp)

queryForMap(String id, Object parameterObject, String keyProp)

예제 소스의 SimpleExample에 보면 다음처럼 사용한 경우가 있다.

```
public static Map selectAccountMap() throws SQLException {
  return sqlMapper.queryForMap("selectAllAccounts", null, "id");
}
```

여기서 인자를 순서대로 보면 첫번째 인자는 Account.xml에서 "selectAllAccounts" 라는 id를 가리키는 것으로 다음 쿼리문을 실행하도록 지정하는 셈이다.

```
select * from ACCOUNT
```

두번째 인자는 parameter객체를 나타내는 것으로 현재 예제에서는 일단 전체 레코를 가져오는 것이라 조회조건을 나타내기 위한 파라미터 객체가 없다. 그래서 null로 지정했다. 조건문을 만들고자 할때는 이 객체를 적절히 셋팅해서 전달하면 된다.

세번째 인자는 queryForMap() 메소드 호출로 인해 반환되는 Map에서 key역할을 담당하는 것으로 여기서 반환되는 객체는 Account라는 객체를 가지는 Map객체이다. 현재 데이터가 다음과 같다면 두개의 Account객체가 넘어오는데 여기서 id가 1인 Account와 2인 Account를 구별하기 위한 Map의 키 역할을 담당할 칼럼을 지정하는 의미라고 보면 이해가 빠를듯 하다. 하지만 여기서 주의할 점은 세번째 인자의 값을 칼럼명이 아닌 해당 칼럼에 대응되는 Account객체의 변수명을 지정해줘야 한다는 것이다.

```
mysql> select * from account;
+--------+----------------+---------------+------------------+
| ACC_ID | ACC_FIRST_NAME | ACC_LAST_NAME | ACC_EMAIL        |
+--------+----------------+---------------+------------------+
| 1      | DongGuk        | Lee           | fromm0@gmail.com |
| 2      | DongGuk        | Lee           | fromm0@gmail.com |
+--------+----------------+---------------+------------------+
2 rows in set (0.00 sec)
```

이론적으로는 이렇게 실제 테스트 코드를 돌려보자.

```
Map map = SimpleExample.selectAccountMap();
System.out.println("# Map 정보 \n"+map);

account = (Account) map.get(1);
System.out.println("# id : " + account.getId());
System.out.println("# emailAddress : " + account.getEmailAddress());
System.out.println("# firstName : " + account.getFirstName());
System.out.println("# lastName : " + account.getLastName());
```

결과는 다음과 같다

```
# Map 정보
{2=com.mydomain.domain.Account@3eca90, 1=com.mydomain.domain.Account@64dc11}
# id : 1
# emailAddress : fromm0@gmail.com
# firstName : DongGuk
# lastName : Lee
```

queryForMap(String id, Object parameterObject, String keyProp, String valueProp)

예제 소스의 SimpleExample에 보면 다음처럼 사용한 경우가 있다.

```
public static Map selectAccountMapValue() throws SQLException {
  return sqlMapper.queryForMap("selectAllAccounts", null, "id", "emailAddress");
}
```

인자는 네번째를 제외하고는 앞의 예제와 동일하다. 네번째 인자는 Account객체에서 어떤값만 반환받을지는 결정하는 값이라고 보면 된다. 여기서는 selectAccountMapValue() 메소드에서 볼수 있듯이 emailAddress만을 반환받고자 했다.

이론적으로는 이렇게 실제 테스트 코드를 돌려보자.

```
map = SimpleExample.selectAccountMapValue();
System.out.println("# Map 정보 \n"+map);
String email = (String) map.get(1);
System.out.println("# 반환값 : " + email);
```

결과는 다음과 같다.

```
# Map 정보
{2=fromm0@gmail.com, 1=fromm0@gmail.com}
# 반환값 : fromm0@gmail.com
```

여기서는 간단한 예제를 통해 queryForMap()메소드를 알아보았다. 이 예제를 기반으로 본인이 직접 사용해본다면 충분히 이해를 할수있으리라 생각한다. 그래도 이해가 되지 않는다면 질문하길 바란다.