SQLMap Editor

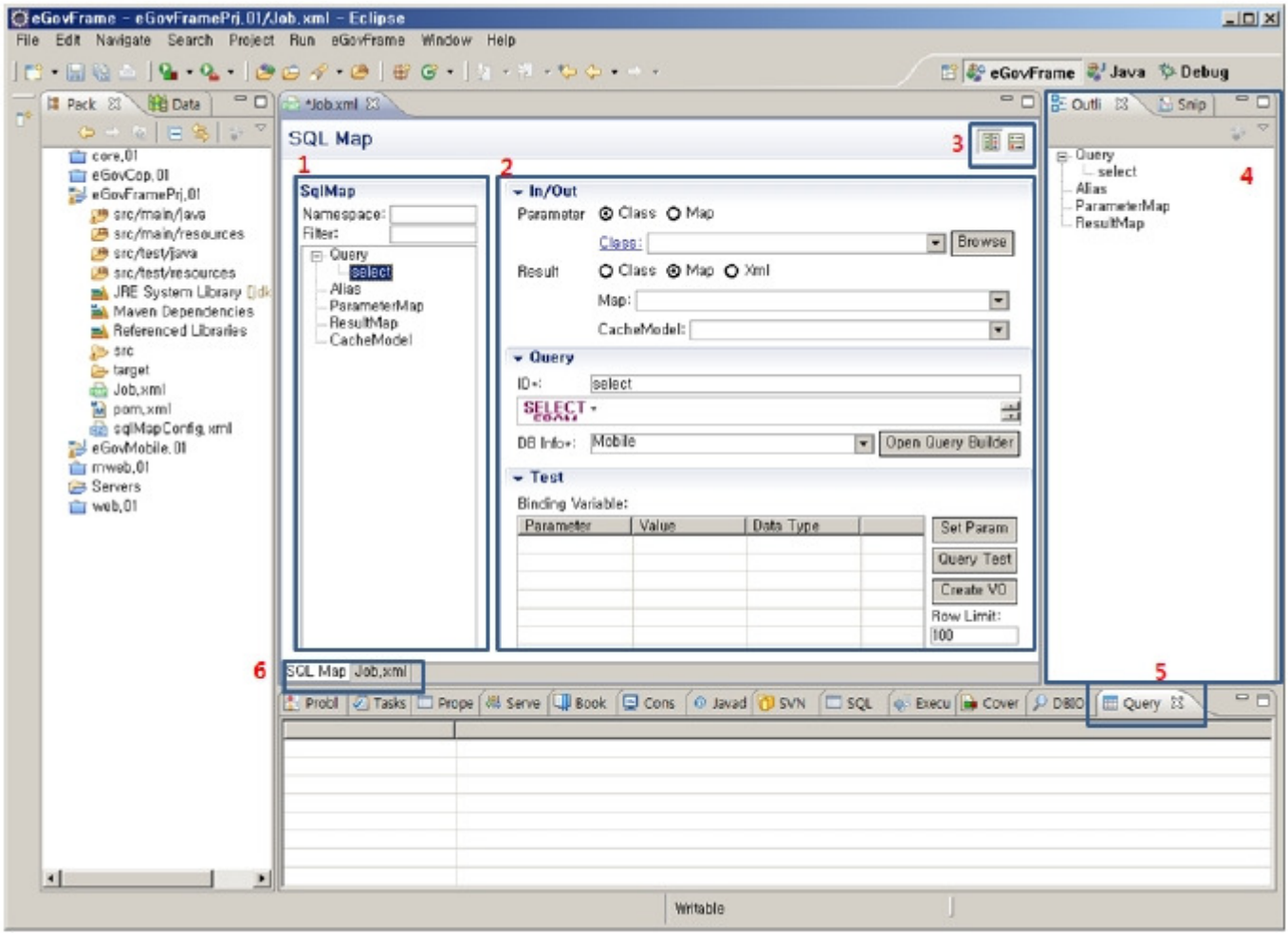
Summary

The SQLMap Editor is a tool for conveniently editing SQL map files. Knowledge of XML syntax is not required in order to use the editor's form UI to create SQL map files. The editor also helps minimizing typographic errors when editing XML files. Main features of the editor are as below.

- Add/modify/remove CRUD Queries
- - Query Builder
- - Query Test
- - ParameterMap configuration
- - ResultMap configuration
- - Alias setting
- - Duplicate group ID detection
- - Shortcut using group IDs via Outline
- - Custom Layout
-

Description

Display



<Figure 1> SQL Map Editor

① SqlMap Namespace, group component tree.
② Different editing view for elements such as SqlMap Query, ParameterMap, ResultMap, Alias and so on.
③ Custom layout settings. (Horizontal or Vertical)
④ SqlMap composition Outline.
⑤ Query Result View screen, displays Query Test results.
⑥ User can choose between using Form UI or manually editing XML source when editing SqlMap files.

Main features

SQLMap Tree

SQLMap Editor displays Query, ParameterMap, ResultMap and Alias groups in tree view so that the SQLMap composition is easier to understand. (See <Figure 1> #1) Also, SQLMap Tree enables easy access to each element so that add, modify and remove element can be easily performed.

Building CRUD Queries

SQLMap files will contain different XML elements depending on query mapping statements (see <Table 1>). SQL Map Editor provides appropriate editing views for each query types (see <Figure 1> #2).

Statement	Properties	Descripti
select	id, parameterC lass,resultC lass, parameterMap, resultMap	Search data
insert	id, parameterC lass,parameterMap	Insert data
update	id, parameterC lass,parameterMap	Modify data
delete	id, parameterC lass,parameterMap	Delete data

<Table 1> Query mapping statement types and corresponding XML elements

Query editing view can be categorized as In/Out, Query, Test tabs.

In/Out

For creating Parameter and Result for queries. Parameter and Result can be either a Class or a ParameterMap. Result only applies to SELECT queries. If user

has created Aliases in advance, they will be displayed as Class options.

Query

For modifying query IDs, editing queries, and selecting database information needed for Query Tests.

Test
Binding variables for queries can be assigned or entered, and said queries can also be tested. If row limit is specified, it will limit the number of lines for query result view.

Query Builder
Since SQLMap Editor provides query building functionality, users do not need to use a separate database client to check table and column names. SQL Map Editor's Query Builder lets users browse database table lists and column names, and can even run simple queries via auto-run feature to check the query results.

Query Test

It is likely that users will use parameters on top of simple queries to formulate complex queries. For said queries, testing feature will come in as useful when Formulationg them.

Creating ParameterMaps
Users can easily create ParameterMaps via SQL Map Editor's Form UI. Entires other than ID and class can be added from the property list separately. User-defined ParameterMap elements can be used when setting parameters for CRUD queries.

Creating ResultMaps
ResultMap is the <resultMap> element of SQL Map files. <resultMap> elements can be easily created by using SQL Map Editor's Form UI. Like ParameterMaps entires other than ID and class can be added from the property list separately. User-defined resultMap elements can be used when setting results for SELECT queries.

Creating aliases
Alias is the <typeAlias> element of SQL Map files. <typeAlias> sets aliases for classes. Aliases can be used for quickly setting frequently used classes when setting parameters or results.

Duplicate group ID detection
Duplicate group IDs that are created when users create or edit multiple SQLMap elements will likely cause SQLMaps to malfunction. SQLMap Editor automatically checks for duplicate IDs when users add, modify or remove elements. If duplicate IDs are found, it displays warnings.

Custom Layout
SQLMap Editor provides horizontal and vertical layouts. (See <Figure 1> #3)

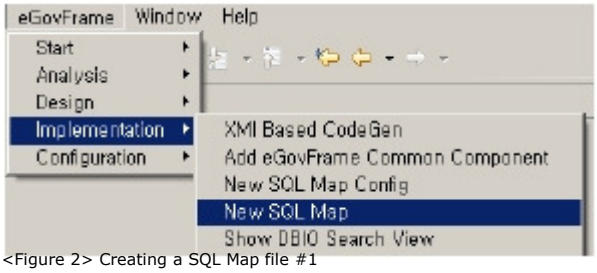
SQLMap Outline
SQLMap Editor provides outline view when users manually edit XML sources. (See <Figure 1> #4)

Manual

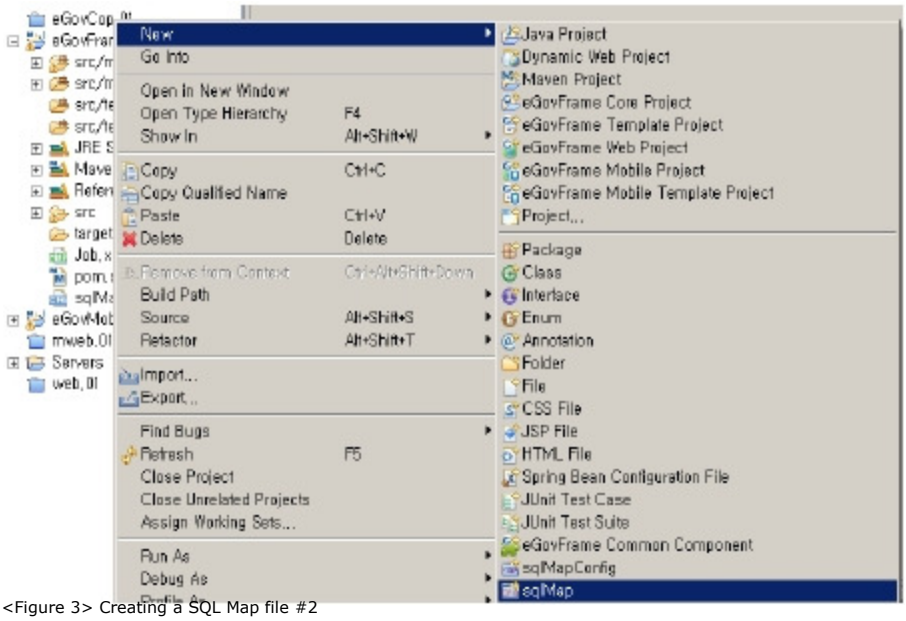
Create a SqlMap File

1. Choose e-GovFrame → Implementation → New SQL Map from the menu, or New → sqlMap from the context menu to create a new file. (See <Figure 2> and <Figure 3>)

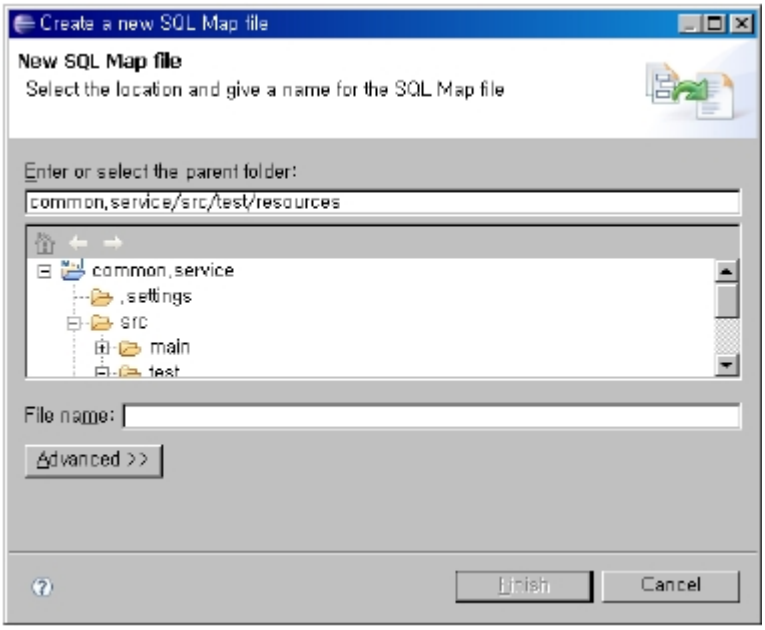
2. SqlMapConfig 파일이 위치할 폴더를 선택하고 파일명을 입력한다.(<그림 4> 참조)



<Figure 2> Creating a SQL Map file #1



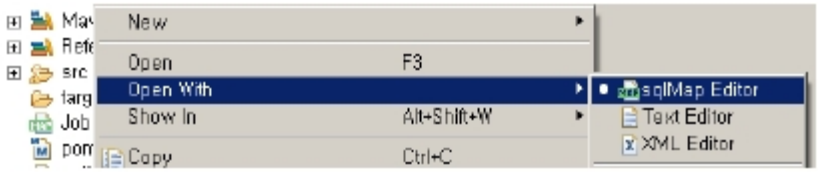
<Figure 3> Creating a SQL Map file #2



<Figure 4> Creating a SQL Map file #3

Opening SqlMap Editor

Double-clicking or selecting Open on a SQL Map File from the Package Explorer will open the file. However, if there are issues preventing the file from being so choose Open With from the context menu to open with the SQL Map Editor. (See Figure 5>)



<Figure 5> Opening with the SQL Map Editor via a context menu

Create a Query Map

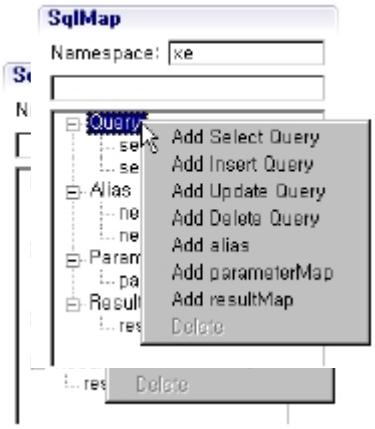
1

1. Right-clicking on the SQL[2)] Map Tree will bring up the context menu. (See <Figure 6>)
2. Choose Add XXX Query from the context menu. (Choose from Select/Insert/Update/Delete.)
3. Editing view for the new query will appear on the right of the SQLMap Tree.
4. Choose Map or Class for the Parameter from the In/Out tab.
   If radio buttons are used for choosing between Map and Class, appropriate further entries will be enabled or disabled for each of them.
   Also, previously registered Aliases will also appear here. (See <Figure 7>)

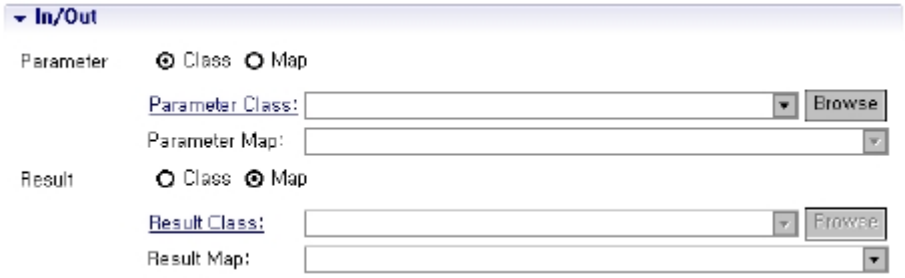
1. For Select queries, Result item will also appear on the In/Out tab for you to fill out.

2. Query tab can be used to modify query ID or queries themselves. (See <Figure 8>) An easier way to build queries is to use Query Builder. Query Builder requires a database connection to be set up in advance, so you can use the Data source explorer to configure a database.

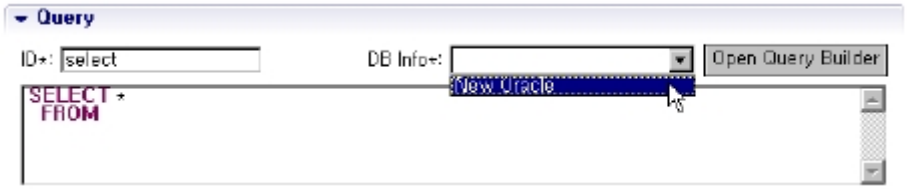
.



<Figure 6> Context menu of a SQL Map Tree



<Figure 7> In/Out settings of QueryMap

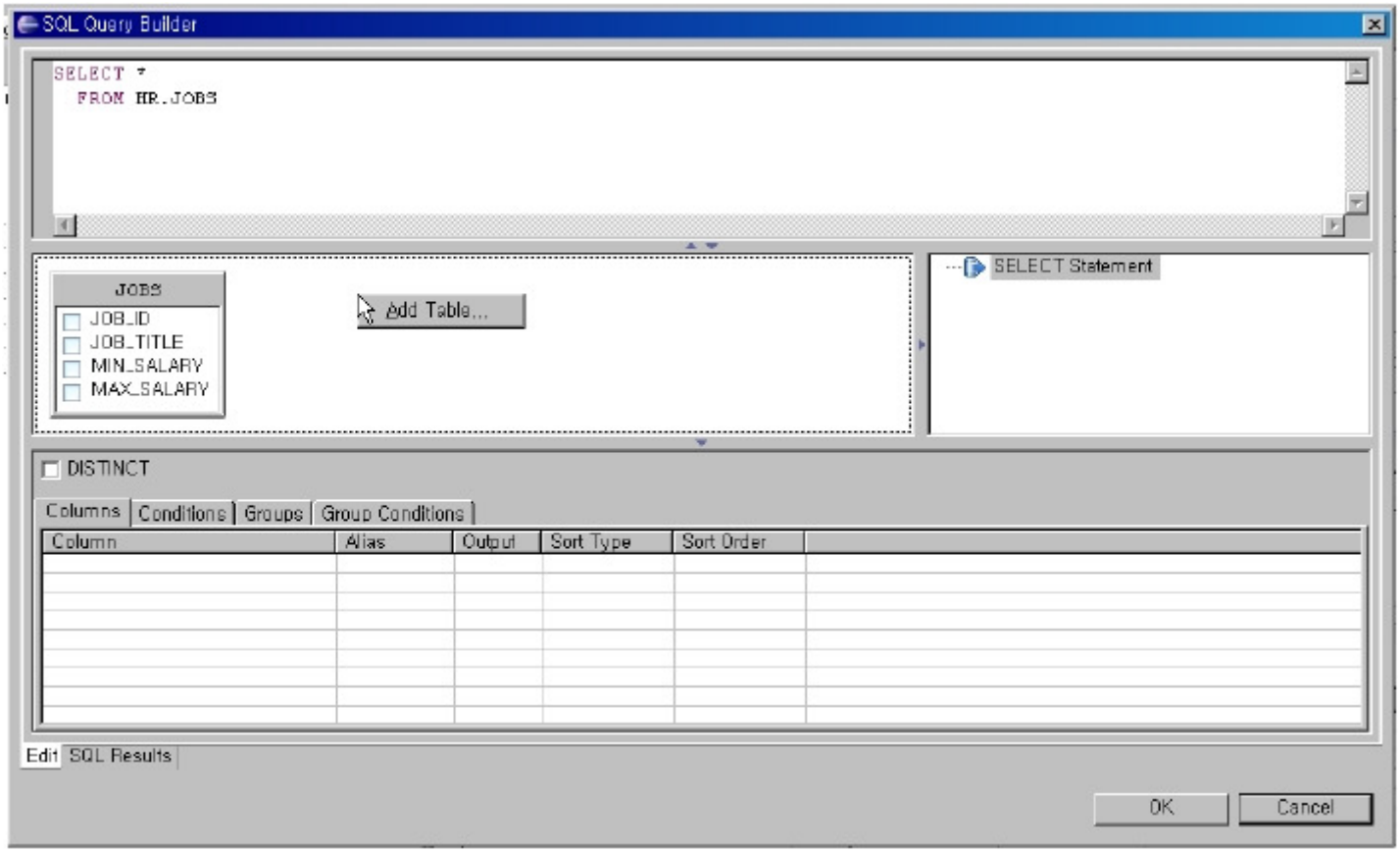


<Figure 8> Editing queries via SQL Map Editor

Using Query Builder

1. Query tab inside the SQLMap Editor's query editor includes the built-in Query Builder feature.
2. In order to use the Query Builder, choose a database connection for the DB Info* item. If a database connection was not setup in advance this item will not be visible.
3. Click on the Open Query Builder right to the DB Info* item to bring up the SQL Query Builder

Query Builder



<Figure 9> Running Query Builder from SQL Map Editor

Adding a table

1. Right-click on the table view will bring up Add Table... menu item.
2. Choose Add Table... to bring up Add Table dialog.
3. Choose a table of view from the dialog.
4. Table alias can be set by using the Table Alias item.
5. Click OK to bring up the schema of the table in the table view, and add the table to the Query Builder's query editor.

Joining tables

1. Add two or more tables in the table view.
2. Click a column, and drag to another column, to connect the columns with a line, and the query editor will add a JOIN statement.
3. Right-click on the JOIN line to set the JOIN type if desired.

Selecting a column

1. Default SELECT query in the Query Builder is SELECT * FROM.
2. Table view will display all the columns that can be used in statements.
3. Checking the check box on the left of columns will tell you which columns can be used in the SELECT statements.
4. Use the column tab on the bottom of Query Builder in order to add Column Aliases if desired.

Where conditional statements

1. Use the Conditions tab on the bottom of Query Builder to build WHERE conditional statements.
2. Choose base name for the column entry, and set Operator and Value entries to create the WHERE conditional statement.
3. To combine multiple WHERE conditional statements, use AND/OR.

Using Group By

1. Use the Groups tab on the bottom of Query Builder to build GROUP BY conditional statements.
2. Select the columns appropriately from the columns.
3. To use HAVING, use the Group Conditions tab. The tab's usage is identical to the Conditions tab.

Using Unions

1. Use the Query Table on the center right of the Query Builder to build UNION conditional statements.
2. Right-click on the Query Tree to bring up the context menu.
3. Choose Convert to FULLSELECT (UNION) to add UNION to the query and also expand the Query Tree.

Running queries

1. Query Builder can also be used to run queries and see the results. Use the editor on the top of the Query Builder..

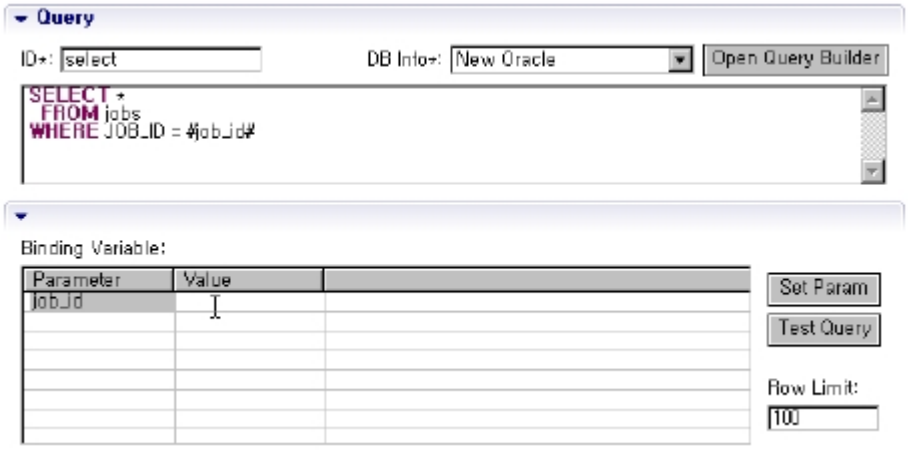
2. Right-click on the editor to bring up the context menu.
3. Select Run SQL in contextual menu. Choose Run SQL.
4. SQL Result tab will display the query result.

Miscellaneous

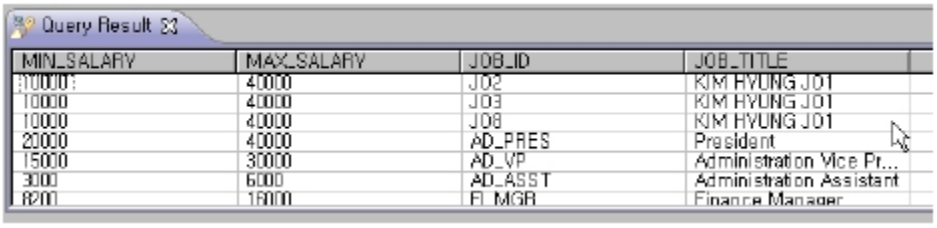
- There are plenty of other features of the Query Builder that are not mentioned in this document.

Testing queries

1. Database connection is required for testing queries, refer to the DBIO: Database Connection Settings guideline.
2. Use the DB Info* item in the Query tab to see if appropriate database connection has been chosen for testing the query.
3. If variables are used in the query, Binding Variables values must be entered in the SQLMap Editor's Test tab. Use # or $ to specify variables. (Column name = #Variable name#, or Column name = $Variable Name$)
4. Clicking on the Set Param button on the right of Binding variables of the Test tab will auto-add variable names to the Parameter entries.
5. Enter appropriate test value in Binding Variables list. (See <Figure 10>)
6. Click the Test Query button on the bottom right to the Binding Variables list, and Result View will automatically show query results. (See <Figure 11>).
7. Row Limit on the right of Binding Variables can be specified to limit the number of results.



<Figure 10> Query Test view of SQL Map Editor



<Figure 11> Query Test results in SQL Map Editor

Available data types for Binding Variables

Type	Description
String	Expresses string text.
Byte	Expresses byte value.
Integer	Expresses integer value (ranging from -2^31 ~ 2^31 -1)
Long	Expresses integer value (ranging from -2^63 ~ 2^63 -1).
Float	Expresses floating point value (ranging from -2^-149 ~ 2-(2^-23) x2^127).
Double	Expresses floating point value (ranging from -2^-1074 ~ 2-(2^-52) x2^1023).
BigDecimal	Expresses decimal value (ranging from unscaledValue x 10^-scale).

※Refer to database manuals to see details of their respective data types.

Testable SQL ("?" binding variable test unavailable)

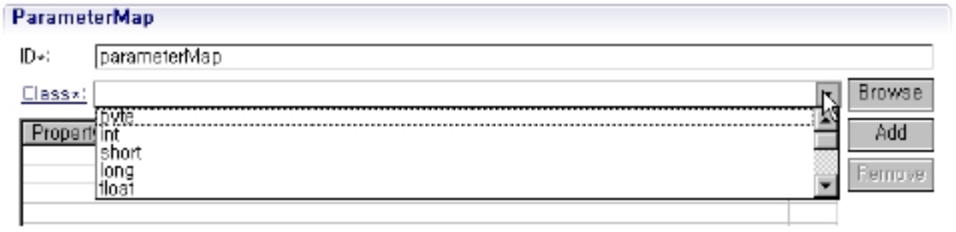
- Using CDATA
  <![CDATA[
  SELECT *
  FROM PERSON
  WHERE AGE > #value#
  ]]>
  Auto-generated key
  <insert id="insertProduct-ORACLE" parameterClass="com.domain.Product">
  <selectKey resultClass="int" keyProperty="id" >

SELECT STOCKIDSEQUENCE.NEXTVAL AS ID FROM DUAL
</selectKey>
insert into PRODUCT (PRD_ID,PRD_DESCRIPTION)
values (#id#,#description#)
</insert>

- Storing procedure
  <procedure id="testInOutProcedure">
  {call INOUT_TEST(#RESULT_MSG,javaType=java.lang.String,jdbcType=VARCHAR,mode=INOUT#, #char_value#)}
  </procedure>
- parameterClass
  <statement id="statementName" parameterClass=" examples.domain.Product">
  insert into PRODUCT values (#id#, #description#, #price#)
  </statement>
- Inline parameter Maps
  <statement id="insertProduct" parameterClass="com.domain.Product">
  insert into PRODUCT (PRD_ID, PRD_DESCRIPTION)
  values (#id:NUMERIC:-999999#, #description:VARCHAR:NO_ENTRY#);
  </statement>
- Dynamically mapped Statements
  <select id="dynamicGetAccountList"
  cacheModel="account-cache"
  resultMap="account-result" >
  select * from ACCOUNT
  <isGreaterThan prepend="and" property="id" compareValue="0">
  where ACC_ID = #id#
  </isGreaterThan>
  order by ACC_LAST_NAME
  </select>
- Binary conditions
  <isEqual>,<isNotEqual>, <isGreaterThan>, <isGreaterEqual>, <isLessThan>, <isLessEqual>
- Unary conditions (partial)
  <isPropertyAvailable>, <isNotPropertyAvailable>, <isNull>, <isNotNull>, <isEmpty>, <isNotEmpty>
- Simple dynamic SQL elemnts
  $binding variable$, #binding variable#

Creating a ParameterMap

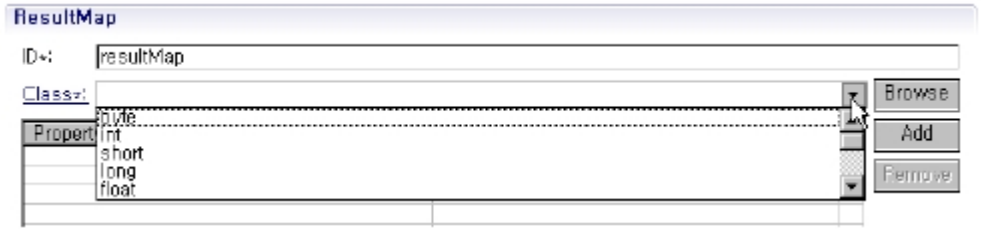
1. Right-click on the SQLMap Tree on the right of SQLMap Editor to bring up the context menu.
2. Choose Add ParameterMap in the context menu.
3. New ParameterMap editor will appear on the right of SQLMap Tree. (See <Figure 12>)
4. Modify the ID or set the Class for ParameterMap. Default Classes will be selectable from the drop-down menu. For custom Classes, click the Browse button to search for the class, or click Class * to create a new class.
5. To add a property, click the Add button on the right of the Property list.
6. To delete a property, choose a property to delete, and click the Remove button on the right.



<Figure 12> ParameterMap editor in SQL Map Editor

Creating a ResultMap

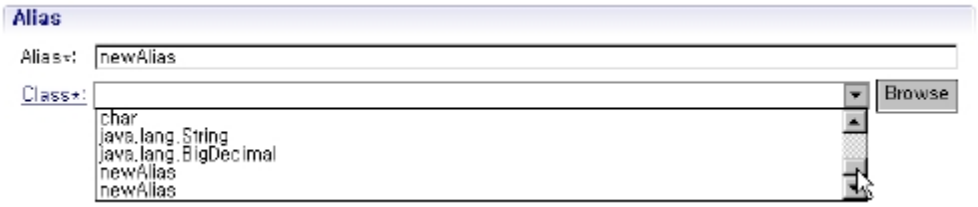
1. Right-click on the SQLMap Tree to bring up the context menu.
2. Choose Add ResultMap.
3. ResultMap editor will appear on the right of SQLMap Tree. (See <Figure 13>)
4. Modify the ID or set the Class for ResultMap. Default Classes will be selectable from the drop-down menu. For custom Classes, click the Browse button to search for the class, or click Class * to create a new class.
5. To add a property, click the Add button on the right of the Property list.
6. To delete a property, choose a property to delete, and click the Remove button on the right.



<Figure 13> Creating ResultMap in SQL Map Editor

Setting Aliases

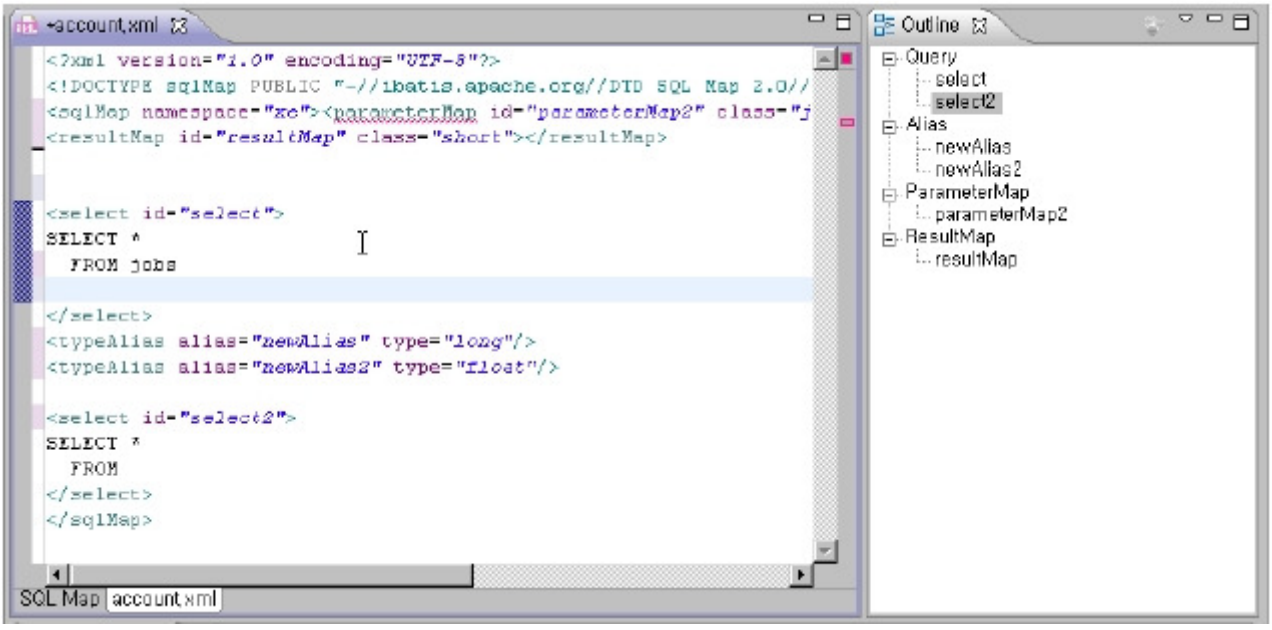
1. Right-click on the SQLMap Tree to bring up the context menu.
2. Choose Add Alias.
3. Alias editor will appear on the right of SQLMap Tree. (See <Figure 14>)
4. Modify the ID or set the Class for Alias. Default Classes will be selectable from the drop-down menu. For custom Classes, click the Browse button to search for the class, or click Class * to create a new class.



<Figure 14> Creating Alias in SQL Map Editor

Manually editing SqlMap.XML source code

1. Click on the [filename].xml tab on the right of the SQL Map tab to manually edit the XML sources.
2. SQLMap Outline will open up to display the SQLMap Tree. (See Figure 15.)



<Figure 15> Editing XML Sources in the SQL Map Editor

---

1)eGovFrame menu will only appear in the eGovFrame Perspective.

2)SQLMap Editor's layout can be set as horizontal or vertical, but this guide assumes the default, which is vertical.