XBLIP OleDB data source driver

Supplied DAIL data source specs

By: Capt. Gal Kahana

Table of Contents:

1. Introduction

DAIL library includes a data source implementation for that uses the .net OleDb components, for connecting an OleDb data base - SimpleOleDBDataSource. The data source implements IDAILDataSourceDriver to retrieve and modify data in OleDB supporting data bases. Apart from just being a data source its may also be used as a supporting data source for more sophisticated data sources.

2. The Track Body Interface

When a data source is used, there has to be an interface that defines what track body is expected. For this driver the expected text is an SQL statement. SimpleOleDBDataSource accepts SQL statements for "insert", "delete" and "update" and in return hands the number of tracks modified. For data retrieve statements the driver accepts a "select" statement in a certain special form. The driver also allows for stored procedure/function calls and bind variables queries.

2.1 A Select Statement

The select statement that is expected is differs from a normal select statement in that it retrieves for each record in the record-set a one or more fields that once concatenated together hold an XML tag that define a single item in a standard response. For example, the following returns a single field for each item that matches the profile:

```
SELECT '<Item id="' || PersonID || '">' ||
                '<FirstName>' || FirstName || '</FirstName>'  ||

                '<LastName>' || LastName || '</LastName>'  ||

        '</Item>'
FROM PeopleList

WHERE department = "Development"
```

The select statement (going to an Oracle database) returns a record-set of all the people whose department is "Development". For each a single field is returned which is the concatenation of the required person details - person ID, first and last names - in the form of a single item in a standard response string. This way, the driver only has to loop the record set and concatenate the strings to form a standard response so that DAIL can work with the result.

Some databases will not be able to support that the entire "Item" field will be returned in a single field. In this case one may split the string to 2 or more fields, and the driver will automatically concatenate those fields. For example:

```
SELECT '<Item id="' || PersonID || '">' ||
                '<FirstName>' || FirstName || '</FirstName>'  as firstField,


                '<LastName>' || LastName || '</LastName>'  ||

        '</Item>' as secondField
FROM PeopleList

WHERE department = "Development"
```

This means the same as the first example.

2.2 Modification Statements

"Insert", "Update" and "Delete" statements are handled regularly, and return the number of modified records (as defined by OleDB interface). When in DAILs modify method this will mean that if this track is the last one, the number is returned in the response.

2.3 Stored Procedures/Functions Statements

To perform a stored procedure/stored function call the track body should contain an XML string that specifies the name of the procedure/function, parameters and return value. For example:

```
DB_FUNC_START
<Function type='procedure'>
<Return type="recordSet" name="result" />
<Param type="integer" name="personID">1</Param>
<Text>SinglePersonSelect</Text>
</Function>
DB_FUNC_END
```

The DB_FUNC_START and DB_FUNC_END labels signal for the driver that this is a stored procedure/function call. This specific call goes to an Access data base.

The "Function" tag encapsulates the definition. It has a single attribute named "type". The type attribute should always be "procedure" for stored procedure/function calls (the other option - "text" is for bind variables queries, and will be discussed later).

The "Return" tag is optional (for voids it would not exist) and defines the return value. It has a "type" attribute which may be one of 4 : interger, double, varChar or recordSet.

When defining a "varChar" type, another attribute has to be added with the name of "size" which should hold the maximum size that is expected to return from the function. recordSet return will tell the driver to transform the returning value to a standard response, from the returned record set. The "name" attribute defines a name for the parameter. If exists the "Return" tag should precede all "Param" tags.

The "Param" tag is optional, and appears for each input parameter for the procedure/function. The param tag content is the value of the parameter.
The "type" attribute defines the type of the parameter. This may be one of integer, double or varChar. The "name" attribute defines a name for the parameter.

The "Text" tag is required and holds the name of the stored procedure/function.

If the DB_FUNC_START and DB_FUNC_END labels are surrounded by text, it is being concatenated (unless the return type is recordSet) to the result of the stored procedure/function call. This way, one can build a standard response from a stored function call so that it may be used as a retrieve track. For example:

```
<Response id="Person" action="Retrieve">
        <Item id="Result">
                <Count>
                        DB_FUNC_START
                        <Function type='procedure'>
                        <Return type="integer" name="result" />
                        <Param type="varChar" name="department">
                                Development
                        </Param>
                        <Text>countPersons</Text>
                        </Function>
                        DB_FUNC_END
                </Count>
        </Item>
</Response>
```

The track will return a standard response with the count of people in the "Development" department.

2.4 Bind Variables Queries

Bind variable queries allow the user to perform a regular statement such as select, insert, update or delete using parameters. This should increase the ability to cache queries and thus increase performance. Bind variable queries as track bodies are a combination of the above said about stored function calls, and simple statements. An Example to a usage of bind variable query with the driver is:

```
DB_FUNC_START
<Function type='text'>
<Param type="varChar" name="department">
        Development
</Param>
<Text>DELETE FROM employees WHERE department =  ?</Text>
</Function>
DB_FUNC_END
```

The above example deletes all employees from a given department. The parameters used for this call is "Development".

One important thing to say about bind variable queries is about the "Return" tag for each one: insert, update and delete do not use the "Return" tag. They automatically return the modified records count. A Select statement always return a "recordSet" which means that a "Return" tag is always required with this type.

3.Usage

There are 2 methods to use the data source.

The first method is as a straight forward data source. Meaning, that from the getDataSource implementation of the project specific DAIL create a SimpleOleDBDataSource object (its constructor gets a connection string) and return it for a certain label.

The second method is to use it from another data source implementation. For example when one wishes to utilize pipes for a certain data source, but still use OleDB as means of acquiring data from the database it may use this class "retrieve" and "modify" methods as DAIL uses them, but for it's own purposes.

4.Summery

The document described the usage of SimpleOleDBDataSource. The data source supports calling SQL statements, Stored procedures/functions and bind variables queries.