Archived:Using Symbian SQL API with SQL statements which do not return data

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Overview

The `RSqlStatement` class provides functionality to execute all types of SQL statements when using the SQL API. `RSqlStatement` encapsulates an SQL statement and the data that it returns. `RSqlStatement` statements can be executed with the methods `Exec()` and `Next()`. The method `Exec()` is used when the statement does not return any data and `Next()` when the statement returns data. This snippet gives some examples on how to use the `Exec()` method.

Statements use so-called placeholders (:) in descriptors to mark positions where dynamic data is inserted at runtime. The user can bind the actual data to placeholders with different `Bind()` methods, such as BindText(). It is good practise to resolve the placeholder index with the `RSqlStatement::ParameterIndex()` method, and not to use hardcoded values.

NOTE: SQL statements can also be executed using the `RSqlDatabase::Exec()` methods, but the behaviour is more limited because the `RSqlDatabase::Exec()` method is not able to return any records.

If asyncronous functionality is required, `RSqlDatabase` and `RSqlStatement` classes have also asyncronous versions of `Exec()` methods.

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY euser.lib
LIBRARY sqldb.lib
```

The following capabilities are needed to test the example secure database:

```
CAPABILITY ReadUserData
CAPABILITY WriteUserData
```

Preconditions

Databases nonsecure.db and [UID3]secure.db need to be created before executing this code snippet. See Archived:Using SQL API for creating non-secure and secure databases on Symbian.

Source file

```
#include <e32base.h>
#include <SqlDb.h>

void ExecStatementsL()
  {
  RSqlDatabase database;

  _LIT(KSecureDbName, "[E80000AF]secure.db");

//== Insert into table statement==
  TInt error = database.Open(KSecureDbName);

  if (error == KErrNone)
    {
    CleanupClosePushL(database);

    RSqlStatement sqlInsertIntoBooksStatement;

    _LIT(KSqlInsertIntoBooksTable, "INSERT INTO BOOKS(ID, TITLE, AUTHOR)
VALUES(:Val1, :Val2, :Val3)");
    TInt ret = sqlInsertIntoBooksStatement.Prepare(database, KSqlInsertIntoBooksTable);

    if(ret == KErrNone)
      {
      CleanupClosePushL(sqlInsertIntoBooksStatement);
```


(c) Copyright Nokia 2013. All rights reserved.

```
      TInt paramIndex1 = sqlInsertIntoBooksStatement.ParameterIndex(_L(":Val1"));
      TInt paramIndex2 = sqlInsertIntoBooksStatement.ParameterIndex(_L(":Val2"));
      TInt paramIndex3 = sqlInsertIntoBooksStatement.ParameterIndex(_L(":Val3"));

      User::LeaveIfError(sqlInsertIntoBooksStatement.BindInt(paramIndex1, 10));
      User::LeaveIfError(sqlInsertIntoBooksStatement.BindText(paramIndex2,
_L("RSqlStatementTitle")));
      User::LeaveIfError(sqlInsertIntoBooksStatement.BindText(paramIndex3,
_L("RSqlStatementAuthor")));

      User::LeaveIfError(sqlInsertIntoBooksStatement.Exec());

      //It is possible to reuse statement after reset call
      User::LeaveIfError(sqlInsertIntoBooksStatement.Reset());
      //Bind new values here...

      CleanupStack::PopAndDestroy(1); //KSqlInsertIntoBooksStatement

      }
    else
      {
      //prepare sql statement failed
      }
    CleanupStack::PopAndDestroy(1); //database
    }
  else
    {
    //open database failed
    }

//== Update statement==

  _LIT(KNonSecureDbName, "\\nonsecure.db");

  error = database.Open(KNonSecureDbName);

  if (error == KErrNone)
    {
    CleanupClosePushL(database);

    RSqlStatement sqlUpdateMoviesRowStatement;

    _LIT(KSqlUpdateMoviesRow, "UPDATE MOVIES SET TITLE=:Val1 WHERE ID=:Val2");
    TInt ret = sqlUpdateMoviesRowStatement.Prepare(database, KSqlUpdateMoviesRow);

    if(ret == KErrNone)
      {
      CleanupClosePushL(sqlUpdateMoviesRowStatement);

      TInt paramIndex1 = sqlUpdateMoviesRowStatement.ParameterIndex(_L(":Val1"));
      TInt paramIndex2 = sqlUpdateMoviesRowStatement.ParameterIndex(_L(":Val2"));

      User::LeaveIfError(sqlUpdateMoviesRowStatement.BindText(paramIndex1,
_L("RSqlStatementUpdatedTitle")));
      User::LeaveIfError(sqlUpdateMoviesRowStatement.BindInt(paramIndex2, 1));

      User::LeaveIfError(sqlUpdateMoviesRowStatement.Exec());

      //It is possible to reuse statement after reset call
      User::LeaveIfError(sqlUpdateMoviesRowStatement.Reset());
      //Bind new values here...

      CleanupStack::PopAndDestroy(1); //sqlUpdateMoviesRowStatement

      }
    else
      {
      //prepare sql statement failed
      }
    CleanupStack::PopAndDestroy(1); //database
    }
  else
    {
    //open database failed
    }

//== Delete statement==

  /* Note: It is possible to use a RSqlStatement object to delete rows from table,
   * but it is computationally more expensive and it is not recommended.
   * Use RSqlDatabase::Exec() method instead */

  error = database.Open(KSecureDbName);

  if (error == KErrNone)
    {
    _LIT(KSqlDeleteFromBooksTable, "DELETE FROM BOOKS WHERE ID > 9;");
    TInt ret = database.Exec(KSqlDeleteFromBooksTable);

    database.Close();
    }
  else
    {
    //open database failed
    }

  }
```

Postconditions

The `RSqlStatement` class is used to insert a new row into the table ([UID3]secure.db) and to update the existing row (nonsecure.db). The `RSqlDatabase` is used to delete one row from the table ([UID3]secure.db).

See also

- Archived:Using SQL API for creating non-secure and secure databases on Symbian
- Archived:Using Symbian SQL API for attaching and detaching databases
- Archived:Using Symbian SQL API with SQL statements which return data

 (C) Copyright Nokia 2013. All rights reserved.

- Archived:Using Symbian SQL API with scalar queries
- Archived:Using Symbian SQL API with data streams

 (c) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.