

SQLite

Non-Visible component

Category	Requires	Version
Storage	API 19, Android 4.4 - 4.4.4 KitKat	1

Overview

A non-visible component that accesses the application's SQLite database.

Component credits: Carlos Pedroza

Events

After Execution

Event handler after the SQL statement is executed, returns whether the execution was succesful.

Params	
was Executed	Boolean

After Query

Event handler after the RawQuery or Query is executed and returns a list with the selected data and number of records.

Params	
result	List
number Of Records	Number

Error Occurred

Event handler when an error occurred, returns a string with a message from the error.

Params	
message	Text

Methods

Clear Database

Clears the database to version 1. Use only while developing, this shouldn't be use on production.

Delete

Returns: *Number*

Executes pre-compiled DELETE statement with specified parameters.

Parameters: 1) String table - Name of the table. 2) String whereClause - Optional WHERE clause to apply when deleting (Example: 'ID = ?'), passing an empty a string will delete all rows. 3) List whereArgs - List with arguments for the WHERE clause. These arguments will be replaced by '?' in the whereClause. Returns the number of rows affected if a whereClause is passed in, 0 otherwise.

Params	
table	Text
where Clause	Text
where Args	List

Get Path

Returns: *Text*

Returns the path to the database

Insert

Returns: *Number*

Executes pre-compiled INSERT statement with specified parameters.

Parameters: 1) String table - Name of the table. 2) YailList columns - List with the columns that will contain the data to be inserted in the database. 3) YailList values - List with the data to be inserted in the database. Returns the row ID of the newly inserted row, or -1 if an error occurred.

Params	
table	Text
columns	List
values	List

Multiple SQL

Execute Multiple SQL Statement asynchronously and returns whether the transaction was successful in the AfterExecution Event Handler. Use it when returned data isn't needed. Parameter: 1) List of SQL.

Params	
list	List

Query

Executes pre-compiled QUERY statement with specified parameters.

Parameters: 1) String table: Name of the table. 2) YailList columns: List of which columns to return, passing an empty list will return all columns. 3) String selection: Filter declaring which rows to return, formatted as an SQL WHERE clause, passing an empty string will return all rows. 4) YailList selectionArgs: List with the arguments that will replace onto '?' in the selection filter. 5) String groupBy: A filter declaring how to group rows, formatted as an SQL GROUP BY clause (excluding the GROUP BY itself), passing an empty string will cause the row to not be grouped. 6) String having: A filter declare which row groups to include if row grouping is being used, passing an empty string will cause all row groups to be included. 7) String orderBy: How to order the rows, formatted as an SQL ORDER BY clause (excluding the ORDER BY itself), passing an empty string will use the default sort order (unordered). 8) String limit: Limits the number of rows returned by the query, formatted as LIMIT clause, passing an empty string denotes no LIMIT clause. The result query is available in the AfterQuery event handler

Params	
table	Text
columns	List
selection	Text
selection Args	List
group By	Text
having	Text
order By	Text
limit	Text

Raw Query

Executes the provided rawQuery Statement asynchronously. Returns a YailList with the selected data and number of records in the AfterQuery Event.

Parameter: 1) String sql. 2) YailList selectionArgs: List with the arguments that will replace '?' in where clause in the query, to prevent SQL injections

Params	
sql	Text
selection Args	List

Replace

Returns: *Number*

Executes pre-compiled REPLACE OR INSERT INTO statement with specified parameters. Parameters: 1) String table - Name of the table. 2) YailList columns -

List with the columns that will contain the data to be replaced in the database. 3) YailList values - List with the data to be replaced in the database. Returns the row ID of the newly replaced row, or -1 if an error occurred.

Params	
table	Text
columns	List
values	List

Single SQL

Execute a Single SQL Statement asynchronously and returns whether the transaction was successful in the AfterExecution Event Handler. Use it when returned data isn't needed. Parameter: 1) String sql.

Params	
sql	Text

Update

Returns: *Number*

Executes pre-compiled UPDATE statement with specified parameters.

Parameters: 1) String table - Name of the table. 2) YailList columns - List with the columns that will contain the data to be inserted in the database. 3) YailList values - List with the data to be inserted in the database. 4) String whereClause - optional WHERE clause to apply when updating, leave an empty string to update all rows. Include ?s, which will be updated by the values from whereArgs. 5) YailList whereArgs - List with the columns that will contain the data to be updated in the database. Returns the row ID of the newly inserted row, or -1 if an error occurred.

Params	
table	Text
columns	List
values	List
where Clause	Text
where Args	List

Properties

Return Header

Boolean Default: *false* — Read Write - Designer Blocks

Returns whether the header row should be returned in the result of a Select statement.

Suppress Toast

Boolean Default: *false* — Read Write - Designer Blocks

Returns whether Success Toast should be suppressed.

Last update: January 26, 2020