

JSTableObject

Method Summary

[JSColumn](#) [#createNewColumn](#)(columnName, type, length, allowNull, pkColumn)
Creates a new column in this table.

void [#deleteColumn](#)(columnName)
Deletes the column with the specified name from this table.

[JSColumn](#) [#getColumn](#)(name)
Returns a JSColumn for the named column.

[String\[\]](#) [#getColumnNames](#)()
Returns an array containing the names of all table columns.

[String](#) [#getDataSource](#)()
Returns the table data source uri.

[String](#) [#getQuotedSQLName](#)()
Returns a quoted version of the table name, if necessary, as defined by the actual database used.

[String\[\]](#) [#getRowIdentifierColumnNames](#)()
Returns an array containing the names of the identifier (PK) column(s).

[String](#) [#getSQLName](#)()
Returns the table name.

[String](#) [#getServerName](#)()
Returns the Servoy server name.

Method Details

[createNewColumn](#)

[JSColumn](#) **createNewColumn**(columnName, type, length, allowNull, pkColumn)

Creates a new column in this table. The name, type and length of the new column must be specified. For specifying the type of the column, use the JSColumn constants. The column is not actually created in the database until this table is synchronized with the database using the JSServer.synchronizeWithDB method. The "allowNull" optional argument specifies if the column accepts null values (by default it does). The "pkColumn" optional argument specifies if the column is a primary key column (by default it is not). The method returns a JSColumn instance that corresponds to the newly created column. If any error occurs and the column cannot be created, then the method returns null.

Parameters

columnName
type
length
allowNull
pkColumn

Returns

[JSColumn](#)

Sample

```
var server = plugins.maintenance.getServer("example_data");
if (server)
{
    var table = server.createNewTable("users");
    if (table)
    {
        table.createNewColumn("id", JSColumn.INTEGER, 0, false, true);
        table.createNewColumn("name", JSColumn.TEXT, 100);
        table.createNewColumn("age", JSColumn.INTEGER, 0);
        table.createNewColumn("last_login", JSColumn.DATETIME, 0);
        var result = server.synchronizeWithDB(table);
        if (result) application.output("Table successfully created.");
        else application.output("Table not created.");
    }
}
```

[deleteColumn](#)

void **deleteColumn**(columnName)

Deletes the column with the specified name from this table. The column is not actually deleted from the database until this table is synchronized with the database using the JSServer.synchronizeWithDB method.

Parameters

{[String](#)} columnName

Returns

void

Sample

```
var server = plugins.maintenance.getServer("example_data");
if (server) {
    var table = server.getTable("users");
    if (table) {
        table.deleteColumn("last_login");
        server.synchronizeWithDB(table);
    }
}
```

getColumn

JSColumn getColumn(name)

Returns a JSColumn for the named column.

Parameters

{String} name – The name of the column to return the value from.

Returns

JSColumn – JSColumn column.

Sample

```
var jsTable = databaseManager.getTable('udm', 'campaigns')
var jsColumn = jsTable.getColumn('campaign_name')
```

getColumnNames

String[] getColumnNames()

Returns an array containing the names of all table columns.

Returns

String[] – String array of column names.

Sample

```
var jsTable = databaseManager.getTable('udm', 'campaigns')
var columnNames = jsTable.getColumnNames()
```

getDataSource

String getDataSource()

Returns the table data source uri.

Returns

String – String datasource uri.

Sample

```
var jsTable = databaseManager.getTable('udm', 'campaigns')
var dataSource = jsTable.getDataSource()
```

getQuotedSQLName

String getQuotedSQLName()

Returns a quoted version of the table name, if necessary, as defined by the actual database used.

Returns

String – String table name, quoted if needed.

Sample

```
//use with the raw SQL plugin:
//if the table name contains characters that are illegal in sql, the table name will be quoted
var jsTable = databaseManager.getTable('udm', 'campaigns')
var quotedTableName = jsTable.getQuotedSQLName()
plugins.rawSQL.executeSQL('udm', quotedTableName, 'select * from ' + quotedTableName + ' where is_active = ?', [1])
```

getRowIdentifierColumnNames

String[] getRowIdentifierColumnNames()

Returns an array containing the names of the identifier (PK) column(s).

Returns

[String](#)[] – String array of row identifier column names.

Sample

```
var jsTable = databaseManager.getTable('udm', 'campaigns')
var identifierColumnNames = jsTable.getRowIdentifierColumnNames()
```

getSQLName

[String](#) **getSQLName()**

Returns the table name.

Returns

[String](#) – String table name.

Sample

```
var jsTable = databaseManager.getTable('udm', 'campaigns')
var tableNameForDisplay = jsTable.getSQLName()
```

getServerName

[String](#) **getServerName()**

Returns the Servoy server name.

Returns

[String](#) – String server name.

Sample

```
var jsTable = databaseManager.getTable('udm', 'campaigns')
var serverName = jsTable.getServerName()
```