QBSelect

Feb 26, 2024 09:05

Supported Clients

SmartClient WebClient NGClient

Property Summary

QBLogicalCondition	and	Create an AND-condition to add conditions to.
QBCase	case	Create an case searched expression.
QBColumns	columns	Get all the columns of the datasource that can be used for this query (select or where clause)
String	comment	Specifies a comment of the query.
QBFunctions	functions	Get the functions clause from a query, used for functions that are not tied to a column.
QBGroupBy	groupBy	Get the group by clause from a query
QBLogicalCondition	having	Get the having-part of the query, used to add conditions.
QBJoins	joins	Get the joins clause of this table based clause.
QBLogicalCondition	or	Create an OR-condition to add conditions to.
QBParameters	params	Get the named parameters from a query.
QBTableClause	parent	Get query builder parent table clause, this may be a query or a join clause.
QBResult	result	Get the result part of the query, used to add result columns or values.
QBSelect	root	Get query builder parent.
QBSorts	sort	Get the sorting part of the query.
QBLogicalCondition	where	Get the where-part of the query, used to add conditions.

Methods Summary

QBSelect	clearHaving()	Clear the having-part of the query.
QBCondition	exists(query)	Get an exists-condition from a subquery
QBColumn	getColumn(name)	Get a column from the table.
QBColumn	getColumn(columnTableAlias, name)	Get a column from the table with given alias.
String	getDataSource()	Returns the datasource for this.
QBParameter	getParameter(name)	Get or create a parameter for the query, this used to parameterize queries.
String	getTableAlias()	Returns the table alias for this.
Object	inline(number)	Create an inlined value.
Object	inline(number, columnForType)	Create an inlined value converted to the type of the column.
Object	inline(string)	Create an inlined (quoted) value.
QBCondition	not(cond)	Create an negated condition.
QBCondition	not(cond)	Create an negated condition.

Property Details

and

```
Create an AND-condition to add conditions to.
```

Returns

QBLogicalCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
query.where.add(
          query.or
            .add(
              query.and
                     .add(query.columns.flag.eq(1))
            .add(query.columns.order_date.isNull)
                 )
            .add(
              query.and
                .add(query.columns.flag.eq(2))
                .add(query.column.order_date.gt(new Date()))
             )
        );
```

case

Create an case searched expression.

Returns

QBCase

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.order_details.createSelect();

// case expressions can be added to the result of the query
        query.result.add(query.case.when(query.columns.quantity.ge(1000)).then('BIG').else('small'));

 // they can also be used in conditions
        query.where.add(query.case
                .when(query.columns.discount.gt(10)).then(50)
                .when(query.columns.quantity.le(20)).then(70)
                .else(100)
        .multiply(query.columns.unitprice).lt(10000));
```

columns

Get all the columns of the datasource that can be used for this query (select or where clause)

Returns

QBColumns

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = foundset.getQuery();
query.result.add(query.columns.name, "name");
query.where.add(query.columns.orderdate.isNull)
```

comment

Specifies a comment of the query.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.comment = 'Query comment'
```

functions

Get the functions clause from a query, used for functions that are not tied to a column.

Returns

QBFunctions

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.where.add(query.columns.shipname.upper.eq(query.functions.upper('servoy'))) //$NON-NLS-1$
foundset.loadRecords(query)
```

groupBy

Get the group by clause from a query

Returns

QBGroupBy

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
.root.having.add(query.joins.orders_to_order_details.columns.quantity.count.eq(0))
foundset.loadRecords(query)
```

having

Get the having-part of the query, used to add conditions.
The conditions added here are AND-ed.

Returns

QBLogicalCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.groupBy.addPk() // have to group by on pk when using having-conditions in (foundset) pk queries
.root.having.add(query.joins.orders_to_order_details.columns.quantity.count.eq(0))
foundset.loadRecords(query)
```

joins

Get the joins clause of this table based clause.
Joins added to this clause will be based on this table clauses table.

Returns

QBJoins

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.getQuery().joins
```

or

Create an OR-condition to add conditions to.

Returns

QBLogicalCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
query.where.add(
          query.or
            .add(
              query.and
                     .add(query.columns.flag.eq(1))
            .add(query.columns.order_date.isNull)
                 )
            .add(
              query.and
                .add(query.columns.flag.eq(2))
                .add(query.column.order_date.gt(new Date()))
             )
        );
```

params

Get the named parameters from a query.

Returns

QBParameters

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
        query.where.add(query.columns.contact_id.eq(query.getParameter('mycontactid')))

        // load orders where contact_id = 100
        query.params['mycontactid'] = 100
        foundset.loadRecords(query)

        // load orders where contact_id = 200
        query.params['mycontactid'] = 200
        foundset.loadRecords(query)
```

parent

Get query builder parent table clause, this may be a query or a join clause.

Returns

QBTableClause

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.person.createSelect();
        query.where.add(query.joins.person_to_parent.joins.person_to_parent.columns.name.eq('john'))
        foundset.loadRecords(query)
```

result

Get the result part of the query, used to add result columns or values.

Returns

QBResult

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
query.result.add(query.columns.company_id).add(query.columns.customerid)
```

root
Get query builder parent.

Returns

QBSelect

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var subquery = datasources.db.example_data.order_details.createSelect();

	var query = datasources.db.example_data.orders.createSelect();
	query.where.add(query
		.or
			.add(query.columns.order_id.not.isin([1, 2, 3]))

			.add(query.exists(
					subquery.where.add(subquery.columns.orderid.eq(query.columns.order_id)).
root
			))
		)

	foundset.loadRecords(query)
```

sort
Get the sorting part of the query.

Returns

QBSorts

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.sort
.add(query.joins.orders_to_order_details.columns.quantity.desc)
.add(query.columns.companyid)
foundset.loadRecords(query)
```

where
Get the where-part of the query, used to add conditions.
The conditions added here are AND-ed.

Returns

QBLogicalCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = foundset.getQuery()
query.where.add(query.columns.flag.eq(1))
```

Methods Details

clearHaving()
Clear the having-part of the query.

Returns

QBSelect

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var q = foundset.getQuery()
q.where.add(q.columns.x.eq(100))
query.groupBy.clear.root.clearHaving()
foundset.loadRecords(q);
```

exists(query)

Get an exists-condition from a subquery

Parameters

Object query the sub query

Returns

QBCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.query.where.add(query.exists(query2))
```

getColumn(name)

Get a column from the table.

Parameters

String name the name of column to get

Returns

QBColumn

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.getQuery().getColumn('orderid')
```

getColumn(columnTableAlias, name)

Get a column from the table with given alias.
The alias may be of the main table or any level deep joined table.

Parameters

String columnTableAlias the alias for the table
String name the name of column to get

Returns

QBColumn

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.getQuery().getColumn('orderid', 'opk')
```

getDataSource()

Returns the datasource for this.

Returns

String the dataSource

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
```

getParameter(name)

Get or create a parameter for the query, this used to parameterize queries.

Parameters

String name the name of the parameter

Returns

QBParameter

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
        query.where.add(query.columns.contact_id.eq(query.getParameter('mycontactid')))

        // load orders where contact_id = 100
        query.params['mycontactid'] = 100
        foundset.loadRecords(query)

        // load orders where contact_id = 200
        query.params['mycontactid'] = 200
        foundset.loadRecords(query)
```

getTableAlias()

Returns the table alias for this.

Returns

String the tableAlias

Supported Clients

SmartClient,WebClient,NGClient

Sample

inline(number)

Create an inlined value. An inlined value is a value that will appear literally in the resulting sql.
For example
<pre>
query.where.add(query.columns.custid.eq(query.inline(200)))
</pre>
results in sql
<pre>
where custid = 200
</pre>
And
<pre>
query.where.add(query.columns.custid.eq(200))
</pre>
results in sql
<pre>
where custid = ?
</pre> with prepared statement value 200.
<p>
Inlined values can be used in situations where prepared statement expressions give sql problems, for example in some group-by clauses.
<p>
Note that using the same query with different inlined values effectively disables prepared statement caching for the query and may have a negative performance impact.
<p>
In case of a string will the value be validated, values that contain a single quote will not be inlined.

Parameters

Number number value to inline

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.order_details.createSelect();
        var mult = query.columns.unitprice.multiply(query.inline(100, query.columns.unitprice));
        query.result.add(mult);
        query.result.add(query.columns.discount.max);
        query.groupBy.add(mult);
```

inline(number, columnForType)

Create an inlined value converted to the type of the column.

Parameters

Number number value to inline
QBColumn columnForType convert value to type of the column

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.order_details.createSelect();
        var mult = query.columns.unitprice.multiply(query.inline(100, query.columns.unitprice));
        query.result.add(mult);
        query.result.add(query.columns.discount.max);
        query.groupBy.add(mult);
```

inline(string)

Create an inlined (quoted) value.

Parameters

String string value to inline

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.order_details.createSelect();
        var mult = query.columns.unitprice.multiply(query.inline(100, query.columns.unitprice));
        query.result.add(mult);
        query.result.add(query.columns.discount.max);
        query.groupBy.add(mult);
```

not(cond)

Create an negated condition.

Parameters

QBCondition cond the condition to negate

Returns

QBCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.query.where.add(query.not(query.columns.flag.eq(1)))
```

not(cond)

Create an negated condition.

Parameters

QBLogicalCondition cond the logical condition to negate

Returns

QBCondition

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.query.where.add(query.not(query.columns.flag.eq(1)))
```