

QBSelect

Property Summary

QBLogicalCondi	and	
on		Create an AND-condition to add conditions to.
QBColumns	columns	Get columns from 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
QBLogicalCondi	having	
on		Get the having-part of the query, used to add conditions.
QBJoins	joins	
		Get the joins clause of this table based clause.
QBLogicalCondi	or	
on		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.
QBWhereCondi	where	
on		Get the where-part of the query, used to add conditions.

Method Summary

QBSelect	clearHaving()	Clear the having-part of the query.
QBCondition	exists()	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.
QBParameter	getParameter()	Get or create a parameter for the query, this used to parameterize queries.
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

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()))
        )
);

```

columns

Get columns from query

Returns

[QBColumns](#)

Sample

```
foundset.getQuery().columns
```

functions

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

Returns

[QBFunctions](#)

Sample

```

/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders') // $NON-NLS-1$
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](#)

Sample

```

/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders')
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](#)

Sample

```
/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders')
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](#)

Sample

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

or

Create an OR-condition to add conditions to.

Returns

[QBLogicalCondition](#)

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](#)

Sample

```
/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders')
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](#)

Sample

```
/** @type {QBSelect<db:/example_data/person>} */
var query = databaseManager.createSelect('db:/example_data/person')
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](#)

Sample

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

root

Get query builder parent.

Returns

[QBSelect](#)

Sample

```
/** @type {QBSelect<db:/example_data/order_details>} */
var subquery = databaseManager.createSelect('db:/example_data/order_details')

/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders')
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](#)

Sample

```
/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders')
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

[QBWhereCondition](#)

Sample

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

Method Details**clearHaving**[QBSelect](#) **clearHaving** ()

Clear the having-part of the query.

Returns[QBSelect](#)**Sample**

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

exists[QBCondition](#) **exists** ()

Get an exists-condition from a subquery

Returns[QBCondition](#)**Sample**

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

getColumn[QBColumn](#) **getColumn** (name)

Get a column from the table.

Parameters

[{String}](#) name - the name of column to get

Returns[QBColumn](#)**Sample**

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

getColumn[QBColumn](#) **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](#)**Sample**

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

getParameter

[QBParameter](#) **getParameter** ()

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

Returns

[QBParameter](#)

Sample

```
/** @type {QBSelect<db:/example_data/orders>} */
var query = databaseManager.createSelect('db:/example_data/orders')
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)
```

not

[QBCondition](#) **not** (cond)

Create an negated condition.

Parameters

{[QBCondition](#)} cond - the condition to negate

Returns

[QBCondition](#)

Sample

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

not

[QBCondition](#) **not** (cond)

Create an negated condition.

Parameters

{[QBLogicalCondition](#)} cond - the logical condition to negate

Returns

[QBCondition](#)

Sample

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