JSFoundSet

Nov 22, 2019 06:08

Supported Clients

SmartClient WebClient NGClient MobileClient

Property Summary

Array	alldataproviders	Get all dataproviders of the foundset.
Boolean	multiSelect	Get or set the multiSelect flag of the foundset.

Methods Summary

Boolean	addFoundSetFilterParam(query)	Add a filter parameter that is permanent per user session to limit a specified foundset of records.
Boolean	addFoundSetFilterParam(query, name)	Add a filter parameter that is permanent per user session to limit a specified foundset of records.
Boolean	addFoundSetFilterParam(dataprovider, operator, value)	Add a filter parameter that is permanent per user session to limit a specified foundset of records.
Boolean	addFoundSetFilterParam(dataprovider, operator, value, name)	Add a filter parameter that is permanent per user session to limit a specified foundset of records.
void	clear()	Clear the foundset.
Boolean	deleteAllRecords()	Delete all records in foundset, resulting in empty foundset.
Boolean	deleteRecord()	Delete currently selected record(s).
Boolean	deleteRecord(record)	Delete record from foundset.
Boolean	deleteRecord(index)	Delete record with the given index.
Boolean	dispose()	Dispose a foundset from memory when foundset is no longer needed.
JSFoundSet	duplicateFoundSet()	Get a duplicate of the foundset.
Number	duplicateRecord()	Duplicate current record, change selection to new record, place on top.
Number	duplicateRecord(onTop)	Duplicate selected record, change selection to new record.
Number	duplicateRecord(onTop, changeSelection)	Duplicate selected record.
Number	duplicateRecord(index)	Duplicate record at index in the foundset, change selection to new record, place on top.
Number	duplicateRecord(index, onTop)	Duplicate record at index in the foundset, change selection to new record.
Number	duplicateRecord(index, onTop, changeSelection)	Duplicate record at index in the foundset.
Number	duplicateRecord(index, location)	Duplicate record at index in the foundset, change selection to new record.
Number	duplicateRecord(index, location, changeSelection)	Duplicate record at index in the foundset.
Boolean	find()	Set the foundset in find mode.
Object	forEach(callback)	Iterates over the records of a foundset taking into account inserts and deletes that may happen at the same time.
Object	forEach(callback, thisObject)	Iterates over the records of a foundset taking into account inserts and deletes that may happen at the same time.
String	getCurrentSort()	Get the current sort columns.
Object	getDataProviderValue(dataProviderID)	Get a value based on a dataprovider name.
String	getDataSource()	Get the datasource used.
Array	getFoundSetFilterParams()	Get the list of previously defined foundset filters.
Array	getFoundSetFilterParams(filterName)	Get a previously defined foundset filter, using its given name.
JSDataSet	getOmittedPKs()	Returns a JSDataSet with the PKs omitted on this foundset If no PKs have been omitted, an empty JSDataSet will be returned
QBSelect	getQuery()	Get the query that the foundset is currently using (as a clone; modifying this QBSelect will not automatically change the foundset).
JSRecord	getRecord(index)	Get the record object at the given index.
Number	getRecordIndex(record)	Get the record index.
String	getRelationName()	Gets the relation name (null if not a related foundset).
Number	getSelectedIndex()	Get the current record index of the foundset.
Array	getSelectedIndexes()	Get the indexes of the selected records.
JSRecord	getSelectedRecord()	Get the selected record.
Array	getSelectedRecords()	Get the selected records.
Number	getSize()	Get the number of records in this foundset.
Boolean	hasConditions()	Check wether the foundset has any conditions from a previous find action.
void	invertRecords()	Invert the foundset against all rows of the current table.
Boolean	isInFind()	Check if this foundset is in find mode.
Boolean	loadAllRecords()	Loads all accessible records from the datasource into the foundset.
Boolean	loadOmittedRecords()	Loads the records that are currently omitted as a foundset.
Boolean	loadRecords()	Reloads all last (related) records again, if, for example, after search in tabpanel.
Boolean	loadRecords(foundset)	Copies foundset data from another foundset.
Boolean	loadRecords(dataset)	Loads a primary key dataset, will remove related sort.

Boolean	loadRecords(querybuilder)	Loads records into form foundset based on a query builder object (also known as 'Form by query').
Boolean	loadRecords(uuidpk)	Loads a single record by primary key, will remove related sort.
Boolean	loadRecords(numberpk)	Loads a single record by primary key, will remove related sort.
Boolean	loadRecords(queryString)	Loads records into form foundset based on a query (also known as 'Form by query').
Boolean	loadRecords(queryString, argumentsArray)	Loads records into form foundset based on a query (also known as 'Form by query').
Number	newRecord()	Create a new record on top of the foundset and change selection to it.
Number	newRecord(onTop)	Create a new record in the foundset and change selection to it.
Number	newRecord(onTop, changeSelection)	Create a new record in the foundset.
Number	newRecord(index)	Create a new record in the foundset and change selection to it.
Number	newRecord(index, changeSelection)	Create a new record in the foundset.
Boolean	omitRecord()	Omit selected record(s) (add it to omit records list), to be shown with loadOmittedRecords.
Boolean	omitRecord(index)	Omit record under the given index (add it to omit records list), to be shown with loadOmittedRecords.
void	relookup()	Perform a relookup for the currently selected records Lookups are defined in the dataprovider (columns) auto-enter setting and are normally performed over a relation upon record creation.
void	relookup(index)	Perform a relookup for the record under the given index Lookups are defined in the dataprovider (columns) auto-enter setting and are normally performed over a relation upon record creation.
Boolean	removeFoundSetFilterParam(name)	Remove a named foundset filter.
Number	search()	Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Number	search(clearLastResults)	Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Number	search(clearLastResults, reduceSearch)	Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Boolean	selectRecord(pkid1, pkid2, pkidn)	Select the record based on pk data.
void	setDataProviderValue(dataProviderID, value)	Set a value based on a dataprovider name.
void	setSelectedIndex(index)	Set the current record index.
void	setSelectedIndexes(indexes)	Set the selected records indexes.
void	sort(sortString)	Sorts the foundset based on the given sort string.
void	sort(sortString, defer)	Sorts the foundset based on the given sort string.
void	sort(recordComparisonFunction)	Sorts the foundset based on the given record comparator function.
JSFoundSet	unrelate()	Create a new unrelated foundset that is a copy of the current foundset.

Property Details

alldataproviders

`Get all dataproviders of the foundset.`

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var dataprovidersNames = alldataproviders;
application.output("This foundset has " + dataprovidersNames.length + " data providers.")
for (var i=0; i<dataprovidersNames.length; i++)
        application.output(dataprovidersNames[i]);
```

multiSelect

`Get or set the multiSelect flag of the foundset.`

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// allow user to select multiple rows.
foundset.multiSelect = true;
```

Methods Details

addFoundSetFilterParam(query)

Add a filter parameter that is permanent per user session to limit a specified foundset of records.

Filters on tables touched in the query will not be applied to the query filter.
For example, when a table filter exists on the order_details table,
a query filter with a join from orders to order_details will be applied to the foundset,
but the filter condition on the orders_details table will not be included.

Use clear() or loadAllRecords() to make the filter effective.
Multiple filters can be added to the same dataprovider, they will all be applied.

Parameters

QBSelect query condition to filter on.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.where.add(
   query.or.add(
            query.columns.shipcity.eq('Amersfoort'))
   .add(    query.columns.shipcity.eq('Amsterdam')));

var success = foundset.addFoundSetFilterParam(query, 'cityFilter'); // possible to add multiple
// Named filters can be removed using foundset.removeFoundSetFilterParam(filterName)

foundset.loadAllRecords(); // to make param(s) effective
```

addFoundSetFilterParam(query, name)

Add a filter parameter that is permanent per user session to limit a specified foundset of records.

Filters on tables touched in the query will not be applied to the query filter.
For example, when a table filter exists on the order_details table,
a query filter with a join from orders to order_details will be applied to the foundset,
but the filter condition on the orders_details table will not be included.

Use clear() or loadAllRecords() to make the filter effective.
The filter is removed again using removeFoundSetFilterParam(name).

The table of the query has to be the same as the foundset table.

Parameters

QBSelect query condition to filter on.

Object name String name, used to remove the filter again.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var query = datasources.db.example_data.orders.createSelect();
query.where.add(
   query.or.add(
            query.columns.shipcity.eq('Amersfoort'))
   .add(    query.columns.shipcity.eq('Amsterdam')));

var success = foundset.addFoundSetFilterParam(query, 'cityFilter'); // possible to add multiple
// Named filters can be removed using foundset.removeFoundSetFilterParam(filterName)

foundset.loadAllRecords(); // to make param(s) effective
```

addFoundSetFilterParam(dataprovider, operator, value)

Add a filter parameter that is permanent per user session to limit a specified foundset of records.
Use clear() or loadAllRecords() to make the filter effective.
Multiple filters can be added to the same dataprovider, they will all be applied.

Parameters

String	dataprovider	String column to filter on.
String	operator	String operator: =, <, >, >=, <=, !=, (NOT) LIKE, (NOT) IN, (NOT) BETWEEN and IS (NOT) NULL optionally augmented with modifiers "#" (ignore case) or "^||" (or-is-null).
Object	value	Object filter value (for in array and between an array with 2 elements)

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var success = foundset.addFoundSetFilterParam('customerid', '=', 'BLONP', 'custFilter');//possible to add
multiple
// Named filters can be removed using foundset.removeFoundSetFilterParam(filterName)

// you can use modifiers in the operator as well, filter on companies where companyname is null or equals-ignore-
case 'servoy'
var ok = foundset.addFoundSetFilterParam('companyname', '#^||=', 'servoy')

// Filters with in-conditions can be used with arrays or with custom queries:
success = foundset.addFoundSetFilterParam("productcode", "in", [120, 144, 200]);
success = foundset.addFoundSetFilterParam("city", "in", ["London", "Paris"]);
success = foundset.addFoundSetFilterParam("countrycode", "in", "select country code from countries where region
in ('Europe', 'Asia')");

foundset.loadAllRecords();//to make param(s) effective

// see https://wiki.servoy.com/display/DOCS/Using+Table+Filters
```

addFoundSetFilterParam(dataprovider, operator, value, name)

```
Add a filter parameter that is permanent per user session to limit a specified foundset of records.
Use clear() or loadAllRecords() to make the filter effective.
The filter is removed again using removeFoundSetFilterParam(name).
```

Parameters

String	dataprovider	String column to filter on.
String	operator	String operator: =, <, >, >=, <=, !=, (NOT) LIKE, (NOT) IN, (NOT) BETWEEN and IS (NOT) NULL optionally augmented with modifiers "#" (ignore case) or "^||" (or-is-null).
Object	value	Object filter value (for in array and between an array with 2 elements)
String	name	String name, used to remove the filter again.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var success = foundset.addFoundSetFilterParam('customerid', '=', 'BLONP', 'custFilter');//possible to add
multiple
// Named filters can be removed using foundset.removeFoundSetFilterParam(filterName)

// you can use modifiers in the operator as well, filter on companies where companyname is null or equals-ignore-
case 'servoy'
var ok = foundset.addFoundSetFilterParam('companyname', '#^||=', 'servoy')

// Filters with in-conditions can be used with arrays or with custom queries:
success = foundset.addFoundSetFilterParam("productcode", "in", [120, 144, 200]);
success = foundset.addFoundSetFilterParam("city", "in", ["London", "Paris"]);
success = foundset.addFoundSetFilterParam("countrycode", "in", "select country code from countries where region
in ('Europe', 'Asia')");

foundset.loadAllRecords();//to make param(s) effective

// see https://wiki.servoy.com/display/DOCS/Using+Table+Filters
```

clear()

Clear the foundset.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//Clear the foundset, including searches that may be on it
foundset.clear();
```

deleteAllRecords()

Delete all records in foundset, resulting in empty foundset.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var success = foundset.deleteAllRecords();
```

deleteRecord()

Delete currently selected record(s).
If the foundset is in multiselect mode, all selected records are deleted.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var success = foundset.deleteRecord();
//can return false incase of related foundset having records and orphans records are not allowed by the relation
```

deleteRecord(record)

Delete record from foundset.

Parameters

JSRecord record The record to delete from the foundset.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var success = foundset.deleteRecord(rec);
//can return false incase of related foundset having records and orphans records are not allowed by the relation
```

deleteRecord(index)

Delete record with the given index.

Parameters

Number index The index of the record to delete.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var success = foundset.deleteRecord(4);
//can return false incase of related foundset having records and orphans records are not allowed by the relation
```

dispose()

Dispose a foundset from memory when foundset is no longer needed. Should be used to destroy separate foundsets (is an optimization for memory management).
A related foundset or a foundset which is linked to visible forms/components cannot be disposed. Returns whether foundset was disposed or not.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
foundset.dispose();
```

duplicateFoundSet()

Get a duplicate of the foundset.

Returns

JSFoundSet

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var dupFoundset = foundset.duplicateFoundSet();
foundset.find();
//search some fields
var count = foundset.search();
if (count == 0)
{
        plugins.dialogs.showWarningDialog('Alert', 'No records found','OK');
        foundset.loadRecords(dupFoundset);
}
```

duplicateRecord()

Duplicate current record, change selection to new record, place on top.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(onTop)

`Duplicate selected record, change selection to new record.`

Parameters

Boolean onTop when true the new record is added as the topmost record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(onTop, changeSelection)

`Duplicate selected record.`

Parameters

Boolean onTop when true the new record is added as the topmost record.
Boolean changeSelection when true the selection is changed to the duplicated record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(index)

`Duplicate record at index in the foundset, change selection to new record, place on top.`

Parameters

Number index The index of the record to duplicate; defaults to currently selected index. Ignored if first given parameter is a boolean value.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(index, onTop)

Duplicate record at index in the foundset, change selection to new record.

Parameters

Number index The index of the record to duplicate; defaults to currently selected index. Ignored if first given parameter is a boolean value.
Boolean onTop when true the new record is added as the topmost record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(index, onTop, changeSelection)

Duplicate record at index in the foundset.

Parameters

Number index The index of the record to duplicate; defaults to currently selected index. Ignored if first given parameter is a boolean value.
Boolean onTop when true the new record is added as the topmost record.
Boolean changeSelection when true the selection is changed to the duplicated record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(index, location)

Duplicate record at index in the foundset, change selection to new record.

Parameters

Number index The index of the record to duplicate; defaults to currently selected index. Ignored if first given parameter is a boolean value.
Number location the new record is added at specified index

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

duplicateRecord(index, location, changeSelection)

Duplicate record at index in the foundset.

Parameters

Number	index	The index of the record to duplicate; defaults to currently selected index. Ignored if first given parameter is a boolean value.
Number	location	the new record is added at specified index
Boolean	changeSelection	when true the selection is changed to the duplicated record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.duplicateRecord();
foundset.duplicateRecord(false); //duplicate the current record, adds at bottom
foundset.duplicateRecord(1,2); //duplicate the first record as second record
//duplicates the record (record index 3), adds on top and selects the record
foundset.duplicateRecord(3,true,true);
```

find()

```
Set the foundset in find mode. (Start a find request), use the "search" function to perform/exit the find.

Before going into find mode, all unsaved records will be saved in the database.
If this fails (due to validation failures or sql errors) or is not allowed (autosave off), the foundset will
not go into find mode.
Make sure the operator and the data (value) are part of the string passed to dataprovider (included inside a
pair of quotation marks).
Note: always make sure to check the result of the find() method.

When in find mode, columns can be assigned string expressions (including operators) that are evaluated as:
General:
      c1||c2    (condition1 or condition2)
      c|format  (apply format on condition like 'x|dd-MM-yyyy')
      !c        (not condition)
      #c        (modify condition, depends on column type)
      ^         (is null)
      ^=        (is null or empty)
      &lt;x     (less than value x)
      &gt;x     (greater than value x)
      &lt;=x    (less than or equals value x)
      &gt;=x    (greater than or equals value x)
      x...y     (between values x and y, including values)
      x         (equals value x)

 Number fields:
      =x       (equals value x)
      ^=       (is null or zero)

 Date fields:
      #c       (equals value x, entire day)
      now      (equals now, date and or time)
      //       (equals today)
      today    (equals today)

 Text fields:
      #c                (case insensitive condition)
      = x      (equals a space and 'x')
      ^=       (is null or empty)
      %x%      (contains 'x')
      %x_y%    (contains 'x' followed by any char and 'y')
      \%      (contains char '%')
      \_      (contains char '_')

Related columns can be assigned, they will result in related searches.
For example, "employees_to_department.location_id = headoffice" finds all employees in the specified location).

Searching on related aggregates is supported.
For example, "orders_to_details.total_amount = '&gt;1000'" finds all orders with total order details amount
more than 1000.

Arrays can be used for searching a number of values, this will result in an 'IN' condition that will be used in
the search.
The values are not restricted to strings but can be any type that matches the column type.
For example, "record.department_id = [1, 33, 99]"
```

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
if (foundset.find()) //find will fail if autosave is disabled and there are unsaved records
{
        columnTextDataProvider = 'a search value'
        // for numbers you have to make sure to format it correctly so that the decimal point is in your locales
notation (. or ,)
        columnNumberDataProvider = '>' + utils.numberFormat(anumber, '####.00');
        columnDateDataProvider = '31-12-2010|dd-MM-yyyy'
        foundset.search()
}
```

forEach(callback)

Iterates over the records of a foundset taking into account inserts and deletes that may happen at the same time.
It will dynamically load all records in the foundset (using Servoy lazy loading mechanism). If callback function returns a non null value the traversal will be stopped and that value is returned.
If no value is returned all records of the foundset will be traversed. Foundset modifications(like sort, omit...) cannot be performed in the callback function.
If foundset is modified an exception will be thrown. This exception will also happen if a refresh happens because of a rollback call for records on this datasource when iterating.
When an exception is thrown from the callback function, the iteraion over the foundset will be stopped.

Parameters

Function	callback	The callback function to be called for each loaded record in the foundset. Can receive three parameters: the record to be processed, the index of the record in the foundset, and the foundset that is traversed.

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.forEach(function(record,recordIndex,foundset) {
        //handle the record here
 });
```

forEach(callback, thisObject)

Iterates over the records of a foundset taking into account inserts and deletes that may happen at the same time.
It will dynamically load all records in the foundset (using Servoy lazy loading mechanism). If callback function returns a non null value the traversal will be stopped and that value is returned.
If no value is returned all records of the foundset will be traversed. Foundset modifications(like sort, omit...) cannot be performed in the callback function.
If foundset is modified an exception will be thrown. This exception will also happen if a refresh happens because of a rollback call for records on this datasource when iterating.
When an exception is thrown from the callback function, the iteraion over the foundset will be stopped.

Parameters

Function	callback	The callback function to be called for each loaded record in the foundset. Can receive three parameters: the record to be processed, the index of the record in the foundset, and the foundset that is traversed.
Object	thisObject	What the this object should be in the callback function (default it is the foundset)

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.forEach(function(record,recordIndex,foundset) {
        //handle the record here
 });
```

getCurrentSort()

Get the current sort columns.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//reverse the current sort

//the original sort "companyName asc, companyContact desc"
//the inversed sort "companyName desc, companyContact asc"
var foundsetSort = foundset.getCurrentSort()
var sortColumns = foundsetSort.split(',')
var newFoundsetSort = ''
for(var i=0; i<sortColumns.length; i++)
{
        var currentSort = sortColumns[i]
        var sortType = currentSort.substring(currentSort.length-3)
        if(sortType.equalsIgnoreCase('asc'))
        {
                newFoundsetSort += currentSort.replace(' asc', ' desc')
        }
        else
        {
                newFoundsetSort += currentSort.replace(' desc', ' asc')
        }
        if(i != sortColumns.length - 1)
        {
                newFoundsetSort += ','
        }
}
foundset.sort(newFoundsetSort)
```

getDataProviderValue(dataProviderID)

Get a value based on a dataprovider name.

Parameters

String dataProviderID data provider name

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var val = foundset.getDataProviderValue('contact_name');
```

getDataSource()

Get the datasource used.
The datasource is an url that describes the data source.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var dataSource = foundset.getDataSource();
```

getFoundSetFilterParams()

Get the list of previously defined foundset filters.

For column-based table filters, a row of 5 fields per filter are returned.
The "columns" of a row from this array are: tablename, dataprovider, operator, value, filtername

For query-based filters, a row of 2 fields per filter are returned.
The "columns" of a row from this array are: query, filtername

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var params = foundset.getFoundSetFilterParams()
for (var i = 0; params != null && i < params.length; i++)
{
 if (params[i].length() == 5) {
                application.output('FoundSet filter on table ' + params[i][0] + ': '+ params[i][1] + ' '+params
[i][2] + ' '+params[i][3] + (params[i][4] == null ? ' [no name]' : ' ['+params[i][4]+']'))
        }
 if (params[i].length() == 2) {
                application.output('FoundSet filter with query ' + params[i][0]+ ': ' + (params[i][1] == null ?
' [no name]' : ' ['+params[i][1]+']'))
        }
}
```

getFoundSetFilterParams(filterName)

Get a previously defined foundset filter, using its given name.
The result is an array of:
 [tableName, dataprovider, operator, value, name]

Parameters

String filterName name of the filter to retrieve.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var params = foundset.getFoundSetFilterParams()
for (var i = 0; params != null && i < params.length; i++)
{
        application.output('FoundSet filter on table ' + params[i][0]+ ': '+ params[i][1]+ ' '+params[i][2]+ '
'+params[i][3] +(params[i][4] == null ? ' [no name]' : ' ['+params[i][4]+']'))
}
```

getOmittedPKs()

Returns a JSDataSet with the PKs omitted on this foundset
If no PKs have been omitted, an empty JSDataSet will be returned

Returns

JSDataSet

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.getOmittedPKs();
```

getQuery()

Get the query that the foundset is currently using (as a clone; modifying this QBSelect will not automatically
change the foundset).
When the founset is in find mode, the find conditions are included in the resulting query.
So the query that would be used when just calling search() (or search(true,true)) is returned.
Note that foundset filters are included and table filters are not included in the query.

Returns

QBSelect

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var q = foundset.getQuery()
q.where.add(q.columns.x.eq(100))
foundset.loadRecords(q);
```

getRecord(index)

Get the record object at the given index.
Argument "index" is 1 based (so first record is 1).

Parameters

Number index record index (1 based).

Returns

JSRecord

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var record = foundset.getRecord(index);
```

getRecordIndex(record)

Get the record index. Will return -1 if the record can't be found.

Parameters

JSRecord record Record

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var index = foundset.getRecordIndex(record);
```

getRelationName()

Gets the relation name (null if not a related foundset).

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var relName = foundset.getRelationName();
```

getSelectedIndex()

Get the current record index of the foundset.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
//gets the current record index in the current foundset
var current = foundset.getSelectedIndex();
//sets the next record in the foundset
foundset.setSelectedIndex(current+1);
```

getSelectedIndexes()

Get the indexes of the selected records.
When the founset is in multiSelect mode (see property multiSelect), a selection can consist of more than one index.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// modify selection to the first selected item and the following row only
var current = foundset.getSelectedIndexes();
if (current.length > 1)
{
        var newSelection = new Array();
        newSelection[0] = current[0]; // first current selection
        newSelection[1] = current[0] + 1; // and the next row
        foundset.setSelectedIndexes(newSelection);
}
```

getSelectedRecord()

Get the selected record.

Returns

JSRecord

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var selectedRecord = foundset.getSelectedRecord();
```

getSelectedRecords()

Get the selected records.
When the founset is in multiSelect mode (see property multiSelect), selection can be a more than 1 record.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var selectedRecords = foundset.getSelectedRecords();
```

getSize()

Get the number of records in this foundset.
This is the number of records loaded, note that when looping over a foundset, size() may increase as more records are loaded.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var nrRecords = foundset.getSize()

// to loop over foundset, recalculate size for each record
for (var i = 1; i <= foundset.getSize(); i++)
{
        var rec = foundset.getRecord(i);
}
```

hasConditions()

Check wether the foundset has any conditions from a previous find action.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
if (foundset.hasConditions())
{
                // foundset had find actions
}
```

invertRecords()

Invert the foundset against all rows of the current table.
All records that are not in the foundset will become the current foundset.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.invertRecords();
```

isInFind()

Check if this foundset is in find mode.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
//Returns true when find was called on this foundset and search has not been called yet
foundset.isInFind();
```

loadAllRecords()

Loads all accessible records from the datasource into the foundset.
Filters on the foundset are applied.

Before loading the records, all unsaved records will be saved in the database.
If this fails (due to validation failures or sql errors) or is not allowed (autosave off),
records will not be loaded,

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
foundset.loadAllRecords();
```

loadOmittedRecords()

Loads the records that are currently omitted as a foundset.

Before loading the omitted records, all unsaved records will be saved in the database.
If this fails (due to validation failures or sql errors) or is not allowed (autosave off),
omitted records will not be loaded,

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.loadOmittedRecords();
```

loadRecords()

Reloads all last (related) records again, if, for example, after search in tabpanel.
When in find mode, this will reload the records from before the find() call.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//to reload all last (related) records again, if for example when searched in tabpanel
 foundset.loadRecords();
```

loadRecords(foundset)

Copies foundset data from another foundset.
This will alter the foundset state to the state of the foundset that is given.
If you really just want to use the given foundset on the form itself, then you need to use controller.loadRecords(foundset)
that will change the instance of the foundset that is used for this form. Not just update an existing form.

If you copy over a relation into this foundset, then this foundset will not be a related foundset, it will not automatically update its state
of records are updated or added that belong to that relation. It will only be a snapshot of that related foundsets state.

Foundset filter params are copied over from the source foundset and are merged with the existing filters on this foundset.

Parameters

JSFoundSet foundset The foundset to load records from

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//Copies foundset data from another foundset
foundset.loadRecords(fs);
```

loadRecords(dataset)

Loads a primary key dataset, will remove related sort. Tries to preserve selection based on primary key, otherwise first record is selected.

Parameters

JSDataSet dataset pkdataset

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// loads a primary key dataset, will remove related sort!
//var dataset = databaseManager.getDataSetByQuery(...);
// dataset must match the table primary key columns (alphabetically ordered)
foundset.loadRecords(dataset);
```

loadRecords(querybuilder)

```
Loads records into form foundset based on a query builder object (also known as 'Form by query').
When the founset is in find mode, the find states are discarded, the foundset will go out of find mode and the
foundset will be loaded using the query.
If the foundset is related, the relation-condition will be added to the query.
Tries to preserve selection based on primary key, otherwise first record is selected.
```

Parameters

QBSelect querybuilder the query builder

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.loadRecords(qbselect);
```

loadRecords(uuidpk)

```
Loads a single record by primary key, will remove related sort.

NOTE: This function will return true if the foundset was altered/changed. It is up to the developer to check
for the presence of actual data using getSize().
```

Parameters

UUID uuidpk single-column pk value

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//Loads a single record by primary key, will remove related sort!
foundset.loadRecords(application.getUUID('6b5e2f5d-047e-45b3-80ee-3a32267b1f20'));
```

loadRecords(numberpk)

```
Loads a single record by primary key, will remove related sort.

NOTE: This function will return true if the foundset was altered/changed. It is up to the developer to check
for the presence of actual data using getSize().
```

Parameters

Number numberpk single-column pk value

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//Loads a single record by primary key, will remove related sort!
foundset.loadRecords(123);
```

loadRecords(queryString)

Loads records into form foundset based on a query (also known as 'Form by query'). The query must be a valid sql select.
If the foundset is related this function is not allowed.
Tries to preserve selection based on primary key, otherwise first record is selected.

see foundset.loadRecords(QBSelect).

When possible, the foundset will be loaded with the given query.
This is not always possible because the foundset needs to manipulate the query when adding conditions and joins.
In that case the query will be wrapped: select pk from tab where pk = (queryString)
The result is the same, except for the ordering in the queryString which will be ignored.

The query will be wrapped when one of the following is true:

you have no order-by clause
you have no from keyword
your query is not fully qualified on the main table
you have a group-by, having, join or union keyword

Parameters

String queryString select statement

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//loads records in to the foundset based on a query (also known as 'Form by query')
foundset.loadRecords(sqlstring);
```

loadRecords(queryString, argumentsArray)

Loads records into form foundset based on a query (also known as 'Form by query'). The query must be a valid sql select.
If the foundset is related this function is not allowed.
Tries to preserve selection based on primary key, otherwise first record is selected.

see foundset.loadRecords(QBSelect).

When possible, the foundset will be loaded with the given query.
This is not always possible because the foundset needs to manipulate the query when adding conditions and joins.
In that case the query will be wrapped: select pk from tab where pk = (queryString)
The result is the same, except for the ordering in the queryString which will be ignored.

The query will be wrapped when one of the following is true:

you have no order-by clause
you have no from keyword
your query is not fully qualified on the main table
you have a group-by, having, join or union keyword

Parameters

String queryString select statement
Array argumentsArray arguments to query

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//loads records in to the foundset based on a query (also known as 'Form by query')
foundset.loadRecords(sqlstring,parameters);
```

newRecord()

Create a new record on top of the foundset and change selection to it. Returns -1 if the record can't be made.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
// foreign key data is only filled in for equals (=) relation items
var idx = foundset.newRecord(false); // add as last record
// foundset.newRecord(); // adds as first record
// foundset.newRecord(2); //adds as second record
if (idx >= 0) // returned index is -1 in case of failure
{
        foundset.some_column = "some text";
        application.output("added on position " + idx);
        // when adding at the end of the foundset, the returned index
        // corresponds with the size of the foundset
}
```

newRecord(onTop)

Create a new record in the foundset and change selection to it. Returns -1 if the record can't be made.

Parameters

Boolean onTop when true the new record is added as the topmost record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// foreign key data is only filled in for equals (=) relation items
var idx = foundset.newRecord(false); // add as last record
// foundset.newRecord(); // adds as first record
// foundset.newRecord(2); //adds as second record
if (idx >= 0) // returned index is -1 in case of failure
{
        foundset.some_column = "some text";
        application.output("added on position " + idx);
        // when adding at the end of the foundset, the returned index
        // corresponds with the size of the foundset
}
```

newRecord(onTop, changeSelection)

Create a new record in the foundset. Returns -1 if the record can't be made.

Parameters

Boolean	onTop	when true the new record is added as the topmost record; when false the record is added to the end, if all records are loaded, otherwise it will be added to the top
Boolean	changeSelection	when true the selection is changed to the new record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// foreign key data is only filled in for equals (=) relation items
var idx = foundset.newRecord(false); // add as last record
// foundset.newRecord(); // adds as first record
// foundset.newRecord(2); //adds as second record
if (idx >= 0) // returned index is -1 in case of failure
{
        foundset.some_column = "some text";
        application.output("added on position " + idx);
        // when adding at the end of the foundset, the returned index
        // corresponds with the size of the foundset
}
```

newRecord(index)

Create a new record in the foundset and change selection to it. Returns -1 if the record can't be made.

Parameters

Number index the new record is added at specified index.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// foreign key data is only filled in for equals (=) relation items
var idx = foundset.newRecord(false); // add as last record
// foundset.newRecord(); // adds as first record
// foundset.newRecord(2); //adds as second record
if (idx >= 0) // returned index is -1 in case of failure
{
        foundset.some_column = "some text";
        application.output("added on position " + idx);
        // when adding at the end of the foundset, the returned index
        // corresponds with the size of the foundset
}
```

newRecord(index, changeSelection)

Create a new record in the foundset. Returns -1 if the record can't be made.

Parameters

Number index the new record is added at specified index.
Boolean changeSelection when true the selection is changed to the new record.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
// foreign key data is only filled in for equals (=) relation items
var idx = foundset.newRecord(false); // add as last record
// foundset.newRecord(); // adds as first record
// foundset.newRecord(2); //adds as second record
if (idx >= 0) // returned index is -1 in case of failure
{
        foundset.some_column = "some text";
        application.output("added on position " + idx);
        // when adding at the end of the foundset, the returned index
        // corresponds with the size of the foundset
}
```

omitRecord()

Omit selected record(s) (add it to omit records list), to be shown with loadOmittedRecords. T
his operation returns false only when foundset is in bad state (table not accessible or not having a valid
selected record)
or the record is in an edit state and can't be saved (autosave is false).

Note: The omitted records list is discarded when these functions are executed: loadAllRecords, loadRecords
(dataset), loadRecords(sqlstring), invertRecords()

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var success = foundset.omitRecord();
```

omitRecord(index)

Omit record under the given index (add it to omit records list), to be shown with loadOmittedRecords. If index
is null it behaves just like omitRecord().
This operation returns false when index is invalid (should be between 1 and foundset size) or foundset is in
bad state (its table not accessible)
or the record is in an edit state and can't be saved (autosave is false). Any retrievable record can be
ommitted.

Note: The omitted records list is discarded when these functions are executed: loadAllRecords, loadRecords
(dataset), loadRecords(sqlstring), invertRecords()

Parameters

Number index The index of the record to omit, starting with 1 .

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var success = foundset.omitRecord();
```

relookup()

Perform a relookup for the currently selected records
Lookups are defined in the dataprovider (columns) auto-enter setting and are normally performed over a relation
upon record creation.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.relookup(1);
```

relookup(index)

Perform a relookup for the record under the given index
Lookups are defined in the dataprovider (columns) auto-enter setting and are normally performed over a relation
upon record creation.

Parameters

Number index record index (1-based)

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.relookup(1);
```

removeFoundSetFilterParam(name)

Remove a named foundset filter.
Use clear() or loadAllRecords() to make the filter effective.

Parameters

String name String filter name.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var success = foundset.removeFoundSetFilterParam('custFilter');// removes all filters with this name
foundset.loadAllRecords();//to make param(s) effective
```

search()

Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Clear results from previous searches.

Note: Omitted records are automatically excluded when performing a search - meaning that the foundset result by default will not include omitted records.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var recordCount = foundset.search();
//var recordCount = foundset.search(false,false); //to extend foundset
```

search(clearLastResults)

Start the database search and use the results, returns the number of records, make sure you did "find" function first.
Reduce results from previous searches.

Note: Omitted records are automatically excluded when performing a search - meaning that the foundset result by default will not include omitted records.

Parameters

Boolean clearLastResults boolean, clear previous search, default true

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var recordCount = foundset.search();
//var recordCount = foundset.search(false,false); //to extend foundset
```

search(clearLastResults, reduceSearch)

Start the database search and use the results, returns the number of records, make sure you did "find" function first.

Note: Omitted records are automatically excluded when performing a search - meaning that the foundset result by default will not include omitted records.

Parameters

Boolean clearLastResults boolean, clear previous search, default true
Boolean reduceSearch boolean, reduce (true) or extend (false) previous search results, default true

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var recordCount = foundset.search();
//var recordCount = foundset.search(false,false); //to extend foundset
```

selectRecord(pkid1, pkid2, pkidn)

```
Select the record based on pk data.
Note that if the foundset has not loaded the record with the pk, selectrecord will fail.

In case of a table with a composite key, the pk sequence must match the alphabetical
ordering of the pk column names.
```

Parameters

Object pkid1 primary key
Object pkid2 second primary key (in case of composite primary key)
Object pkidn nth primary key

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.selectRecord(pkid1,pkid2,pkidn);//pks must be alphabetically set! It is also possible to use an array
as parameter.
```

setDataProviderValue(dataProviderID, value)

```
Set a value based on a dataprovider name.
```

Parameters

String dataProviderID data provider name
Object value value to set

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.setDataProviderValue('contact_name','mycompany');
```

setSelectedIndex(index)

```
Set the current record index.
```

Parameters

Number index index to set (1-based)

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
//gets the current record index in the current foundset
var current = foundset.getSelectedIndex();
//sets the next record in the foundset
foundset.setSelectedIndex(current+1);
```

setSelectedIndexes(indexes)

```
Set the selected records indexes.
```

Parameters

Array indexes An array with indexes to set.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// modify selection to the first selected item and the following row only
var current = foundset.getSelectedIndexes();
if (current.length > 1)
{
        var newSelection = new Array();
        newSelection[0] = current[0]; // first current selection
        newSelection[1] = current[0] + 1; // and the next row
        foundset.setSelectedIndexes(newSelection);
}
```

sort(sortString)

```
Sorts the foundset based on the given sort string.
Tries to preserve selection based on primary key. If first record is selected or cannot select old record it
will select first record after sort.
TIP: You can use the Copy button in the developer Select Sorting Fields dialog to get the needed syntax string
for the desired sort fields/order.
```

Parameters

String sortString the specified columns (and sort order)

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.sort('columnA desc,columnB asc');
```

sort(sortString, defer)

```
Sorts the foundset based on the given sort string.
Tries to preserve selection based on primary key. If first record is selected or cannot select old record it
will select first record after sort.
TIP: You can use the Copy button in the developer Select Sorting Fields dialog to get the needed syntax string
for the desired sort fields/order.
```

Parameters

String	sortString	the specified columns (and sort order)
Boolean	defer	when true, the "sortString" will be just stored, without performing a query on the database (the actual sorting will be deferred until the next data loading action).

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.sort('columnA desc,columnB asc');
```

sort(recordComparisonFunction)

```
Sorts the foundset based on the given record comparator function.
Tries to preserve selection based on primary key. If first record is selected or cannot select old record it
will select first record after sort.
The comparator function is called to compare
two records, that are passed as arguments, and
it will return -1/0/1 if the first record is less/equal/greater
then the second record.

The function based sorting does not work with printing.
It is just a temporary in-memory sort.

NOTE: starting with 7.2 release this function doesn't save the data anymore
```

Parameters

Function recordComparisonFunction record comparator function

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
foundset.sort(mySortFunction);

function mySortFunction(r1, r2)
{
	var o = 0;
	if(r1.id < r2.id)
	{
		o = -1;
	}
	else if(r1.id > r2.id)
	{
		o = 1;
	}
	return o;
}
```

unrelate()

Create a new unrelated foundset that is a copy of the current foundset.
If the current foundset is not related, no copy will made.

Returns

JSFoundSet

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
foundset.unrelate();
```