ViewFoundSet

Apr 07, 2024 05:33

Supported Clients

SmartClient WebClient NGClient

Constants Summary

Number	MONITOR_AGGREGATES	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for changes in columns (selected) of the given datasource in the query that can affect aggregates.
Number	MONITOR_COLUMNS	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen also for column changes of the given table/datasource.
Number	MONITOR_DELETES	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for deletes on the given table/datasource.
Number	MONITOR_DELETES_FOR_PRIMARY_TABLE	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for deletes on the given table/datasource which should be the primary/main table of this query.
Number	MONITOR_INSERT	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for inserts on the given table/datasource.
Number	MONITOR_JOIN_CONDITIONS	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen also for column changes of the given table/datasource in the join statement - like order_lines.
Number	MONITOR_WHERE_CONDITIONS	Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen also for column changes of the given table/datasource that are used in the where statement - like order_lines.
String	VIEW_FOUNDSET	

Property Summary

Boolean	multiSelect	Returns true if this foundset is in multiselect mode and false if it's in single-select mode.
void	setMultiSelect	Puts this foundset in multi-select or single-select mode.

Methods Summary

Boolean	dispose()	Dispose and unregisters a view foundset from memory when is no longer needed.
void	enableDatabroadcastFor(queryTable)	Databroadcast can be enabled per select table of a query, the select table can be the main QBSelect or on of it QBJoins By default this monitors only the column values that are in the result of the QBSelect, you can only enable this default monitoring for a table if for that table also the PK is selected in the results.
void	enableDatabroadcastFor(queryTableclause, flags)	Enable the databroadcast for a specific table of the QBSelect or QBJoin with flags for looking for join or where criteria or deletes/inserts.
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 last sort columns that were set using viewfoundset sort api.
String	getDataSource()	Returns the datasource (view:name) for this ViewFoundSet.
Array	getEditedRecords()	Get the edited records of this view foundset.
Array	getFailedRecords()	Get the records which could not be saved.
QBSelect	getQuery()	Get the cloned query that created this ViewFoundSset (modifying this QBSelect will not change the foundset).
JSRecord	getRecord(index)	Get the ViewRecord object at the given index.
Number	getSelectedIndex()	Get the current record index of the viewfoundset.
Array	getSelectedIndexes()	Get the indexes of the selected records.
JSRecord	getSelectedRecord()	
Array	getSelectedRecords()	Get the selected records.
Number	getSize()	Get the number of records in this viewfoundset.
Boolean	hasRecordChanges()	Check whether the foundset has record changes.
Boolean	hasRecords()	Returns true if the viewfoundset has records.
void	loadAllRecords()	This will reload the current set of ViewRecords in this foundset, resetting the chunk size back to the start (default 200).
void	revertEditedRecords()	Revert changes of all unsaved view records of the view foundset.
void	revertEditedRecords(rec)	Revert changes of the provided view records.
Boolean	save()	Saves all records in the view foundset that have changes.
Boolean	save(record)	Saved a specific record of this foundset.
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.
JSRecordMarkers	validate(record)	Validates the given record, it runs first the method that is attached to the entity event "onValidate".
JSRecordMarkers	validate(record, customObject)	Validates the given record, it runs first the method that is attached to the entity event "onValidate".

Constants Details

MONITOR_AGGREGATES

```
Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for changes in
columns (selected) of the given datasource in the query that can affect aggregates. This means that when
there are deletes, inserts or updates on columns selected from that datasource, a full re-query will
happen to refresh the aggregates.

IMPORTANT: in general, this flag should be set on (possible multiple) datasources from the query that
have group by on their columns, and the columns don't contain the pk, or that have the actual aggregates
on their columns (because all those could influence the value of aggregates).

For example (ignoring the fact that in a real-life situation these fields might not change), a view
foundset based on this query:

SELECT orders.customerid, orders.orderdate, SUM(order_details.unitprice) FROM orders
   LEFT OUTER JOIN order_details ON orders.orderid = order_details.orderid
   GROUP BY orders.customerid, orders.orderdate
          ORDER BY orders.customerid asc, orders.orderdate desc

will want to enable databroadcast flag MONITOR_AGGREGATES on both "orders" (because if "orderdate" or
"customerid" - that are used in GROUP BY - change/are corrected on a row, that row could move from one
group to the other, affecting the SUM(order_details.unitprice) for the groups involved) and "order_details"
(because if "unitprice" changes/is corrected, the aggregate will be affected).

But if the above query would also select the orders.odersid (and also group by that) then the orders row
that you select for that sum will always be unique and only #MONITOR_COLUMNS has to be used for
those - if needed.
```

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

MONITOR_COLUMNS

```
Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen also for column
changes of the given table/datasource. This is used by default if you just use enableDatabroadcastFor()
without flags. If you use the one with the flags you need to give this one if you just want to listen to
column changes that are in the result for a given datasource and pk.

This constants needs to have the pk's selected for the given datasource (should be in the results).
```

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

MONITOR_DELETES

```
Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for deletes on the
given table/datasource. This will always result in a full query to detect changes whenever an delete on
that table happens.
```

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

MONITOR_DELETES_FOR_PRIMARY_TABLE

Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for deletes on the given table/datasource which should be the primary/main table of this query. If a delete comes in for this table, then we will only remove the records from the ViewFoundSet that do have this primary key in its value. So no need to do a full query. So this will only work if the query shows order_lines for the order_lines table, not for the products table that is joined to get the product_name. Only 1 of the 2 monitors for deletes should be registered for a table/datasource.

This constants needs to have the pk's selected for the given datasource (should be in the results)

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

MONITOR_INSERT

Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen for inserts on the given table/datasource. This will always result in a full query to detect changes whenever an insert on that table happens.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

MONITOR_JOIN_CONDITIONS

Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen also for column changes of the given table/datasource in the join statement - like order_lines.productid that has a join to orders and is displaying the productname. If a change in such a join condition (like order_lines.productid in the sample above) is seen then the query will be fired again to detect changes.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

MONITOR_WHERE_CONDITIONS

Constant for the flags in #enableDatabroadcastFor(QBTableClause,int) to listen also for column changes of the given table/datasource that are used in the where statement - like order_lines.unit_price > 100. If a change is seen on that datasource on such a column used in the where a full query will be fired again to detect changes.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

VIEW_FOUNDSET

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

Property Details

multiSelect

```
Returns true if this foundset is in multiselect mode and false if it's in single-select mode.
```

Returns

Boolean true if this foundset is in multiselect mode and false if it's in single-select mode.

Supported Clients

SmartClient,WebClient,NGClient

Sample

setMultiSelect

```
Puts this foundset in multi-select or single-select mode. If this foundset is shown in a form, this call can be
ignored as the form decides the foundset's multiselect.
```

Supported Clients

SmartClient,WebClient,NGClient

Sample

Methods Details

dispose()

```
Dispose and unregisters a view foundset from memory when is no longer needed.
Returns whether foundset was disposed.
If linked to visible form or component, view foundset cannot be disposed.

Normally ViewFoundSets are not hold on to by the system, so if you only use this inside a method it will be
disposed by itself.
This method is then just helps by also calling clear()

For ViewFoundSets that are also registered  by using true as the last argument in the call: databaseMananager.
getViewFoundSet(name, query, boolean register)
are hold on to by the system and Forms can use it for there foundset. Calling dispose on those will remove it
from the system, so it is not usable anymore in forms.
```

Returns

Boolean boolean foundset was disposed

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
vfs.dispose();
```

enableDatabroadcastFor(queryTable)

```
Databroadcast can be enabled per select table of a query, the select table can be the main QBSelect or on of it
QBJoins
By default this monitors only the column values that are in the result of the QBSelect, you can only enable
this default monitoring for a table if for that table also the PK is selected in the results.

you can use #enableDatabroadcastFor(QBTableClause,int) to specify what should be monitored more besides pure
column values per pk.
Those have impact on performance because for the most part if we see a hit then a full query is done to see if
there are changes.
```

Parameters

QBTableClause queryTable The QBSelect or QBJoin of a full query where this foundset should listen for data changes.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var select = datasources.db.example_data.order_details.createSelect();
 var join = select.joins.add("db:/example_data/products");
 join.on.add(select.columns.productid.eq(join.columns.productid));
 select.result.add(); // add columns of the select or join
 var vf = databaseManager.getViewFoundSet("myorders",select)
 vf.enableDatabroadcastFor(select);
 vf.enableDatabroadcastFor(join);
```

enableDatabroadcastFor(queryTableclause, flags)

```
Enable the databroadcast for a specific table of the QBSelect or QBJoin with  flags for looking for join or
where criteria or deletes/inserts.
These  flags can be a performance hit because the query needs to be executed again to see if there are any
changes.
For certain flags #MONITOR_COLUMNS and #MONITOR_DELETES_FOR_PRIMARY_TABLE the pk for that table must be in the
results.
```

Parameters

QBTableClause	queryTableclause	The QBSelect or QBJoin of a full query where this foundset should listen for data changes.
Number	flags	One or more of the ViewFoundSet.XXX flags added to each other.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var select = datasources.db.example_data.order_details.createSelect();
 var join = select.joins.add("db:/example_data/products");
 join.on.add(select.columns.productid.eq(join.columns.productid));
 select.result.add(); // add columns of the select or join
 var vf = databaseManager.getViewFoundSet("myorders",select)
 // monitor for the main table the join conditions (orders->product, when product id changes in the orders
table) and requery the table on insert events, delete directly the record if a pk delete happens.
 vf.enableDatabroadcastFor(select, ViewFoundSet.MONITOR_JOIN_CONDITIONS | ViewFoundSet.MONITOR_INSERT |
ViewFoundSet.MONITOR_DELETES_FOR_PRIMARY_TABLE);
 vf.enableDatabroadcastFor(join);
```

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 Object the return value of the callback

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 Object the return value of the callback

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getCurrentSort()

```
Get the last sort columns that were set using viewfoundset sort api.s
```

Returns

String String sort columns

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)
```

getDataSource()

```
Returns the datasource (view:name) for this ViewFoundSet.
```

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
solutionModel.getForm("x").dataSource  = viewFoundSet.getDataSource();
```

getEditedRecords()

Get the edited records of this view foundset.

Returns

Array an array of edited records

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var editedRecords = foundset.getEditedRecords();
for (var i = 0; i < editedRecords.length; i++)
{
   application.output(editedRecords[i]);
}
```

getFailedRecords()

Get the records which could not be saved.

Returns

Array an array of failed records

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
```

getQuery()

Get the cloned query that created this ViewFoundSset (modifying this QBSelect will not change the foundset). The ViewFoundSets main query can't be altered after creation; you need to make a new ViewFoundSet for that (it can have the same datasource name).

Returns

QBSelect query.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var q = foundset.getQuery()
q.where.add(q.columns.x.eq(100))
var newVF = databaseManager.getViewFoundset("name", q);
```

getRecord(index)

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

Parameters

Number index record index (1 based).

Returns

JSRecord ViewRecord record.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getSelectedIndex()

    Get the current record index of the viewfoundset.

Returns

Number int current index (1-based)

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getSelectedIndexes()

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

Returns

Array Array current indexes (1-based)

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()

Returns

JSRecord

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
```

getSelectedRecords()

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

Returns

Array Array current records.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getSize()

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

Returns

Number int current size.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var nrRecords = vfs.getSize()

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

hasRecordChanges()

```
Check whether the foundset has record changes.
```

Returns

Boolean true if the foundset has any edited records, false otherwise

Supported Clients

SmartClient,WebClient,NGClient

Sample

hasRecords()

```
Returns true if the viewfoundset has records.
```

Returns

Boolean true if the viewfoundset has records.

Supported Clients

SmartClient,WebClient,NGClient

Sample

loadAllRecords()

```
This will reload the current set of ViewRecords in this foundset, resetting the chunk size back to the start
(default 200).
All edited records will be discarded! So this can be seen as a full clean up of this ViewFoundSet.
```

Supported Clients

SmartClient,WebClient,NGClient

Sample

revertEditedRecords()

```
Revert changes of all unsaved view records of the view foundset.
```

Supported Clients

SmartClient,WebClient,NGClient

Sample

revertEditedRecords(rec)

```
Revert changes of the provided view records.
```

Parameters

Array rec an array of view records

Supported Clients

SmartClient,WebClient,NGClient

Sample

save()

```
Saves all records in the view foundset that have changes.
You can only save columns from a table if the pks of that table are also selected by the view foundset's query.
```

Returns

Boolean true if the save was successfull, false if not and then the record will hav the exception set.

Supported Clients

SmartClient,WebClient,NGClient

Sample

save(record)

```
Saved a specific record of this foundset.
You can only save columns from a table if also the pk is selected of that table
```

Parameters

JSRecord record ;

Returns

Boolean true if the save was successfull, false if not and then the record will hav the exception set.

Supported Clients

SmartClient,WebClient,NGClient

Sample

setSelectedIndex(index)

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

Parameters

Number index index to set (1-based)

Supported Clients

SmartClient,WebClient,NGClient

Sample

setSelectedIndexes(indexes)

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

Parameters

Array indexes An array with indexes to set.

Supported Clients

SmartClient,WebClient,NGClient

Sample

sort(sortString)

```
Sorts the foundset based on the given sort string.
Column in sort string must already exist in ViewFoundset.
```

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.
Column in sort string must already exist in ViewFoundset.
```

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

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;
}
```

validate(record)

```
Validates the given record, it runs first the method that is attached to the entity event "onValidate".
Those methods do get a parameter JSRecordMarkers where the problems can be reported against.
All columns are then also null/empty checked and if they are and the Column is marked as "not null" an error
will be
added with the message key "servoy.record.error.null.not.allowed" for that column.

An extra state object can be given that will also be passed around if you want to have more state in the
validation objects
(like giving some ui state so the entity methods know where you come from)

It will return a JSRecordMarkers when the record had validation problems
```

Parameters

JSRecord record ;

Returns

JSRecordMarkers Returns a JSRecordMarkers if the record has validation problems

Supported Clients

SmartClient,WebClient,NGClient

Sample

validate(record, customObject)

```
Validates the given record, it runs first the method that is attached to the entity event "onValidate".
Those methods do get a parameter JSRecordMarkers where the problems can be reported against.
All columns are then also null/empty checked and if they are and the Column is marked as "not null" an error
will be
added with the message key "servoy.record.error.null.not.allowed" for that column.

An extra state object can be given that will also be passed around if you want to have more state in the
validation objects
(like giving some ui state so the entity methods know where you come from)

It will return a JSRecordMarkers when the record had validation problems
```

Parameters

JSRecord	record	The ViewRecord to validate
Object	customObject	An extra customObject to give to the validate method.

Returns

JSRecordMarkers

Supported Clients

SmartClient,WebClient,NGClient

Sample