JSFoundSetUpdater

Performing Batch Updates

Foundsets are typically updated on a record-by-record basis, either as the user operates on a foundset-bound GUI component, or through programmatic interactions. However, sometimes it is necessary to perform an update to an entire foundset. For performance reasons, it is not advised that this be done by programmatically iterating over the foundset's records. Rather, it is recommended that batch updates be performed using the JSFoundsetUpdater API.

The Foundset Updater API is ideal to use for the following situations:

Updating an Entire Foundset

This essentially has the effect of issuing a SQL UPDATE statement using the WHERE clause that constrains the foundset. This presents a significant performance advantage over updating records individually. In the example below, a related foundset is updated, meaning all orders belonging to the selected customer will be affected. **Please note:** This method will not trigger any associated Table Events or modification columns.

```
var fsUpdater = databaseManager.getFoundSetUpdater(customers_to_orders);
fsUpdater.setColumn('status',101);
fsUpdater.performUpdate();
```

Updating a Partial Foundset with Different Values for Each Record

The Foundset Updater API can also be used to update part of a foundset. Moreover, unlike the above example, this approach allows for different values for each record. In the example below, the first 4 records (starting from the selected index) are updated by specifying an array of values for each column that is affected.

```
//        update first four records
var fsUpdater = databaseManager.getFoundSetUpdater(foundset);
fsUpdater.setColumn('customer_type',[1,2,3,4]);
fsUpdater.setColumn('my_flag',new [1,0,1,0]);
fsUpdater.performUpdate();
```

⚠ When using this approach, it matters what the selected index of the foundset is. The update will start with this record.

Method Summary

Boolean	next() Go to next record in this updater, returns true if successful.
Boolean	performUpdate() Do the actual update in the database, returns true if successful.
void	resetIterator() Start over with this iterator 'next' function (at the foundset selected record).
Boolean	setColumn(name, value) Set the column value to update, returns true if successful.

Method Details

next

Boolean **next** ()

Go to next record in this updater, returns true if successful.

Returns

Boolean - true if proceeded to next record, false otherwise

Sample

```
controller.setSelectedIndex(1)
var count = 0
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
while(fsUpdater.next())
{
        fsUpdater.setColumn('my_flag',count++)
}
```

performUpdate

Boolean **performUpdate** ()

Do the actual update in the database, returns true if successful.
There are 3 types of possible use with the foundset updater
1) update entire foundset by a single sql statement; that is not possible when the table of the foundset has tracking enabled then it will loop over the whole foundset.
When a single sql statement is done, modification columns will not be updated, because it does the update directly in the database, without getting the records.
2) update part of foundset, for example the first 4 row (starts with selected row)
3) safely loop through foundset (starts with selected row)

after the perform update call there are no records in edit mode, that where not already in edit mode, because all of them are saved directly to the database,
or in mode 1 the records are not touched at all and the database is updated directly.

Returns

Boolean - true if succeeded, false if failed.

Sample

```
//1) update entire foundset
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
fsUpdater.setColumn('customer_type',1)
fsUpdater.setColumn('my_flag',0)
fsUpdater.performUpdate()

//2) update part of foundset, for example the first 4 row (starts with selected row)
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
fsUpdater.setColumn('customer_type',new Array(1,2,3,4))
fsUpdater.setColumn('my_flag',new Array(1,0,1,0))
fsUpdater.performUpdate()

//3) safely loop through foundset (starts with selected row)
controller.setSelectedIndex(1)
var count = 0
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
while(fsUpdater.next())
{
        fsUpdater.setColumn('my_flag',count++)
}
```

resetIterator

void **resetIterator** ()

Start over with this iterator 'next' function (at the foundset selected record).

Returns

void

Sample

```
controller.setSelectedIndex(1)
var count = 0
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
while(fsUpdater.next())
{
	fsUpdater.setColumn('my_flag',++count)
}
fsUpdater.resetIterator()
while(fsUpdater.next())
{
	fsUpdater.setColumn('max_flag',count)
}
```

setColumn

Boolean **setColumn** (name, value)

Set the column value to update, returns true if successful.

Parameters

{String} name - The name of the column to update.
{Object} value - The new value (can be an array with data for x number of rows) to be stored in the specified column.

Returns

Boolean - true if succeeded, false if failed.

Sample

```
//1) update entire foundset
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
fsUpdater.setColumn('customer_type',1)
fsUpdater.setColumn('my_flag',0)
fsUpdater.performUpdate()

//2) update part of foundset, for example the first 4 row (starts with selected row)
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
fsUpdater.setColumn('customer_type',new Array(1,2,3,4))
fsUpdater.setColumn('my_flag',new Array(1,0,1,0))
fsUpdater.performUpdate()

//3) safely loop through foundset (starts with selected row)
controller.setSelectedIndex(1)
var count = 0
var fsUpdater = databaseManager.getFoundSetUpdater(foundset)
while(fsUpdater.next())
{
	fsUpdater.setColumn('my_flag',count++)
}
```