

JSDataSet

Property Summery

Number [#rowIndex](#)
Get or set the record index of the dataset.

Method Summery

| | |
|------------------------|--|
| Boolean | #addColumn (name, [index], [type]) adds a column with the specified name to the dataset. |
| void | #addHTMLProperty (row, col, name, value) Add an HTML property to an HTML tag produced in getAsHTML(). |
| void | #addRow (row) |
| void | #addRow ([index], array) Add a row to the dataset. |
| String | #createDataSource (name) |
| String | #createDataSource (name, [types]) Create a data source from the data set with specified name and using specified types. |
| String | #getAsHTML ([escape_values], [escape_spaces], [multi_line_markup], [pretty_indent], [add_column_names]) Get the dataset as an html table. |
| String | #getAsText (column_separator, row_separator, value_delimiter, add_column_names) Get the dataset as formatted text. |
| Object[] | #getColumnAsArray (index) Get the column data of a dataset as an Array. |
| String | #getColumnName (index) Get a column name based on index. |
| ServoyException | #getException () Get the database exception if an error occurred. |
| Number | #getMaxColumnIndex () Get the number of columns in the dataset. |
| Number | #getMaxRowIndex () Get the number of rows in the dataset. |
| Object[] | #getRowAsArray (index) Get the row data of a dataset as an Array. |
| Object | #getValue (row, col) Get the value specified by row and column position from the dataset. |
| Boolean | #hadMoreData () Return true if there is more data in the resultset then specified by maxReturnedRows at query time. |
| Boolean | #removeColumn (index) Remove a column by index from the dataset. |
| void | #removeRow (row) Remove a row from the dataset. |
| void | #setValue (row, col, obj) Set the value specified by row and column position from the dataset. |
| void | #sort (col, sort_direction) Sort the dataset on the given column in ascending or descending. |

Property Details

[rowIndex](#)
Get or set the record index of the dataset.

Returns

Number

Sample

```
//assuming the variable dataset contains a dataset
//to set the rowIndex:
dataset.rowIndex = 1 //sets the rowIndex to the first row (dataset is 1-based)
//to retrieve the rowIndex of the currently selected row
var currRow = dataset.rowIndex
```

Method Details

addColumn

Boolean **addColumn**(name, [index], [type])

adds a column with the specified name to the dataset.

Parameters

name – column name.

[index] – column index number between 1 and getMaxColumnIndex().

[type] – the type of column, see JSColumn constants.

Returns

Boolean – true if succeeded, else false.

Sample

```
//assuming the variable dataset contains a dataset
var success = dataset.addColumn('columnName',1);
```

addHTMLProperty

void **addHTMLProperty**(row, col, name, value)

Add an HTML property to an HTML tag produced in getAsHTML().

For row and col parameters use:

1 = applies to the container

0 = applies to all

>0 = applies to specific cell

Parameters

{**Number**} row – row number

{**Number**} col – column number

{**String**} name – String property name

{**String**} value – String property value

Returns

void

Sample

```
//adds a container property (to TABLE tag)
dataset.addHTMLProperty(-1,-1,'cellspacing','3');

//adds a row property to all rows (to TR tag)
dataset.addHTMLProperty(0,0,'class','text');

//adds a row property to second row (to TR tag)
dataset.addHTMLProperty(2,0,'class','text');

//adds a column property to all 3rd columns (to TD tag)
dataset.addHTMLProperty(0,3,'class','redcolumn') ;

//adds a specific cell property (to TD tag)
dataset.addHTMLProperty(2,4,'color','blue');

globals.html_field = '<html>'+dataset.getAsHTML()+'</html>';
```

addRow

void **addRow**([index], array)

Add a row to the dataset.

Parameters

{**Number**} [index] – index to add row (1-based), add by default as last row

{**Object[]**} array – row data

Returns

void

Sample

```
//assuming the variable dataset contains a dataset
dataset.addRow(new Array(1,2,3,4,5,6,7,7)); //adds a row with 8 columns
dataset.addRow(2, new Array(1,2,3,4,5,6,7,7)); //adds a row with 8 columns after row 2
```

createDataSource

String **createDataSource**(name, [types])

Create a data source from the data set with specified name and using specified types.

Parameters

[{String}](#) name – data source name

[{Object}](#) [types] – array of types as defined in JSColumn, when null or not supplied types are inferred from the data if possible.

Returns

[String](#) – String uri reference to the created data source.

Sample

```
ds.addColumn('my_id');
    ds.addColumn('my_label');
    var uri = ds.createDataSource('mydata', [JSColumn.INTEGER, JSColumn.TEXT]);
    var jsform = solutionModel.newForm(fname, uri, null, true, 300, 300);

var query = 'select customerid, address, city, country from customers';
var ds2 = databaseManager.getDataSetByQuery('example_data', query, null, 999);
var uri2 = ds2.createDataSource('mydata2'); // types are inferred from query result
```

getAsHTML

[String](#) **getAsHTML**([escape_values], [escape_spaces], [multi_line_markup], [pretty_indent], [add_column_names])

Get the dataset as an html table.

Parameters

[escape_values] – if true, replaces illegal HTML characters with corresponding valid escape sequences.

[escape_spaces] – if true, replaces text spaces with non-breaking space tags () and tabs by four non-breaking space tags.

[multi_line_markup] – if true, multiLineMarkup will enforce new lines that are in the text; single new lines will be replaced by
, multiple new lines will be replaced by <p>

[pretty_indent] – if true, adds indentation for more readable HTML code.

[add_column_names] – if false, column headers will not be added to the table.

Returns

[String](#) – String html.

Sample

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()),query,args,maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
globals.html_field = '<html>'+dataset.getAsHTML()+ '</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

getAsText

[String](#) **getAsText**(column_separator, row_separator, value_delimiter, add_column_names)

Get the dataset as formatted text.

Parameters

[{String}](#) column_separator – any specified column separator; examples: tab '\t'; comma ','; semicolon ';'; space ' '.

[{String}](#) row_separator – the specified row separator; examples: new line '\n'.

[{String}](#) value_delimiter – the specified value delimiter; example: double quote '"

[{Boolean}](#) add_column_names – boolean if true column names will be added as a first row.

Returns

[String](#) – String formatted text.

Sample

```
//assuming the variable dataset contains a dataset
//you can create csv or tab delimited results
var csv = dataset.getAsText(',', '\n', '', true)
var tab = dataset.getAsText('\t', '\n', '', true)
```

getColumnAsArray

Object[] **getColumnAsArray**(index)

Get the column data of a dataset as an Array.

Parameters

{**Number**} index – index of column (1-based).

Returns

Object[] – Object array of data.

Sample

```
//assuming the variable dataset contains a dataset
var dataArray = dataset.getColumnAsArray(1); //puts the contents from the first column of the dataset into an array
//once you have it as an array you can loop through it or feed it to a custom valuelist for example
```

getColumnName

String **getColumnName**(index)

Get a column name based on index.

Parameters

{**Number**} index – index of column (1-based).

Returns

String – String column name.

Sample

```
//assuming the variable dataset contains a dataset
var firstColumnName = dataset.getColumnName(1) //retrieves the first columnname into the variable firstColumnName
//using a loop you can get all columnnames in an array:
var query = 'select * from customers';
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()), query, null, 100);
var colArray = new Array()
for (var i = 1; i <= dataset.getMaxColumnIndex(); i++)
{
    colArray[i-1] = dataset.getColumnName(i)
    //note the -1, because an array is zero based and dataset is 1 based.
}
```

getException

ServoyException **getException**()

Get the database exception if an error occurred.

Returns

ServoyException – ServoyException exception or null when not available.

Sample

```
//assuming the variable dataset contains a dataset
var dbException = dataset.getException();
```

getMaxColumnIndex

Number **getMaxColumnIndex**()

Get the number of columns in the dataset.

Returns

Number – int number of columns.

Sample

```
//assuming the variable dataset contains a dataset
for (var i = 1; i <= dataset.getMaxColumnIndex(); i++)
{
    colArray[i-1] = dataset.getColumnNames(i)
    //have to subtract 1, because an array is zero based and a dataset is 1 based.
}
```

getMaxRowIndex

Number **getMaxRowIndex()**

Get the number of rows in the dataset.

Returns

Number – int number of rows.

Sample

```
//assuming the variable dataset contains a dataset
var totalRows = dataset.getMaxRowIndex();
```

getRowAsArray

Object[] **getRowAsArray(index)**

Get the row data of a dataset as an Array.

Parameters

{**Number**} index – index of row (1-based).

Returns

Object[] – Object array of data.

Sample

```
//assuming the variable dataset contains a dataset
var dataArray = dataset.getRowAsArray(1); //puts the contents from the first row of the dataset into an array
//once you have it as an array you can loop through it
```

getValue

Object **getValue(row, col)**

Get the value specified by row and column position from the dataset.

Parameters

{**Number**} row – row number, 1-based

{**Number**} col – column number, 1-based

Returns

Object – Object value

Sample

```
//assuming the variable dataset contains a dataset
var dataAtRow2Col1 = dataset.getValue(2, 1);
```

hadMoreData

Boolean **hadMoreData()**

Return true if there is more data in the resultset then specified by maxReturnedRows at query time.

Returns

Boolean – boolean more data available

Sample

```
var ds = databaseManager.getDataSetByQuery('example_data', 'select order_id from orders', null, 10000)
if (ds.hadMoreData())
{
    // handle large result
}
```

removeColumn

Boolean **removeColumn(index)**

Remove a column by index from the dataset.

Parameters

{[Number](#)} index – index of column to remove (1-based)

Returns

[Boolean](#) – true if succeeded, else false.

Sample

```
//assuming the variable dataset contains a dataset
var success = dataset.removeColumn(1); // removes first column
```

removeRow

void **removeRow**(row)

Remove a row from the dataset.

Parameters

{[Number](#)} row – row index to remove, -1 for all rows

Returns

void

Sample

```
//assuming the variable dataset contains a dataset
dataset.removeRow(1); //removes the first row
dataset.removeRow(-1); //removes all rows
```

setValue

void **setValue**(row, col, obj)

Set the value specified by row and column position from the dataset.

Use row = -1, to set columnnames.

Parameters

{[Number](#)} row – row number, 1-based

{[Number](#)} col – column number, 1-based

{[Object](#)} obj – the value to be stored at the given row and column.

Returns

void

Sample

```
//assuming the variable dataset contains a dataset
dataset.getValue(2, 1, 'data');
```

sort

void **sort**(col, sort_direction)

Sort the dataset on the given column in ascending or descending.

Parameters

{[Number](#)} col – column number, 1-based

{[Boolean](#)} sort_direction – boolean true for ascending, false for descending

Returns

void

Sample

```
//assuming the variable dataset contains a dataset
dataset.sort(1, false)
```