The Basics

Simple Declaration

```
<ntb:grid id="grid1" width="630" height="200" mode="livescrolling"
          gethandler="data.jsp">
  <ntb:columns>
    <ntb:textcolumn label="Contact Name" xdatafld="Name" width="200">
    </ntb:textcolumn>
    <ntb:textcolumn label="Contact Email" xdatafld="Email"  width="200">
    </ntb:textcolumn>
  </ntb:columns>
</ntb:grid>
```

Initialization

```
nitobi.loadComponent("grid1");
```

Needed for component to work!

Namespace

```
<html xmlns:ntb="http://www.nitobi.com">
```

Declaration Details

Column Types

```
<ntb:textcolumn></ntb:textcolumn>
<ntb:numbercolumn></ntb:numbercolumn>
<ntb:datecolumn></ntb: datecolumn>
```

Column Editor Types

```
<ntb:dateeditor></ntb:dateeditor>
<ntb:checkboxeditor></ntb:checkboxeditor>
<ntb:imageeditor></ntb:imageeditor>
<ntb:linkeditor></ntb:linkeditor>
<ntb:listboxeditor></ntb:listboxeditor>
<ntb:lookupeditor></ntb:lookupeditor>
<ntb:numbereditor></ntb:numbereditor>
<ntb:passwordeditor></ntb:passwordeditor>
<ntb:textareaeditor></ntb:textareaeditor>
<ntb:texteditor></ntb:texteditor>
```

Modes Summary

Sample Declaration

```
<ntb:grid id="grid1" mode="livescrolling" >
```

livescrolling - Ajax driven, seamless scrolling through large datasets. Retrieves blocks of data from the getHandler as the user scrolls down.
locallivescrolling - Static data only. Livescrolling is performed on static data embedded in the html declaration.
nonpaging - All the data is rendered immediately. No paging or livescrolling. Databound - a gethandler is needed.
localstandard - Paging is used, but on local static data. No gethandler.
standard - Ajax paging with a gethandler.

Events

Events are subscribed to via the declaration. Eg:

```
<ntb:grid id="grid1" oncellclickevent="myFunction(eventArgs)" >
```

Just a few events:

See the section on eventArgs

<ntb:grid>

```
oncellclickevent / oncelldoubleclickevent
ondatareadyevent / onhtmlreadyevent
onbeforeloadpreviouspageevent / onbeforeloadnextpageevent
onbeforeresizeevent / onafterresizeevent
onhandlererrorevent
onbeforerefreshevent / onafterrefreshevent
oncellfocusevent / oncellblurevent
oncontextmenuevent
```

Can also subscribe to events using JavaScript.

<ntb:column>

```
onheaderclickevent / oncellclickevent
oncellvalidateevent
oncellfocusevent / oncellblurevent
```

Commonly Used Attributes

<ntb:column> <ntb:textcolumn> <ntb:datecolumn> <ntb:numbercolumn>

classname - Use a css class on a particular column
cssstyle - Embed css styling information right in the column
label - The title of the column
xdatafld - The data field to bind to the column
sortdirection - asc or desc
sortenabled - true or false
initial - The starting value when a new record is inserted
editable - true or false
align - left or right

<ntb:datecolumn>

mask - the date mask to use. example: “EEE, MMM d, "yy”

<ntb:numbercolumn>

mask - the number mask to use. example: “$###,###.00”

<ntb:imageeditor>

imageurl - if no datasource field is defined, this will be used as the source for the image

<ntb:linkeditor>

openwindow - true or false

<ntb:listboxeditor>
<ntb:lookupeditor>
<ntb:numbereditor>
<ntb:texteditor>
<ntb:dateeditor>
<ntb:checkboxeditor>

These are used in <ntb:*column> tags.

Important Methods

Getting the Grid Object:

```
nitobi.getComponent(gridID)
```

Important methods for nitobi.grid.Grid

```
deleteRow() / insertRow(<int> rowIndex)
edit(e) - edits the current cell
getCellObject(<int> row,<int> col)
getCellValue(row,col) / setCellValue(row,col)
getRowCount()
getRowObject() / getColumnObject()
getSelectedCellObject()
getSelectedRow(<bool> rel) / getSelectedRows()
move(<int> x, <int> y) - moves active cell by x,y
resize(<int> width, <int> height)
save()
selectAllRows()
selectCellByCoords(<int> row, <int> col)
```

Important methods for nitobi.grid.Column

```
setWidth(<int> width)
setSortEnabled(<bool> value)
setOnCellClickEvent(<string> value)
setEditable(<bool> value)
setClassName(<string> value)
setAlign(<string> value)
```

Important methods for nitobi.grid.Row

```
getCell(<int> index) / getRowNumber(el)
```

Important methods for nitobi.grid.Cell

```
edit() - puts the cell into edit mode
getColumn() - zero based index of the column to
which the cell belongs
getHtml() / getRow() / getValue()
getColumnNumber(el) / getRowNumber(el)
```

Using eventArgs

Many events are able to pass an object to the recipient function that contains transient event information created when the event is triggered. This is achieved by including an eventArgs argument in the declaration for your event. eg:

```
<ntb:textcolumn oncellvalidateevent=“myfunct
(eventArgs)”></ntb:textcolumn>
```

Then in your function:

```
function myfunct(eA) { myRow = eA.getRow(); }
```

The following properties are available in eventArgs:

```
<string> oldValue  /  <string> newValue
<function> getSource() - Returns grid object
<function> getCell() - Returns cell object
<function> getEvent() - Returns the event object
<function> getRow() - Returns the row object
```

Sample Use Cases

Programmatically Resize a Grid

```
var myGrid = nitobi.getGrid('myGridID');
myGrid.setWidth(x);
myGrid.setHeight(y);
myGrid.generateCss();
myGrid.alignSurfaces();
```

Add Querystring Parameters to GetHandler

Since grid adds its own parameters to the gethandler, its necessary to keep track of custom ones in a special array. Grid provides special methods to add and remove attributes.

To add attributes:

```
var myDataGrid= nitobi.getComponent('myGridID');
myDataGrid.getDataSource().setGetHandlerParameter('myvar', '1');
myDataGrid.getDataSource().setGetHandlerParameter('myvar2', '2');
myDataGrid.dataBind();
```

To remove an attribute

```
myDataGrid.getDataSource().setGetHandlerParameter('myvar', null);
```

Force a cell into Edit Mode

```
var myGrid = nitobi.getComponent('myGridID');
var myCell = myGrid.getCellObject(row, col);
myCell.edit();
```

Pass Messages from Handlers to Grid

It's is possible to pass messages from the server side handlers back to the Grid in the event of an error, or for any purpose.

On the Server-Side

First generate the error on the server using the API:

* Classic ASP: EBAGetHandler_SetErrorMessage(ErrorMessage)
  EBASaveHandler_SetErrorMessage(ErrorMessage)
* PHP: EBAGetHandler::setErrorMessage($message)
  EBASaveHandler::setErrorMessage($message)
* JSP: GetHandler.setErrorMessage(String message)
  SaveHandler.setErrorMessage(String message)

In JavaScript

The Grid can access any errors generated by the get and save handlers by using the onhandlererrorevent attribute defined for the <ntb:grid> element. If an error message has been set on the server side, the handlererror event will fire.

To access the error message that was set by the handler, you can use the nitobi.data.DataTable.getHandlerError() method. E.g.

```
var dataSource = nitobi.getComponent('gridID').getDataSource();
var errorMessage = dataSource.getHandlerError();
```

Conditional Formatting

Its possible to perform conditional formatting in Grid using tokens. In your data, insert a class name or styling attribute for each row of data. For argument's sake, let's create an imaginary dataset with the following columns: ID, Status, Price, Item, myCSSStyle. In your dataset, make each entry of myCSSStyle some CSS styling information like a color: "#FFFFFF", "#CCCCCC", "#F0F0F0", and so-on.

Next, in the grid declaration, create some columns. In each column, use the cssstyle field, use a token for the myCSSStyle field:

```
<ntb:textcolumn  xdatafld="ID" cssstyle="color:{$myCSSStyle}">
<ntb:textcolumn  xdatafld="Status" cssstyle="color:{$myCSSStyle}">
<ntb:textcolumn  xdatafld="Price" cssstyle="color:{$myCSSStyle}">
<ntb:textcolumn  xdatafld="Item" cssstyle="color:{$myCSSStyle}">
```

When the grid is rendered, each row will use the XML field for myCSSStyle, making each row a different color.

This technique can also be applied to the classname attribute, allowing you the use actual CSS classnames instead of explicit CSS styling information.

Using Multi-Select

Using multirowselectenabled its possible to let the user select non-contiguous rows by CTRL-clicking on the desired rows. This allows for more robust manipulation of records in the Grid. Two tag attributes are needed:

```
<ntb:grid id="MultiSelectGrid"
   rowhighlightenabled="true"
   rowselectenabled="true"
   multirowselectenabled="true">
```

To loop through the selected rows use the getSelectedRows() method. The following snippet will loop through the selected rows and set the value of a cell.

```
var msGrid = nitobi.getGrid('MultiSelectGrid');
var selectedRows = msGrid.getSelectedRows();

for( var i = 0; i < selectedRows.length; i++ ) {
 var xi = selectedRows[i].getAttribute("xi");
 var mycell = msGrid.getCellObject( xi, col);
 mycell.setValue( "This row is selected" );
}
```

GetHandler Parameters

```
get.asp?StartRecordIndex=12&pagesize=20&sortcolumn=CustomerName&
sortdirection=ASC&searchstring=abc
```

StartRecordIndex - The record number to start returning data at.
PageSize - The number of records to return.
SortColumn - The data should already be sorted by this xml field.
SortDirection - The direction the data should be sorted.
SearchString - The text to filter by (LOOKUP gethandlers only).

Working with Grid Data

Getting an arbitrary cell value

```
function getGridValue(GridID, Row,Col) {
  var myGridObj = nitobi.getComponent(GridID);
  var CellValue = myGridObj.getCellObject
(Row,Col).getValue();
  return CellValue;
}
```

Get a field of XML by row number

```
function getXMLField(GridID, Row, FieldName) {
  var myGridObj = nitobi.getComponent(GridID);
  var myGridDataTable = myGridObj.getDataSource
("data");
  var FieldValue = myGridDataTable.getRecord
(Row).getAttribute(myGridDataTable.fieldMap
[FieldName].substring(1));
  return FieldValue;
}
```

Troubleshooting

Before losing hope, make sure you check the following things.

Grid Won't Initialize

* Did you define the xmlns:ntb namespace?
* Is your stylesheet/javascript included?
* Did you remember to include nitobi.toolkit.js?
* Remember to call nitobi.loadComponent('myID')?

No Data in Grid

* getHandler is broken? Check using fiddler (http://www.fiddlertool.com) or Firebug (http://www.getfirebug.com)

Not Saving

* saveHandler is broken? Check using fiddler or Firebug.
* Key handler is not set up.

Troubleshooting Resources

Nitobi Forums - http://forums.nitobi.com

Nitobi Knowledgebase - http://www.nitobi.com/kb/

Email Support - support@nitobi.com

Fiddler Tool - http://www.fiddlertool.com

Firebug - http://www.getfirebug.com