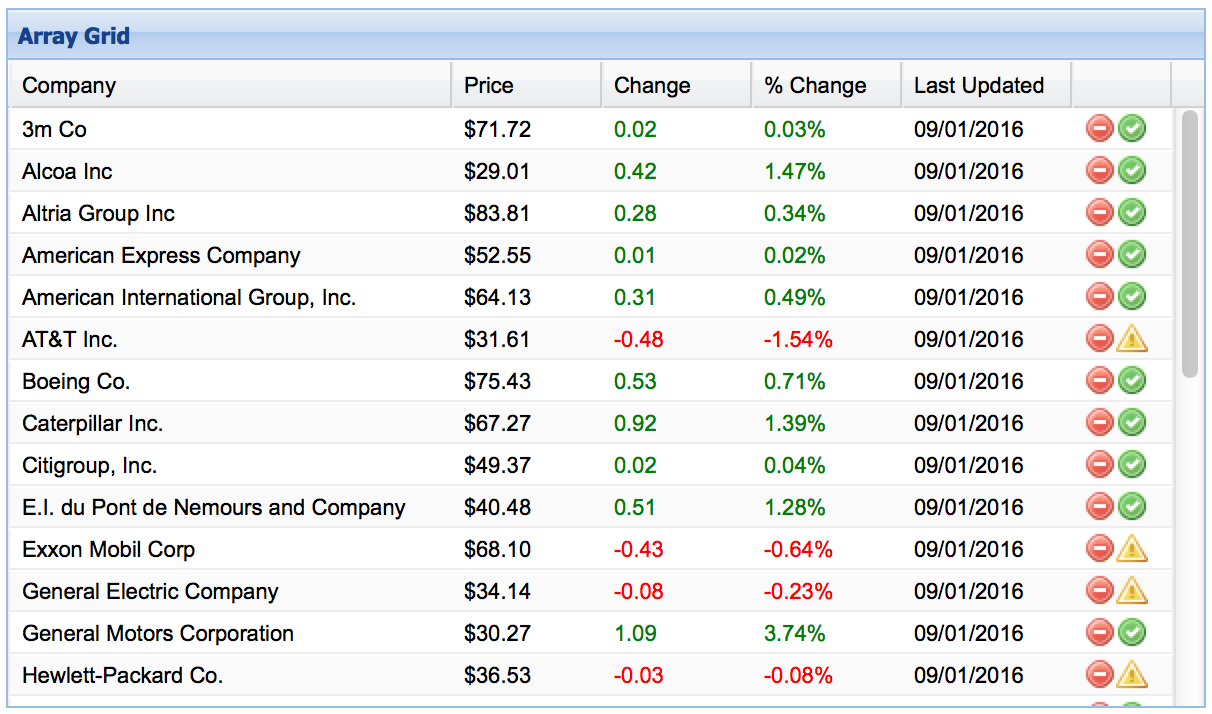
Olga Petrova

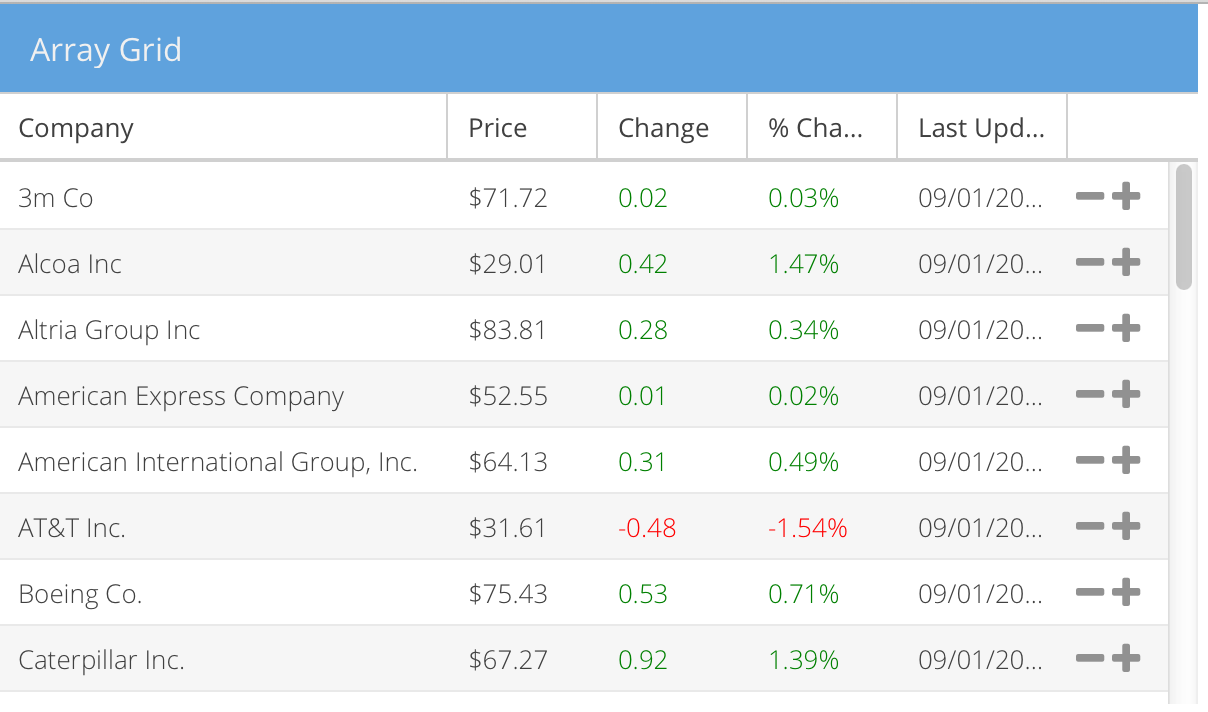
Ext JS 3.4 to 6 Migration Guide

This guide will cover a migration of Ext JS 3.4 grid example ({your\_extjs\_3.4\_folder}/examples/grid/array-grid.html) to Ext JS 6. You can find the original source code of Ext JS 3.4 example in Appendix.

The original example:



The final application:



Ext JS 6 application will be build based on MVVM pattern. So understanding of the pattern is prior requirement for this guide.

# Generate a new empty ext js 6 project

Go to Ext JS 6 SDK folder and run a command:

sencha generate app -classic -starter=false Migration ../migration

# Initialize new Project

## Go to migration folder and run commands:

### sencha generate view main.Main

## Create the file app/Application.js with this content:

Ext.define(**'Migration.Application'**, {  
 extend: **'Ext.app.Application'**,  
 name: **'Migration'**});

## Run the following commands from migration folder:

## sencha app build

## sencha web start

## Navigate to <http://localhost:1841/> with your browser to check that application is working. You should the following screen:

# Migration of Model

## Migration of the fields definition into separate Model class

In Ext JS 3.4 fields were defined inside the store definition:  
  
**var** store = **new** Ext.data.ArrayStore({  
 fields: [  
 {name: **'company'**},  
 {name: **'price'**, type: **'float'**},  
 {name: **'change'**, type: **'float'**},  
 {name: **'pctChange'**, type: **'float'**},  
 {name: **'lastChange'**, type: **'date'**, dateFormat: **'n/j h:ia'**}  
 ]  
});

Since Ext JS 4 there is a separate Model class where user can define fields and other logic related to your data. So we will move our fields definition into new Model class.

## Create Model

* 1. Create app/model/Company.js file with this content:  
       
     Ext.define(**'Migration.model.Company'**, {  
      extend: **'Ext.data.Model'**

});

* 1. Copy fields property from Ext JS 3.4 store to our model:  
       
     Ext.define(**'Migration.model.Company'**, {  
      extend: **'Ext.data.Model'**,  
      fields: [  
      {name: **'company'**},  
      {name: **'price'**, type: **'float'**},  
      {name: **'change'**, type: **'float'**},  
      {name: **'pctChange'**, type: **'float'**},  
      {name: **'lastChange'**, type: **'date'**, dateFormat: **'n/j h:ia'**}  
      ]  
     });

## Create Store

## Create app/store/Company.js file with this content: Ext.define(**'Migration.store.Company'**, { extend: **'Ext.data.ArrayStore'**, alias: **'store.company'** });

## Add reference to our Model class to Store definition: Ext.define(**'Migration.store.Company'**, { extend: **'Ext.data.ArrayStore'**, alias: **'store.company'**, requires: [**'Migration.model.Company'**], model: **'Migration.model.Company'** });

# Migration of View

## Run the following command from migration folder: sencha generate view company.Grid

## Edit app/view/company/CompanyGrid.js file and set base class to Ext.grid.Panel: extend: **"Ext.grid.Panel"**, add xtype: **'companygrid'** property and remove html property.

## Copy Ext JS 3.4 grid’s column definition and other properties into new CompanyGrid view and comment out all functions for now (we will migrate them later).

## Remove autoExpandColumn: 'company' config and add flex: 1 config to the Company column.

## Change icon and getClass properties of action column to iconCls with references to FontAwesome icons. When you will finish your code should look similar to this: Ext.define(**"Migration.view.company.Grid"**,{ extend: **"Ext.grid.Panel"**, requires: [ **"Migration.view.company.GridController"**, **"Migration.view.company.GridModel"** ], controller: **"company-grid"**, viewModel: { type: **"company-grid"** }, columns: [ { id :**'company'**, header : **'Company'**, width : 160, sortable : **true**, dataIndex: **'company'**, flex: 1 }, { header : **'Price'**, width : 75, sortable : **true**, *//renderer : 'usMoney',* dataIndex: **'price'** }, { header : **'Change'**, width : 75, sortable : **true**, *//renderer : change,* dataIndex: **'change'** }, { header : **'% Change'**, width : 75, sortable : **true**, *//renderer : pctChange,* dataIndex: **'pctChange'** }, { header : **'Last Updated'**, width : 85, sortable : **true**, *//renderer : Ext.util.Format.dateRenderer('m/d/Y'),* dataIndex: **'lastChange'** }, { xtype: **'actioncolumn'**, width: 50, items: [{ iconCls: 'x-fa fa-minus',tooltip: **'Sell stock'***/\*, handler: function(grid, rowIndex, colIndex) { var rec = store.getAt(rowIndex); alert("Sell " + rec.get('company')); }\*/* }, { iconCls: 'x-fa fa-plus’, */\* handler: function(grid, rowIndex, colIndex) { var rec = store.getAt(rowIndex); alert("Buy " + rec.get('company')); }\*/* }] } ], stripeRows: **true**, height: 350, width: 600, title: **'Array Grid'**, stateful: **true**, stateId: **'grid'** });

## Add our CompanyGrid to the Main view: Ext.define(**"Migration.view.main.Main"**,{ extend: **"Ext.panel.Panel"**, requires: [ **"Migration.view.main.MainController"**, **"Migration.view.main.MainModel"**, **"Migration.view.company.Grid"** ], controller: **"main-main"**, viewModel: { type: **"main-main"** }, items: [{ xtype: **'companygrid'** }] }); Do not forget to add CompanyGrid class into requires array of Main view.

## Run the following commands from migration folder:

## sencha app build

## sencha web start

## Navigate to <http://localhost:1841/> with your browser to check that application is working. You should the following screen:

# Creation of ViewModel

## Edit ViewModel file app/view/main/GridModel.js, add company store definition and copy data property from Ext JS 3.4 example: Ext.define(**'Migration.view.company.GridModel'**, { extend: **'Ext.app.ViewModel'**, alias: **'viewmodel.company-grid'**, stores: { company : { type: **'company'**, data: [71.72, 0.02, 0.03, **'9/1 12:00am'**], [**'Alcoa Inc'**, 29.01, 0.42, 1.47, **'9/1 12:00am'**],… } } });

# Migration of Code into viewController

## Move change function from Ext JS 3.4 example into GridController.js file and rename it to columnRenderer.

## Move ptcChange function from Ext JS 3.4 example into GridController.js file and rename it to ptcColumnRenderer.

## Move handler function of actioncolumn delete button to GridController.js and rename it to sellStock. Change reference to the store.

## Move handler function of actioncolumn add button to GridController.js and rename it to buyStock. Change Reference to the store.

When you will finish your GridController.js file should look similar to the following code:

Ext.define(**'Migration.view.company.GridController'**, {  
 extend: **'Ext.app.ViewController'**,  
 alias: **'controller.company-grid'**,  
  
 columnRenderer : **function**(val) {  
 **if** (val > 0) {  
 **return '<span style="color:green;">'** + val + **'</span>'**;  
 } **else if** (val < 0) {  
 **return '<span style="color:red;">'** + val + **'</span>'**;  
 }  
 **return** val;  
 },  
  
 pctColumnRenderer : **function** (val) {  
 **if** (val > 0) {  
 **return '<span style="color:green;">'** + val + **'%</span>'**;  
 } **else if** (val < 0) {  
 **return '<span style="color:red;">'** + val + **'%</span>'**;  
 }  
 **return** val;  
 },  
  
 sellStock: **function**(grid, rowIndex, colIndex) {  
 **var** rec = grid.getStore().getAt(rowIndex);  
 alert(**"Sell "** + rec.get(**'company'**));  
 },  
  
 buyStock: **function**(grid, rowIndex, colIndex) {  
 **var** rec = grid.getStore().getAt(rowIndex);  
 alert(**"Buy "** + rec.get(**'company'**));  
 }  
  
});

## Add references to this functions inside Grid view:

Ext.define(**"Migration.view.company.Grid"**,{  
 extend: **"Ext.grid.Panel"**,  
  
 requires: [  
 **"Migration.view.company.GridController"**,  
 **"Migration.view.company.GridModel"** ],  
  
 xtype: **'companygrid'**,  
  
 controller: **"company-grid"**,  
 viewModel: {  
 type: **"company-grid"** },  
  
 bind: {  
 store: **'{company}'** },  
  
 viewConfig: {  
  
 },  
 columns: [  
 {  
 id :**'company'**,  
 header : **'Company'**,  
 flex : 1,  
 sortable : **true**,  
 dataIndex: **'company'** },  
 {  
 header : **'Price'**,  
 width : 75,  
 sortable : **true**,  
 renderer : Ext.util.Format.usMoney,  
 dataIndex: **'price'** },  
 {  
 header : **'Change'**,  
 width : 75,  
 sortable : **true**,  
 renderer : **'columnRenderer'**,  
 dataIndex: **'change'** },  
 {  
 header : **'% Change'**,  
 width : 75,  
 sortable : **true**,  
 renderer : **'pctColumnRenderer'**,  
 dataIndex: **'pctChange'** },  
 {  
 header : **'Last Updated'**,  
 width : 85,  
 sortable : **true**,  
 renderer : Ext.util.Format.dateRenderer(**'m/d/Y'**),  
 dataIndex: **'lastChange'** },  
 {  
 xtype: **'actioncolumn'**,  
 width: 50,  
 items: [{  
 iconCls: **'x-fa fa-minus'**,  
 tooltip: **'Sell stock'**,  
 handler: **'sellStock'** }, {  
 iconCls: **'x-fa fa-plus'**,  
 handler: **'buyStock'** }  
 ]  
 }  
 ],  
  
 stripeRows: **true**,  
 height: 350,  
 width: 600,  
 title: **'Array Grid'**,  
 stateful: **true**,  
 stateId: **'grid'**});

## Navigate to <http://localhost:1841/> with your browser to check that application is working. You should the following screen:

# Conclusion

# Appendix

## The source code of Ext JS 3.4 example:

*/\*  
 This file is part of Ext JS 3.4  
  
 Copyright (c) 2011-2013 Sencha Inc  
  
 Contact: http://www.sencha.com/contact  
  
 Commercial Usage  
 Licensees holding valid commercial licenses may use this file in accordance with the Commercial  
 Software License Agreement provided with the Software or, alternatively, in accordance with the  
 terms contained in a written agreement between you and Sencha.  
  
 If you are unsure which license is appropriate for your use, please contact the sales department  
 at http://www.sencha.com/contact.  
  
 Build date: 2013-11-19 14:40:42  
 \*/*Ext.onReady(**function**(){  
 Ext.QuickTips.init();  
  
 *// NOTE: This is an example showing simple state management. During development,  
 // it is generally best to disable state management as dynamically-generated ids  
 // can change across page loads, leading to unpredictable results. The developer  
 // should ensure that stable state ids are set for stateful components in real apps.* Ext.state.Manager.setProvider(**new** Ext.state.CookieProvider());  
  
 *// sample static data for the store* **var** myData = [  
 [**'3m Co'**, 71.72, 0.02, 0.03, **'9/1 12:00am'**],  
 [**'Alcoa Inc'**, 29.01, 0.42, 1.47, **'9/1 12:00am'**],  
 [**'Altria Group Inc'**, 83.81, 0.28, 0.34, **'9/1 12:00am'**],  
 [**'American Express Company'**, 52.55, 0.01, 0.02, **'9/1 12:00am'**],  
 [**'American International Group, Inc.'**, 64.13, 0.31, 0.49, **'9/1 12:00am'**],  
 [**'AT&T Inc.'**, 31.61, -0.48, -1.54, **'9/1 12:00am'**],  
 [**'Boeing Co.'**, 75.43, 0.53, 0.71, **'9/1 12:00am'**],  
 [**'Caterpillar Inc.'**, 67.27, 0.92, 1.39, **'9/1 12:00am'**],  
 [**'Citigroup, Inc.'**, 49.37, 0.02, 0.04, **'9/1 12:00am'**],  
 [**'E.I. du Pont de Nemours and Company'**, 40.48, 0.51, 1.28, **'9/1 12:00am'**],  
 [**'Exxon Mobil Corp'**, 68.1, -0.43, -0.64, **'9/1 12:00am'**],  
 [**'General Electric Company'**, 34.14, -0.08, -0.23, **'9/1 12:00am'**],  
 [**'General Motors Corporation'**, 30.27, 1.09, 3.74, **'9/1 12:00am'**],  
 [**'Hewlett-Packard Co.'**, 36.53, -0.03, -0.08, **'9/1 12:00am'**],  
 [**'Honeywell Intl Inc'**, 38.77, 0.05, 0.13, **'9/1 12:00am'**],  
 [**'Intel Corporation'**, 19.88, 0.31, 1.58, **'9/1 12:00am'**],  
 [**'International Business Machines'**, 81.41, 0.44, 0.54, **'9/1 12:00am'**],  
 [**'Johnson & Johnson'**, 64.72, 0.06, 0.09, **'9/1 12:00am'**],  
 [**'JP Morgan & Chase & Co'**, 45.73, 0.07, 0.15, **'9/1 12:00am'**],  
 [**'McDonald\'s Corporation'**, 36.76, 0.86, 2.40, **'9/1 12:00am'**],  
 [**'Merck & Co., Inc.'**, 40.96, 0.41, 1.01, **'9/1 12:00am'**],  
 [**'Microsoft Corporation'**, 25.84, 0.14, 0.54, **'9/1 12:00am'**],  
 [**'Pfizer Inc'**, 27.96, 0.4, 1.45, **'9/1 12:00am'**],  
 [**'The Coca-Cola Company'**, 45.07, 0.26, 0.58, **'9/1 12:00am'**],  
 [**'The Home Depot, Inc.'**, 34.64, 0.35, 1.02, **'9/1 12:00am'**],  
 [**'The Procter & Gamble Company'**, 61.91, 0.01, 0.02, **'9/1 12:00am'**],  
 [**'United Technologies Corporation'**, 63.26, 0.55, 0.88, **'9/1 12:00am'**],  
 [**'Verizon Communications'**, 35.57, 0.39, 1.11, **'9/1 12:00am'**],  
 [**'Wal-Mart Stores, Inc.'**, 45.45, 0.73, 1.63, **'9/1 12:00am'**]  
 ];  
  
 */\*\*  
 \* Custom function used for column renderer  
 \** ***@param*** *{Object} val  
 \*/* **function** change(val) {  
 **if** (val > 0) {  
 **return '<span style="color:green;">'** + val + **'</span>'**;  
 } **else if** (val < 0) {  
 **return '<span style="color:red;">'** + val + **'</span>'**;  
 }  
 **return** val;  
 }  
  
 */\*\*  
 \* Custom function used for column renderer  
 \** ***@param*** *{Object} val  
 \*/* **function** pctChange(val) {  
 **if** (val > 0) {  
 **return '<span style="color:green;">'** + val + **'%</span>'**;  
 } **else if** (val < 0) {  
 **return '<span style="color:red;">'** + val + **'%</span>'**;  
 }  
 **return** val;  
 }  
  
 *// create the data store* **var** store = **new** Ext.data.ArrayStore({  
 fields: [  
 {name: **'company'**},  
 {name: **'price'**, type: **'float'**},  
 {name: **'change'**, type: **'float'**},  
 {name: **'pctChange'**, type: **'float'**},  
 {name: **'lastChange'**, type: **'date'**, dateFormat: **'n/j h:ia'**}  
 ]  
 });  
  
 *// manually load local data* store.loadData(myData);  
  
 *// create the Grid* **var** grid = **new** Ext.grid.GridPanel({  
 store: store,  
 columns: [  
 {  
 id :**'company'**,  
 header : **'Company'**,  
 width : 160,  
 sortable : **true**,  
 dataIndex: **'company'** },  
 {  
 header : **'Price'**,  
 width : 75,  
 sortable : **true**,  
 renderer : **'usMoney'**,  
 dataIndex: **'price'** },  
 {  
 header : **'Change'**,  
 **width** : 75,  
 **sortable** : **true**,  
 renderer : *change*,  
 dataIndex: **'change'** },  
 {  
 header : **'% Change'**,  
 width : 75,  
 sortable : **true**,  
 renderer : pctChange,  
 dataIndex: **'pctChange'** },  
 {  
 header : **'Last Updated'**,  
 width : 85,  
 sortable : **true**,  
 renderer : Ext.util.Format.dateRenderer(**'m/d/Y'**),  
 dataIndex: **'lastChange'** },  
 {  
 xtype: **'actioncolumn'**,  
 width: 50,  
 items: [{  
 icon : **'../shared/icons/fam/delete.gif'**, *// Use a URL in the icon config* tooltip: **'Sell stock'**,  
 handler: **function**(grid, rowIndex, colIndex) {  
 **var** rec = store.getAt(rowIndex);  
 alert(**"Sell "** + rec.get(**'company'**));  
 }  
 }, {  
 getClass: **function**(v, meta, rec) { *// Or return a class from a function* **if** (rec.get(**'change'**) < 0) {  
 **this**.items[1].tooltip = **'Do not buy!'**;  
 **return 'alert-col'**;  
 } **else** {  
 **this**.items[1].tooltip = **'Buy stock'**;  
 **return 'buy-col'**;  
 }  
 },  
 handler: **function**(grid, rowIndex, colIndex) {  
 **var** rec = store.getAt(rowIndex);  
 alert(**"Buy "** + rec.get(**'company'**));  
 }  
 }]  
 }  
 ],  
 stripeRows: **true**,  
 autoExpandColumn: **'company'**,  
 height: 350,  
 width: 600,  
 title: **'Array Grid'**,  
 *// config options for stateful behavior* stateful: **true**,  
 stateId: **'grid'** });  
  
 *// render the grid to the specified div in the page* grid.render(**'grid-example'**);  
});