SENCHA EXTJS GUIDE FOR BEGINNER

IRFAN MAULANA

PT SML TECHNOLOGIES www.smltech.co.id

1. Daftar Isi

2. Preview

Sencha ExtJS adalah Javascript *framework* yang menawarkan arsitektur MVC/MVVM dan menspesialisasikan dirinya pada pengambangan *Rich-Internet-Application* (RIA).

Sebagai salah satu *framework*, Sencha berusaha memenuhi semua kebutuhan developer dan user akan berbagai macam *plugin* dan fungsi yang kesemuanya telah di-*include* didalam packagenya. Jadi sebenarnya kita tidak perlu menambahkan lagi jQuery ataupun plugin-plugin lainnya sendiri karena Sencha telah mengakomodirnya.

Sebagai tools untuk membuat aplikasi berbasis RIA, maka Sencha ExtJS dilengkapi berbagai macam widget dan komponen yang sangat berguna dalam mempercepat proses development.

Begitu juga dengan arsitektur yang ditawarkan (MVC/MVVM) yang dimaksudkan untuk memudahkan proses development dan *debugging web application*, apalagi dengan konsep tersebut project bisa dikerjakan secara modular dan paralel oleh satu tim. Baca lebih lanjut disini.

3. Arsitektur

Sencha menawarkan MVC (Model-View-Controller) dan MVVM (Model-View-ViewModel) arsitektur, dimana MVVM baru bisa diaplikasikan di v.5 keatas sedangkan untuk v.4 menggunakan MVC.

Pada dasarnya kita bisa saja tidak menggunakan arsitektur yang ditawarkan dan memilih untuk menggunakan cara tersendiri dalam men-*develop* aplikasi, namun arsitektur ini akan terasa sangat membantu ketika

bekerja dalam team, dimana kemudahan untuk *debugging* sangat diperlukan. Baca lebih lanjut disini.

Berikut saya akan membahas beberapa bagian dari arsitektur ini :

2. 1. Model

Model adalah representasi dari struktur data yang akan ditampilkan, di sini akan dijelaskan field-field apa saja yang berada dalam model ini, tipe data, serta hubungan dengan model lainnya. Baca lebih lanjut disini.

```
Ext.define('UserModel', {
    extend: 'Ext.data.Model',
    fields: [
        {name: 'name',  type: 'string'},
        {name: 'age',   type: 'int'},
        {name: 'phone', type: 'string'},
        {name: 'alive', type: 'boolean'}
    ]
});
```

2. 2. Store

Store adalah data collection dari JSON response, artinya store otomatis terisi ketika mendapat balikan JSON yang sesuai dengan model yang berada di store tersebut. Store juga memiliki kemampuan filter, sort, bahkan query yang tentunya ini dilakukan di client-side dan dapat mengurangi jumlah komunikasi dengan back-side yang tidak diperlukan. Perlu diketahui bahwa store bisa langsung berkorelasi dengan komponen yang menggunakannya, sehingga ketika data dalam

store berubah maka akan mempengaruhi data yang ditampilkan dalam satu komponen.

Satu store hanya bisa menampung satu model di dalamnya, namun satu instance model bisa digunakan dalam beberapa store bila memang memiliki struktur yang sama tanpa mempengaruhi data di dalamnya. Baca lebih lanjut disini.

```
Ext.create('Ext.data.Store', {
    model: 'User',
    autoLoad: true,
    proxy: {
        type: 'ajax',
        url: '/users.json',
        reader: {
            type: 'json',
            root: 'users'
        }
    }
});
```

2.3 View

View merupakan tempat dimana kita membuat tampilan sebuah aplikasi. View bisa berupa container seperti panel, window, dll maupun komponen seperti grid, combobox, checkbox, dll. Di dalam view juga kita akan menggunakan layout yang telah kita pelajari di depan. Dalam membuat *mockup* sebuah aplikasi mungkin hanya akan dibutuhkan view saja, tanpa membutuhkan bagian lain dari arsitektur.

Ini mengapa memahami view harus menjadi modal yang penting bagi anda yang akan bergelut dengan profesi *Front-End Developer*.

```
Ext.create('Ext.grid.Panel', {
    title: 'Simpsons',
    store: Ext.data.StoreManager.lookup('simpsonsStore'),
    columns: [
        { text: 'Name',  dataIndex: 'name' },
        { text: 'Email', dataIndex: 'email', flex: 1 },
        { text: 'Phone', dataIndex: 'phone' }
    ],
    height: 200,
    width: 400,
    renderTo: Ext.getBody()
});
```

2.4 Controller

Controller adalah bagian yang akan mengatur interaksi dari semua bagian arsitektur. Di dalam controller juga biasanya di daftarkan berbagai macam bagian arsitektur seperti : Models, Stores, Views.

Tapi perlu diketahui bahwa semua yang di daftarkan di controller akan di load di awal aplikasi berjalan, itu kenapa untuk view biasanya saya memilih hanya akan me-load satu view utama dan memilih menggunakan requires untuk view yang saling berhubungan.

Pada ExtJS v.4 Controller bisa mendengarkan *event* yang di *fire* dari sebuah komponen dan menjalankan sebuah aksi di dalamnya, namun pada ExtJS v.5 fungsi ini dialihkan pada view-controller sehingga beban pada controller utama lebih ringan dan membuat aplikasi lebih

modular dengan mempunyai controller per masing-masing view. Baca lebih lanjut disini.

```
Ext.define('MyApp.controller.Users', {
    extend: 'Ext.app.Controller',
    init: function() {

    }
});
```

2.5 View-Controller

Merupakan controller dari satu view saja. Bisa mendengarkan berbagai event dan komponen yang berada dalam view yang berada dibawahnya. Baca lebih lanjut disini.

```
Ext.define('MyApp.view.foo.FooController', {

    extend: 'Ext.app.ViewController',
    alias: 'controller.foo',
    onBarChange: function (barTextField) {
        // called by 'change' event
    }
});
```

2.6 View-Model

Merupakan implementasi dari maraknya penggunaan *data-binding* dalam sebuah web aplikasi. View-Model memiliki kemampuan yang hampir sama dengan library *data-binding* pada umumnya, yakni

mengatur suatu data dan kemudian melemparkan nya langsung ke semua view yang menggunakan view model ini sehingga sebuah aplikasi akan terlihat lebih *real-time* dalam hal perubahan data, namun memang perlu diketahui bahwa semua perubahan data ini hanya dilakukan dalam sisi klien sehingga data untuk benar-benar melakukan perubahan data di sisi *server* tetap perlu dilakukan komunikasi *client-server*. Baca lebih lanjut disini.

```
Ext.define('MyApp.view.TestViewModel2', {

    extend: 'Ext.app.ViewModel',
    alias: 'viewmodel.test2',
    formulas: {
        x2: function (get) {
            return get('x') * 2;
        }
    }
});
```

2.7 Config Class

Ini sebenarnya bukan benar-benar bagian dari arsitektur MVC/MVVM dan baru diperkenalkan setelah v.5 keatas, namun berdasarkan *best-practice* hal ini memang perlu disampaikan dan seharusnya menjadi standard atau *coding style* dari para *developer*.

Seperti yang kita tahu bahwa penggunaan *global variable* di bahasa JavaScript sangat liar dan susah untuk diatur, untuk itulah ExtJS menggunakan config class ini untuk menangani penggunaan *global*

variable sehingga diharapkan para developer bisa mengurangi penggunaan *global variabel* dan beralih menggunakan ini.

Config class menawarkan fasilitas *getter-setter* untuk setiap *variable*-nya sehingga tidak dianjurkan untuk akses langsung ke dalam variabel tersebut. Baca lebih lanjut disini.

```
Ext.define('TEST.config.GlobalVariable',{
    singleton : true,
    config : {
        x : 1,
        y : 3,
    },
    constructor : function(config){
        this.initConfig(config);
    }
});
```

4. Bekerja dengan Layout

Salah satu kelebihan yang ditawarkan oleh Sencha ExtJS adalah layout manager-nya, sehingga memudahkan dan mempercepat kita dalam membuat suatu tampilan.

Bahkan menurut saya, bagi anda para beginner layouting adalah hal yang perlu anda ketahui sebelum memahami arsitekturnya sendiri. Bagi anda yang ditempatkan sebagai *Front-End developer* akan sangat sering berhadapan dengan hal ini dan pemahaman dasar mengenai jenis-jenis layout yang ada serta fungsi dari masing-masing layout akan sangat berpengaruh kedepannya.

Tidak jarang bahkan para *developer* yang sudah lebih dulu datang bertanya soal hal-hal dasar seperti ini, itu mengapa saya letakkan layouting ini

di halaman paling depan. Untuk layout yang tersedia dalam ExtJS v.4 bisa dilihat selengkapnya disini, dan untuk ExtJS v.5 bisa dilihat langsung disini.

Saya tidak akan menjelaskan semua layout yang ada, namun ada beberapa layout yang penting untuk diketahui karena akan sangat sering digunakan, antara lain :

3. 1. Fit Layout

Fit Layout adalah layout yang akan menyesuaikan berapapun width dan height dari container parent-nya, jadi ketika kita ingin suatu component yang akan stretch mengikuti besaran parent-nya, gunakanlah layout jenis ini. Baca lebih lanjut disini.

```
layout:'fit',
    items: {
        title: 'Fit Panel',
        html: 'Content',
        border: false
    }
```

3. 2. Border Layout

Border Layout adalah layout yang telah ditentukan region-region nya. Layout ini sangat mudah digunakan untuk membagi-bagi suatu container ke dalam region-region yang ditetapkan besaran height maupun width-nya. Salah satu syarat ketika menggunakan layout ini adalah bahwa minimal harus ada satu region yaitu center dibawahnya. Baca lebih lanjut disini.

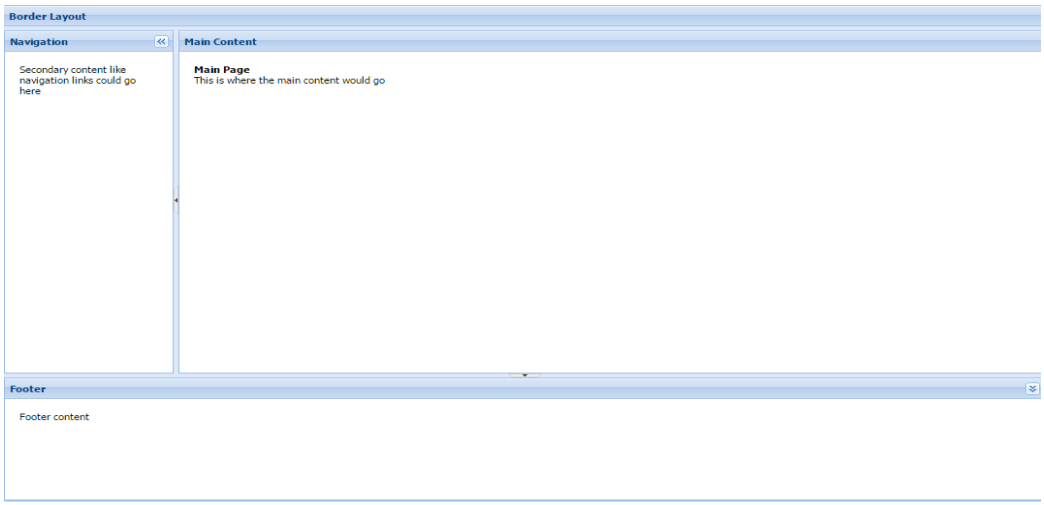
Perlu diketahui bahwa center adalah region yang akan memenuhi semua sisi yang tersisa dari border layout.

Beberapa region yang ada dalam border layout adalah :

1. North : Komponen bagian atas (harus diset height-nya).
2. South : Komponen bagian bawah (harus diset height-nya).
3. East : Komponen bagian kanan (harus diset width-nya).
4. West : Komponen bagian kiri (harus diset width-nya).
5. Center : Komponen di region ini akan mengisi semua ruang tersisa (tidak perlu diset height maupun width).

```
layout:'border',
defaults: {
    collapsible: true,
    split: true,
    bodyStyle: 'padding:15px'
},
items: [{
    title: 'Footer',
    region: 'south',
    height: 150,
    minSize: 75,
    maxSize: 250,
 },{
    title: 'Navigation',
    region:'west',
    margins: '5 0 0 0',
    width: 175,
    minSize: 100,
    maxSize: 250
},{
```

```
        title: 'Main Content',
        collapsible: false,
        region:'center',
        margins: '5 0 0 0'
    }]
```

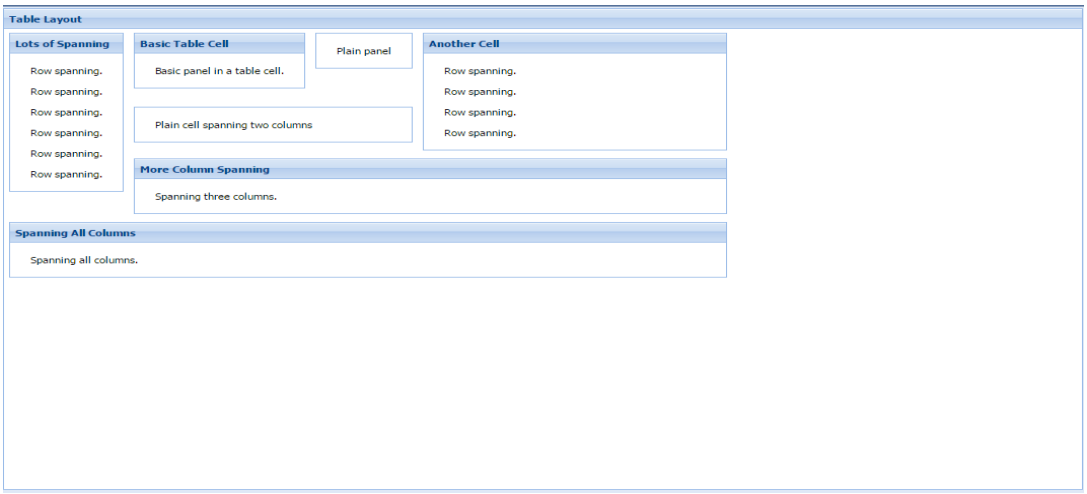


3. 3. Table Layout

Layout ini menghasilkan seperti *standard HTML table tag*, biasa digunakan ketika kita membutuhkan layout kompleks yang memerlukan *row spanning* ataupun *column spanning* di dalamnya. Ketika menggunakan layout ini anda perlu men-set jumlah maximal column pada layout tablenya. Anda juga bisa menambahkan *tdAttrs* dan *trAttrs* yang biasanya merupakan property maupun *style* dari kolom dan row di dalam layout ini. Baca lebih lanjut disini.

```
    layout: {
        type: 'table',
        columns: 4
    },
```

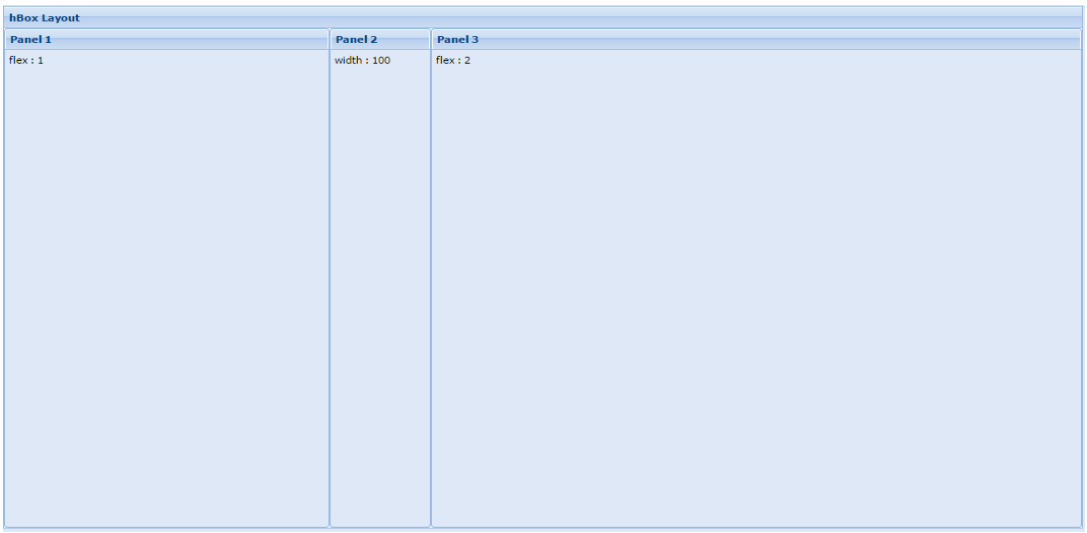
```
items: [
    {html:'1,1',rowspan:3},
    {html:'1,2'},
    {html:'1,3'},
    {html:'2,2',colspan:2},
    {html:'3,2'},
    {html:'3,3'}
]
```



3. 4. Hbox Layout

Layout ini akan mengatur *items* dibawahnya secara mendatar / horizontal (menyamping) dan secara berurutan. Layout ini akan sangat berguna ketika kita mesti membagi container ke dalam beberapa bagian tanpa mengetahui berapapun besaran *parent* nya, anda bisa gunakan *flex* untuk ini. Anda bisa menyisipkan *tbfill* untuk mengisi kekosongan yang ada untuk kemudian menambahkan komponen di ujung lainnya. Baca lebih lanjut disini.

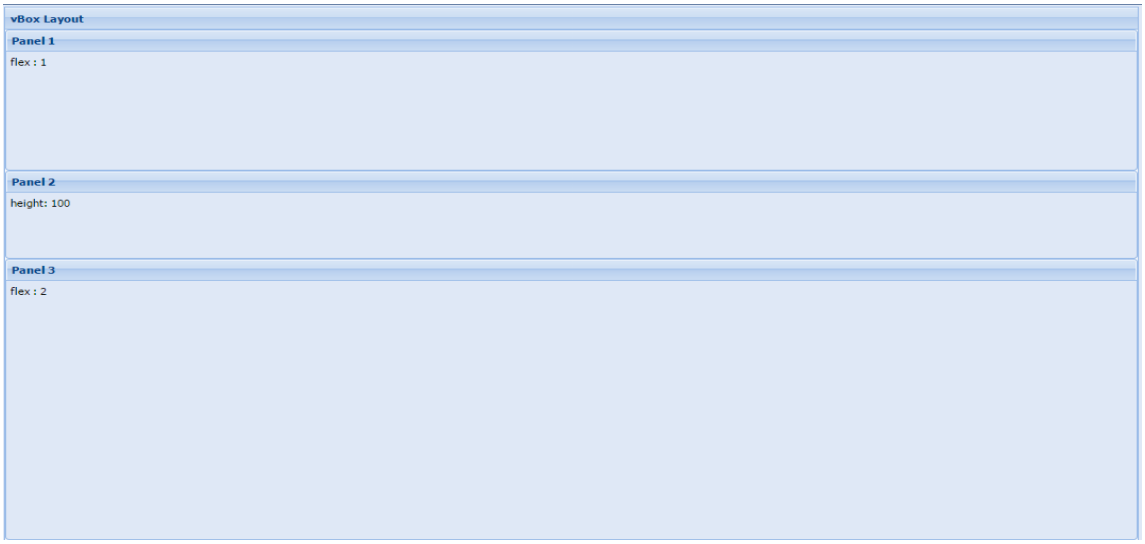
```
layout: {
    type: 'hbox',
    pack: 'start',
    align: 'stretch'
},
items: [
    {html:'panel 1', flex:1},
    {html:'panel 2', width:150},
    {html:'panel 3', flex:2}
]
```



3. 5. Vbox Layout

Layout ini akan mengatur *items* dibawahnya secara *vertical* (menurun) dan secara berurutan. Sama seperti hbox penggunaannya hanya saja dengan urutan kebawah. Baca lebih lanjut disini.

```
layout: {
    type: 'vbox',
    align : 'stretch',
    pack  : 'start',
},
items: [
    {html:'panel 1', flex:1},
    {html:'panel 2', height:150},
    {html:'panel 3', flex:2}
]
```

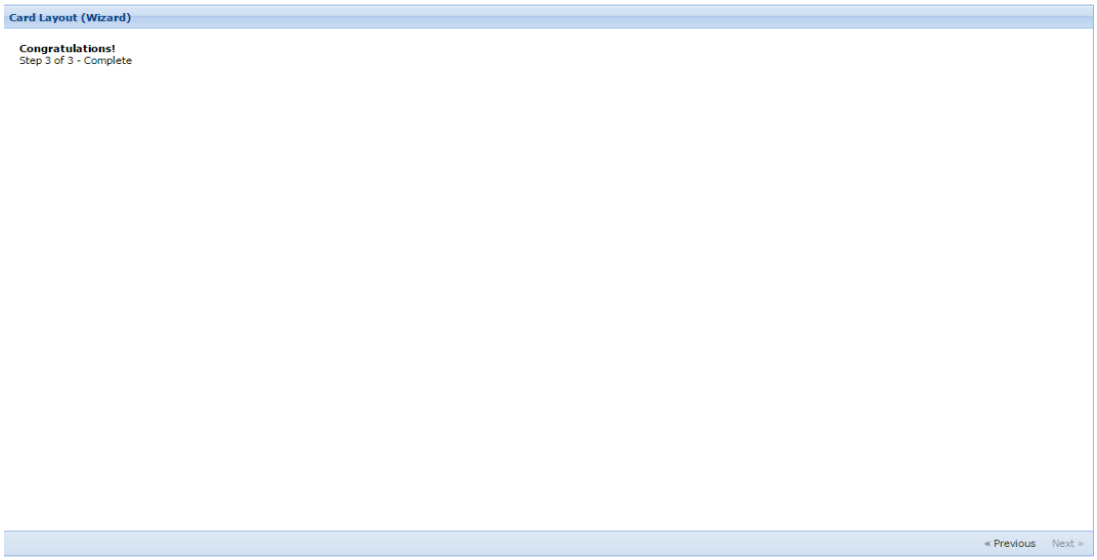


3. 6. Card Layout

Layout ini memiliki perspektif seperti mengatur kartu, yakni beberapa items dibawahnya ditata seperti kartu dan berurutan sesuai urutan penambahan nya, sehingga hanya akan ada satu item yang akan terlihat dalam satu waktu. Ini digunakan ketika kita ingin membuat layaknya *wizard pane* dengan tombol *next* dan *back* nya ataupun ketika hanya akan menampilkan satu items di satu waktu dan

items lainnya berada di bawah nya untuk kemudian di balik. Baca lebih lanjut disini.

```
layout:'card',
activeItem: 0, // index or id
bbar: ['->',
{
        id: 'card-prev',
        text: '&laquo; Previous'
},{
       id: 'card-next',
        text: 'Next &raquo;'
 }],
 items: [
{
       id: 'card-0',
       html: 'Step 1'
 },{
       id: 'card-1',
       html: 'Step 2'
},{
       id: 'card-2',
       html: 'Step 3'
}]
```

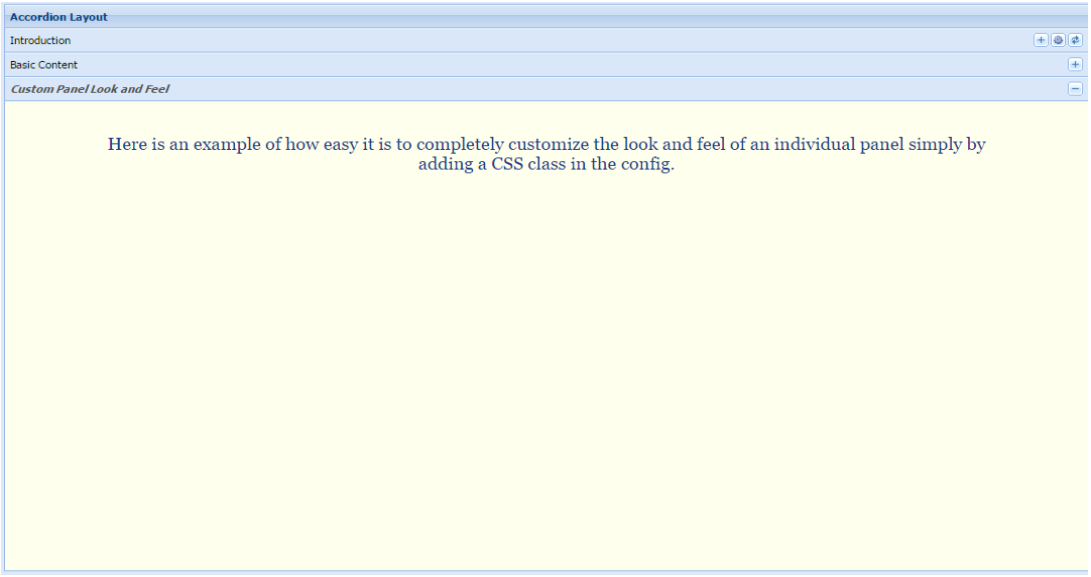


3. 7. Accordion Layout

Menampilkan satu tampilan dalam sekali waktu dan secara berurutan ke bawah dengan *accordion style*. Tidak dibutuhkan konfigurasi tambahan untuk layout ini, secara otomatis akan diatur oleh ExtJS dalam bentuk *accordion*. Baca lebih lanjut disini.

```
layout: 'accordion',
items:[
{
 title: 'Panel 1',
 html: 'Content'
},{
 title: 'Panel 2',
 id: 'panel2',
 html: 'Content'
}]
```

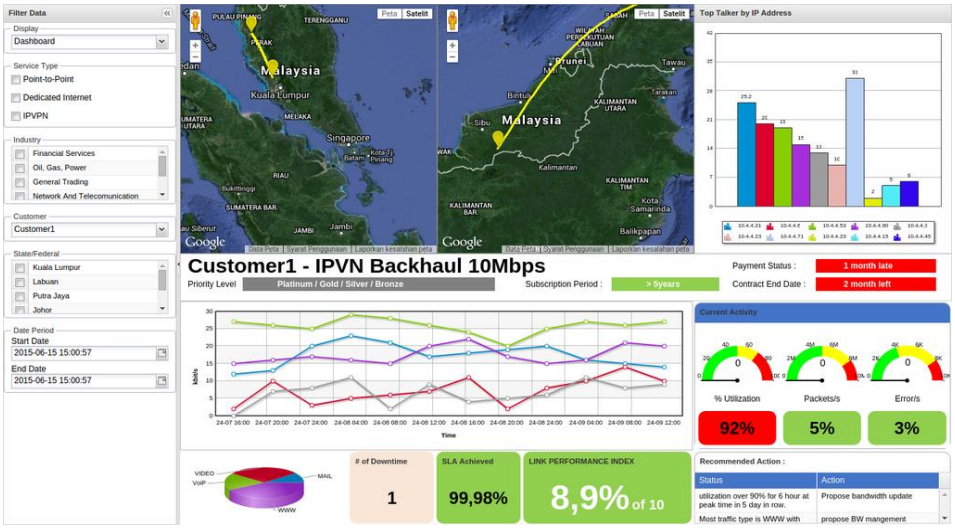
```
// css
#panel2 .x-panel-body {
  background:#ffe;
  color:#15428B;
}
#panel2 .x-panel-header-text {
  color:#555;
}
```



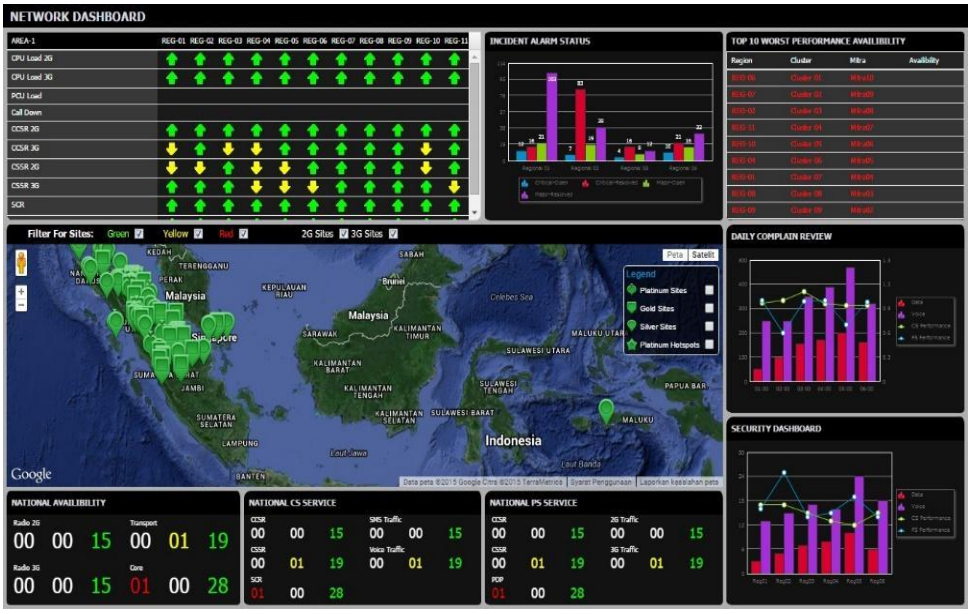
Latihan :

Buatlah mockup layouting seperti dibawah ini menggunakan berbagai layout yang sudah anda ketahui.

http://mazipan.github.io/ExtJS-TNMD/



https://mazipan.github.io/ExtJS-NetworkDashboard/



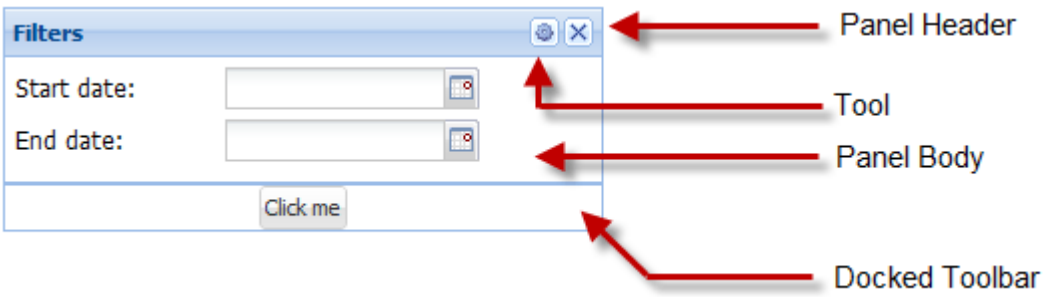
Semua kode sudah tersedia di github saya disini.

5. Rich Components

4.1. Panel dan Container

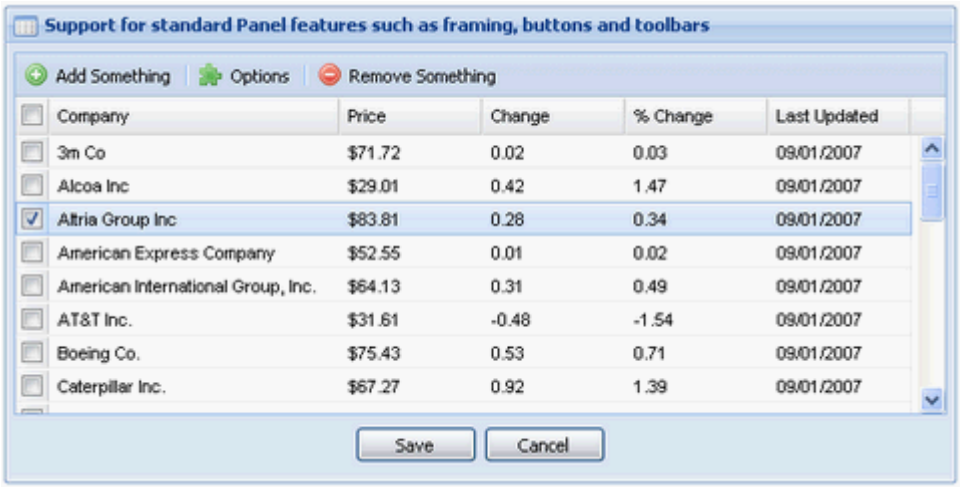
Panel dan Container adalah dua component yang hampir sama secara fungsi yakni sebagai *parent* atau dalam bahasa mudah adalah wadah bagi beberapa component lain diatasnya. Anda mungkin perlu sedikit membaca dokumentasi keduanya, container disini dan panel disini.

Bahwa keduanya memiliki perbedaan yang jelas dan sudah seharusnya kita bisa memilih mana yang sesuai dengan kebutuhan kita. Secara mudah dapat saya gambarkan bahwa perbedaan mendasar dari keduanya adalah container hanya sebuah *div* tanpa property lainnya, sedangkan panel adalah sebuah *div* yang memiliki berbagai property seperti *header, tools, toolbar* dan *body.*
https://mazipanneh.wordpress.com/2015/05/05/extjs-different-panel-vs-container/



4.2. Grid/Table

Grid/Table merupakan component yang sangat sering digunakan untuk presentasi sebuah data. Di ExtJS, grid merupakan component yang begitu *rich* dengan banyaknya *plugin* dan *feature*. Untuk *advance* grid, semakin tinggi jam terbang anda maka akan semakin kaya pengetahuan anda untuk memodifikasi atau membuat *custom* grid, saya tidak akan menjelaskan berbagai *custom* yang bisa dibuat dari sebuah grid.
Perlu diketahui bahwa minimal yang dibutuhkan oleh sebuah grid adalah sebuah store sebagai data yang akan ditampilkan dan *columns* sebagai array columns yang akan ditampilkan sebagai kolom-kolom dalam grid. Baca lebih lanjut disini.




```
Ext.create('Ext.data.Store', {
   storeId:'simpsonsStore',
   fields:['name', 'email', 'phone'],
   data:{'items':[
      {  'name': 'Lisa',
                  "email":"lisa@simpsons.com",
                  "phone":"555-111-1224"  },
      {  'name': 'Bart',
```

```
                    "email":"bart@simpsons.com",
                     "phone":"555-222-1234" },
        {   'name': 'Homer',
                    "email":"homer@simpsons.com",
                    "phone":"555-222-1244"  },
        {   'name': 'Marge',
                    "email":"marge@simpsons.com",
                    "phone":"555-222-1254"  }
    ]},
    proxy: {
        type: 'memory',
        reader: {
            type: 'json',
            rootProperty: 'items'
        }
    }
});

Ext.create('Ext.grid.Panel', {
    title: 'Simpsons',
    store: Ext.data.StoreManager.lookup('simpsonsStore'),
    columns: [
        { text: 'Name',  dataIndex: 'name' },
        { text: 'Email', dataIndex: 'email', flex: 1 },
        { text: 'Phone', dataIndex: 'phone' }
    ],
    height: 200,
    width: 400,
    renderTo: Ext.getBody()
});
```

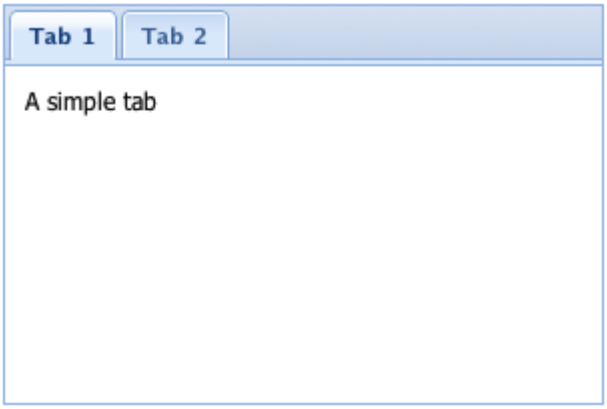
4.3. Combobox

Combobox juga adalah salah satu komponen yang sering digunakan dikarenakan fungsinya sebagai data selection, combobox juga memiliki store sebagai penampung data. Bisa di *custom* menjadi multiple select dengan checkbox, bisa juga digunakan sebagai autocompletion textbox, namun mungkin akan sedikit *tricky* untuk membuatnya.

Combobox memiliki dua value yakni displayField dan valueField, displayField adalah data yang ditampilkan sedangkan valueField merupakan data yang bisa diambil maupun dimasukkan ke dalam combobox tersebut. Baca lebih lanjut disini.

4.4. TabPanel/Tab

TabPanel/Tab dibutuhkan ketika membuat aplikasi kompleks sehingga membutuhkan *tab-ing* berdasarkan modulenya. Pada dasarnya tabpanel memanfaatkan penggunaan layout card namun dengan lebih sedikit konfigurasi sehingga user langsung bisa memilih tab mana yang akan dipilih. Baca lebih lanjut disini.




```
Ext.create('Ext.tab.Panel', {
    width: 400,
    height: 400,
    renderTo: document.body,
    items: [{
        title: 'Foo'
    }, {
        title: 'Bar',
        tabConfig: {
            title: 'Custom Title',
            tooltip: 'A button tooltip'
        }
    }]
});
```

6. Aturan Coding Style

Ada beberapa hal yang diharapkan menjadi aturan baku dalam membuat suatu code, ini berkaitan dengan kemudahan dalam mengatur, membaca dan debug serta memperbaiki code, beberapa aturan tersebut antara lain :

4.1. Penamaan folder menggunakan huruf *lower case* tanpa adanya space atau tanda penghubung lain, contoh : script, view, viewcontroller, dll.

4.2. Penamaan file dalam ExtJS arsitektur menggunakan *camel-case* dengan huruf pertama dari tiap kata adalah *upper*, contoh : AlarmDashboard.js, NetworkPanel.js, dll.

4.3. Berbeda dengan external resource yang ditambahkan di folder script (*baik js maupun css), penamaan file menggunakan *lower case* tanpa space dengan tanda hubung minus (-). Hal ini mengacu pada standard penamaan library diluar sana, contoh : jquery-2.1.1.min.js, index.js, index.css, dll.

4.4. Khusus untuk penamaan file yang termasuk dalam arsitektur ExtJS membiasakan untuk menambahkan tipe dari arsitektur yang file tersebut bawa, contoh : UserModel.js, UserStore.js, UserView.js, dll. Hal ini bisa dikecualikan dengan folder view karena biasanya akan terdapat banyak view dalam satu *case-flow*.

4.5. Penamaan file diharuskan untuk sesuai dengan fungsi dan peranan

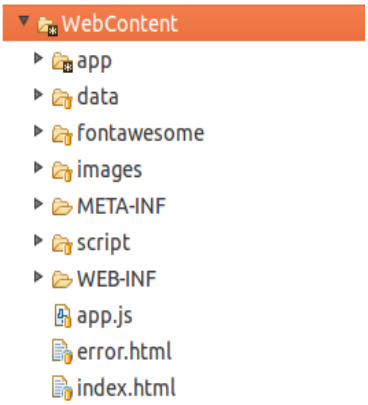
dalam aplikasi, contoh : AlarmChart.js, AlarmGrid.js, dll. Hindari penamaan file yang tidak bermakna dan dikhawatirkan akan menyebabkan kesulitan dalam *debugging* kedepannya, contoh : Panel1.js, Grid.js, dll.

4.6. Struktur folder dan peletakan file bisa dilihat pada bagian “Memulai Project Anda Sendiri”.

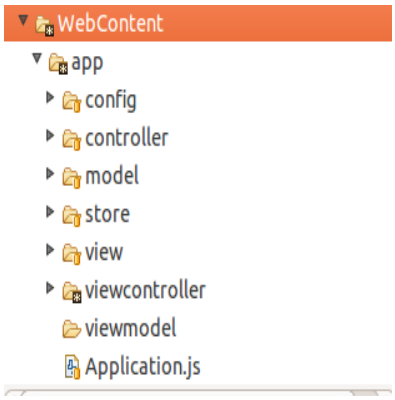
7. Memulai Project Anda Sendiri

1. Buat project baru. Klik File - New - Dynamic Web Project.

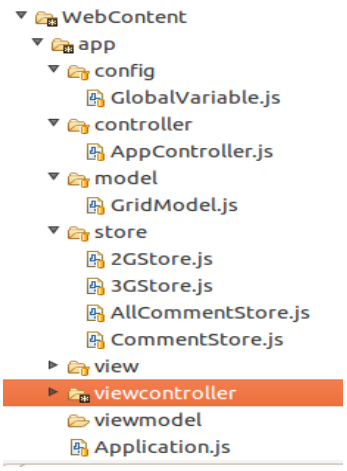
2. Buat folder dibawah *WebContent* dengan struktur seperti gambar dibawah. Untuk aturan *case* huruf dalam penamaan folder ataupun file, bisa dilihat pada bagian *coding style*.



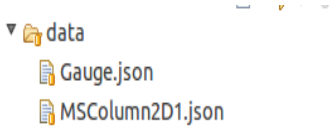
3. Buat folder dibawah folder app sesuai dengan aturan baku dari ExtJS, seperti terlihat dalam gambar dibawah.



4. Folder **app** adalah folder standard untuk ExtJS dan didalamnya terdapat file-file yang telah ditempatkan sesuai dengan posisi nya di dalam arsitektur seperti jika file tersebut berupa controller maka tempatkan di folder controller, jika file tersebut adalah view maka tempatkan di folder view.
Bila aplikasi yang akan dibuat adalah aplikasi besar, di bawah masing-masing folder bisa dibuat folder lagi sesuai dengan modul yang akan dikembangkan.

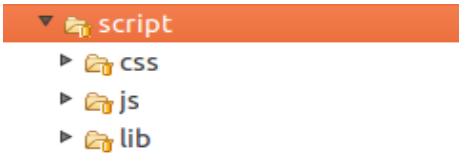


5. Folder **data** merupakan folder yang berisikan berbagai file data static (bisa berupa JSON, CSV ataupun file dengan ekstensi lainnya) yang akan digunakan dalam aplikasi, baik untuk sekedar dummy data maupun data yang memang akan dipakai pada saat production



6. Folder **script** dibawahnya terdapat folder css, js dan lib. Ketiganya merupakan file external diluar arsitektur ExtJS yang akan digunakan didalam aplikasi. Letakkan file sesuai dengan tempat nya masing-masing. Bila file

tersebut berisi file css maka letakkan di folder css, bila file tersebut file Javascript maka letakkan di folder js, dan folder lib digunakan untuk menampung library yang biasanya dalam bentuk folder dan susah untuk dipisahkan seperti sweetalert, bootstrap dll.



7. Setelah semua folder tersusun rapi, kita bisa mulai membuat arsitektur ExtJS kita.

Diawali dari index.html sebagai file html yang akan di-*load* diawal.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>DASHBOARD</title>
   <link rel="stylesheet" href="../EXTJS5/ext-5.0.1/build/packages/ext-theme-
crisp/build/resources/ext-theme-crisp-all.css" />
</head>
<body oncontextmenu="return false;">
          <script type="text/javascript" src="../EXTJS5/ext-5.0.1/build/ext-all.js"></script>
   <!-- ExtJs Main Application -->
   <script type="text/javascript" src="./app.js"></script>
</body>
</html>
```

Paling tidak ada beberapa file yang wajib di-*include* dalam index.html, yakni file

css theme dari ExtJS yang akan dipakai (*contoh, lihat contoh *sourcecode* di atas), setelah itu file ext-all.js (*ext-all-debug.js adalah versi *unminified*-nya) yang merupakan core dari ExtJS, dan file app.js yang merupakan loader dari aplikasi ExtJS kita.

8. Buat file app.js dengan struktur sejajar dengan index.html seperti berikut :

```
Ext.application({
    name: 'APPNAME',
    extend: 'APPNAME.Application'
});
```

9. Di dalam folder app dan buat file Application.js sebagai loader dari arsitektur.

```
Ext.define('APPNAME.Application', {
    extend: 'Ext.app.Application',
    name: 'APPNAME',
    requires : ['APPNAME .config.GlobalVariable'], //config class

    controllers: [
                'AppController' //our app controller
                ],

    launch: function () {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
           items: [{
                xtype: 'mainView' //this is our main view
              }]
        });
```

```
    }
});
```

File ini akan membutuhkan beberapa file lain seperti GlobalVariable.js sebagai config class (*diatas v.5) dan AppController.js sebagai *controller* dan mainView sebagai view utama. Bahwa ExtJS membutuhkan satu *viewport* sebagai container yang akan mengikuti berapapun besarnya browser sehingga file diatas langsung membuat satu *viewport* ketika *launch* aplikasi dan menjadikan mainView sebagai container diatasnya. Ketika anda hanya akan membuat mockup dan tidak melibatkan arsitektur MVC/MVVM anda bisa saja tidak perlu me-*require* config class dan controller namun hanya me-*require* satu mainView saja.

10. Bila menggunakan ExtJS v.5 maka buat config class yakni GlobalVariable.js di dalam folder app-config, seperti contoh *sourcecode* berikut :

```
/**
 * example usage :
 *
 * - setter : APPNAME.config.GlobalVariable.setX(3);
 * - getter : APPNAME.config.GlobalVariable.getX();
 */
Ext.define('APPNAME.config.GlobalVariable',{
    singleton : true,
    config : {
            x : 0,
            y : 0,
    },
    constructor : function(config){
```

```
        this.initConfig(config);
    }
});
```

11. Buat AppController.js di dalam folder app-controller, sebagai berikut :

```
Ext.define('APPNAME .controller.AppController', {
    extend: 'Ext.app.Controller',

    models: [ //include your model here
          ],
    stores: [ //include your store here
          ],
    views : [
          'APPNAME.view.view.MainView' // our main view
          ],
    init: function() {
        this.control({
        });
    }
});
```

12. Buat mainView.js di dalam folder app-view, seperti berikut :

```
Ext.define('APPNAME.view.MainView', {
    extend: 'Ext.container.Container',
    xtype: 'mainView',
    itemId: 'mainView',
    layout : 'fit',
    style : 'background-color: red;'
```

```
});
```

Sampai disini seharusnya aplikasi sudah bisa ditampilkan dan hanya akan menampilkan background dengan warna merah.

13. Untuk membuat *div* diatas view yang sudah kita buat, kita bisa menambahkan items dibawah component yang akan ditambahkan, seperti contoh dibawah ini :

```
Ext.define('APPNAME.view.MainView', {
   extend: 'Ext.container.Container',
   xtype: 'mainView',
   itemId: 'mainView',

   layout : 'border', // using layout border

   items : [{
         xtype : 'container',
         region : 'center', // will be strech
         style : 'background-color: magenta;'
   },{
         xtype : 'panel',
     title : 'Menu Navigation',
         region : 'west',
         width : 175,
         collapsible : true,
         split : true,
         bodyStyle : 'background-color: green;',
   },{
         xtype : 'panel',
```

```
    title : 'Footer Side',
        region : 'south',
        height : 175,
        collapsible : true,
        split : true,
        bodyStyle : 'background-color: yellow;',
  }]
});
```

14. Untuk menambahkan listener / event, kita bisa menambahkan *listeners* dan memanggil event terkait, contoh :

```
        xtype : 'panel',
        title : 'Footer Side',
        region : 'south',
        height : 175,
        collapsible : true,
        split : true,
        bodyStyle : 'background-color: yellow;',
        listeners : {
            afterrender : function(thisCmp){
                alert('Panel has rendered');
                console.log(thisCmp);
            }
        }
```

15. Kita bisa membuat view menjadi lebih modular dengan membuat alias sehingga bisa dipanggil dengan xtype, contoh :

-- MainView.js

```
Ext.define('APPNAME.view.MainView', {
    extend: 'Ext.container.Container',
    xtype: 'mainView',
    itemId: 'mainView',

    requires : ['APPNAME.view.Navigation'],

    layout : 'border', // using layout border

    items : [{
            xtype : 'container',
            region : 'center', // will be strech
            style : 'background-color: magenta;'
    },{
            xtype : 'navigation',
    },{
            xtype : 'panel',
      title : 'Footer Side',
            region : 'south',
            height : 175,
            collapsible : true,
            split : true,
            bodyStyle : 'background-color: yellow;',
    }]
});
```

-- Navigation.js

```
Ext.define('APPNAME.view.Navigation, {
    extend: 'Ext.panel.Panel,
    xtype: 'navigation',
    itemId: 'navigation',

    title : 'Menu Navigation',
    region : 'west',
    width : 175,
    collapsible : true,
    split : true,
    bodyStyle : 'background-color: green;',
});
```

8. Bahan Bacaan

1. **Contoh menggunakan responsiveConfig (v.5 above)**
   https://mazipanneh.wordpress.com/2015/07/30/extjs-custom-responsive-tab-menu-with-responsiveconfig/
2. **Contoh membuat tree beserta live filter**
   https://mazipanneh.wordpress.com/2015/05/11/extjs-tree-filter-live-search-example/
3. **Teknik-teknik meng-Get komponen dalam ExtJs**
   https://mazipanneh.wordpress.com/2015/05/06/extjs-many-ways-to-get-component/
4. **Perbedaan panel dengan container**
   https://mazipanneh.wordpress.com/2015/05/05/extjs-different-panel-vs-container/
5. **Render google maps API v3 di ExtJS**
   https://mazipanneh.wordpress.com/2015/05/04/extjs-render-google-maps-di-ext-container/
6. **Show Hide header dari sebuah panel by hover**
   https://mazipanneh.wordpress.com/2015/04/30/extjs-hide-show-header-panel-by-mouse-over/
7. **Menambahkan style pada header panel**
   https://mazipanneh.wordpress.com/2015/04/28/extjs-menambahkan-style-pada-header-panel/
8. **Contoh menggunakan localStorage HTML 5 untuk menyimpan state**

dari sebuah tab
https://mazipanneh.wordpress.com/2015/03/19/extjs-implement-html5-localstorage-for-saving-tab-state/

9. **Trik memahami dokumentasi ExtJs dengan mudah**
https://mazipanneh.wordpress.com/2015/03/18/extjs-quick-tips-to-read-extjs-documentation/
10. **Menghindari penggunaan global variabel di ExtJs menggunakan config class** https://mazipanneh.wordpress.com/2015/03/06/extjs-avoid-global-variable-usage/
11. **Integrasi FusionChart di ExtJs container**
https://mazipanneh.wordpress.com/2015/03/05/extjs-integrasi-dengan-fusionchart/
12. **UX-ExtJs Datetime field**
https://mazipanneh.wordpress.com/2015/03/04/extjs-best-ux-for-date-time-field/
13. **Membuat combobox multiselection dengan checkbox**
https://mazipanneh.wordpress.com/2015/03/03/extjs-membuat-combobox-multiselection-with-checkbox/
14. **Menempatkan label ditengah container memanfaatkan layouting ExtJs** https://mazipanneh.wordpress.com/2015/03/02/extjs-menempatkan-label-ditengah-container/
15. **Menggunakan Font Awesome untuk aplikasi ExtJs**
https://mazipanneh.wordpress.com/2015/02/05/how-to-use-font-awesome-icon-in-extjs/

9. **Tentang Penulis**

Irfan Maulana, Seseorang dengan profesi sebagai java developer di PT.SML Technologies dari 2013 yang tanpa sengaja kecemplung dan tenggelam dalam mendalami teknologi Sencha ExtJS.

Telah bergelut dengan Sencha ExtJS sejak v.4x.x dan kini sedang mencoba memahami arsitektur di v.5x.x.

Email : mazipanneh@gmail.com

Blog : mazipanneh.com

Facebook : /mazipanneh

Twitter : @Maz_Ipan

Linkedin : /in/irfanmaulanamazipan

Github : mazipan