ExtJS Sencha for dummies

Philippe Depouilly

October 3, 2013

Les points abordés

1. Une brève introduction
   - Motivations
2. Démarrage en douceur
   - Par où débuter
   - Viewport, Panel, Objets, Contrôleurs
   - Par l'exemple...
3. Interactions avec le serveur
   - Parenthèse... Ruby...
4. Un exemple avancé d'auto-complétion
5. Le formulaire d'auto-complétion
   - Les éléments associés
   - Démo...

Introduction

- ExtJS de Sencha est un framework développé au dessus de Javascript, qui permet de développer des interfaces riches dans un navigateur Web avec des fonctionnalités Client-Serveur (AJAX). Et ceci avec les fonctionnalités HTML5 (drag and drop,..)
- ExtJS semble être le plus avancé en terme d'abstraction de Javascript et de fonctionnalités orientées interface utilisateur. Depuis la version 4 il offre un paradigme MVC (Modèle Vue Contrôleur)

Introduction

- ExtJS de Sencha est un framework développé au dessus de Javascript, qui permet de développer des interfaces riches dans un navigateur Web avec des fonctionnalités Client-Serveur (AJAX). Et ceci avec les fonctionnalités HTML5 (drag and drop,..)
- ExtJS semble être le plus avancé en terme d'abstraction de Javascript et de fonctionnalités orientées interface utilisateur. Depuis la version 4 il offre un paradigme MVC (Modèle Vue Contrôleur)
- Une façon simple de se rendre compte du panel de fonctionnalités : `http://docs.sencha.com/extjs/4.2.2/extjs-build/examples/build/KitchenSink`
- Et consulter la documentation qui est indispensable pour s'en sortir avec ExtJS : `http://docs.sencha.com/extjs`

Introduction

- ExtJS de Sencha est un framework développé au dessus de Javascript, qui permet de développer des interfaces riches dans un navigateur Web avec des fonctionnalités Client-Serveur (AJAX). Et ceci avec les fonctionnalités HTML5 (drag and drop,..)
- ExtJS semble être le plus avancé en terme d'abstraction de Javascript et de fonctionnalités orientées interface utilisateur. Depuis la version 4 il offre un paradigme MVC (Modèle Vue Contrôleur)
- Une façon simple de se rendre compte du panel de fonctionnalités : `http://docs.sencha.com/extjs/4.2.2/extjs-build/examples/build/KitchenSink`
- Et consulter la documentation qui est indispensable pour s'en sortir avec ExtJS : `http://docs.sencha.com/extjs`

Motivations

- Arrêter de développer la partie interface utilisateur sur un serveur Web (mix formulaires HTML, PHP,...) et ne laisser sur le serveur que le traitement des données en entrée et sortie.
- Aller au delà de JQuery ou Node.js qui est plutôt orienté Entrées Sorties dynamiques dans un navigateur (AJAX/XMLHttpRequest), qui nécessite toujours d'écrire des formulaires HTML

Motivations

- Arrêter de développer la partie interface utilisateur sur un serveur Web (mix formulaires HTML, PHP,...) et ne laisser sur le serveur que le traitement des données en entrée et sortie.
- Aller au delà de JQuery ou Node.js qui est plutôt orienté Entrées Sorties dynamiques dans un navigateur (AJAX/XMLHttpRequest), qui nécessite toujours d'écrire des formulaires HTML

Par où débuter

Télécharger ExtJS Sencha
http://www.sencha.com/products/extjs/download
Editer un ficher html qui permet de charger :

- le framework (zip javascript)
- l'habillage HTML de l'application (feuille de style)
- l'application

Créer l'arborescence de l'application
demo.js
ext-4.2.2.1144
extjs → ext-4.2.2.1144
index.html
app/view
app/controller

- page html (index.html)

```
<html>
        <head>
        <title>Demo ExtJS</title>
        <link rel="stylesheet" type="text/css" href="extjs/resources/css/ext-all-debug.css" />
        <script src="extjs/ext-all-debug.js" type="text/javascript"></script>
        <script src="demo.js" type="text/javascript"></script>
        </head>
<body>
</body>
</html>
```

- lancement de l'application Sencha (demo.js)

```
Ext.application({
        name: 'App'
        ,controllers:['DemoController']
        ,launch: function() {
                var me = this;
                me.viewport     = Ext.create('App.view.Viewport');
        }
});
```

Remarque : le nom des objets créés correspond à un espace de nommage plaqué sur une arborescence de fichiers (App.view.Viewport correspond au fichier Viewport.js dans le dossier app/view)

Les éléments principaux d'une application

Avec mes mots... de dummy...

- le Viewport est la racine du programme (la zone de la page HTML qui contiendra les objets ExtJS)
- le viewport va contenir un Panel (un conteneur affichable qui va contenir des Panels et/ou des objets)
- dans le modèle MVC, on sépare les objets graphiques (view) le contrôleur (traitements associés aux réflexes click, sélection, entrée/sortie de fenêtre, etc.) ou aux altérations d'objets
- les E/S sont gérées par des modèles (eux même découpés en Model/Proxy/Store)

Par l'exemple...

- Viewport.js

```
Ext.define('App.view.Viewport', {
        extend: 'Ext.container.Viewport',
        layout: 'fit',
        requires:['App.view.DemoPanel'],
        items:[{
                xtype:'demopanel',
        }]
});
```

- DemoPanel.js

```
Ext.define('App.view.DemoPanel', {
        extend:'Ext.panel.Panel',
        alias: 'widget.demopanel',
        title:'MyDemoPanel',
        items:[{
                xtype:'button',
                id: 'demo-button1',
                text:'button1'
        },{
                xtype:'button',
                id: 'demo-button2',
                text:'button2'
        }]
});
```

DemoControleur.js

```
Ext.define('App.controller.DemoController', {
        extend:'Ext.app.Controller'
        ,refs: [
                 { selector:'#demo-button1'  ,ref:'button1'}
         ]
        ,init: function() {
                 var me = this;
                 me.control({
                         ,'#demo-button1' :{click:me.submit }
                         });
         }
        ,submit: function(click,target,event) {
        alert('click');
        }
});
```

Un petit serveur annuaire

```
ActiveLdap::Base.setup_connection(
                :host => 'ldap.math.cnrs.fr',
                :base => 'dc=math,dc=cnrs,dc=fr',
                :allow_anonymous => true,
                :try_sasl => false)

class User < ActiveLdap::Base
        ldap_mapping dn_attribute: 'uid', prefix: '', classes: [ 'inetOrgPerson' ]
end

class Hash
        def map!()
                each do |k,v| store(k,yield(v)) end
        end
end

class Application < Sinatra::Base
        get '/search/:login' do
                users = User.search(:filter => '(|(sn=*'+params[:login]+'*)
(uid=*'+params[:login]+'*)(cn=*'+params[:login]+'*)(mail=*'+params[:login]+'*@*))',
                 :scope => :sub, :attributes => ['uid', 'cn', 'sn', 'mail'])
                users.map! {|k,v| if (v.count==1)
                                v.first.map! do |v| (v.length==1) ? v.first : v; end
                        else
                                v.map! do |v| (v.length==1) ? v.first : v; end
                        end;
                }
                content_type 'application/json'
                erb(:'search/login', { :layout => false, :locals => { :users => users } })
        end
end
```

Un formulaire avec auto-complétion (1)

Première partie de la grille avec la zone (combo) qui permet l'auto-complétion pour rechercher un individu via l'API REST

```
Ext.define('App.view.Users', {
 extend: 'Ext.grid.Panel'
,alias: 'widget.users'
    ,border: false
    ,autoscroll: true
    ,id: 'users-panel'
    ,anchor: '100%'
    ,title: 'Users'
    ,tbar: [{
        xtype: 'combo',
            hideTrigger: true,
   typeAhead: true,
            queryMode: 'remote',
            flex: 1,
            fieldLabel: 'search user',
            emptyText: 'at least 3 characters',
            displayField: 'uid',
            valueField: 'uid',
            tpl: Ext.create('Ext.XTemplate',
             '<tpl for=".">',
                '<div class="x-boundlist-item">{[values.cn[0]]} - {mail}</div>',
                '</tpl>'
                ),
            displayTpl: Ext.create('Ext.XTemplate',
                '<tpl for=".">',
                '{[values.cn[0]]} - {mail}',
                '</tpl>'
                ),
            id: 'search-user'
        }]
```

Un formulaire avec auto-complétion (2)

Fin de la grille avec juste les deux colonnes où seront insérées les choix

```
....
,columns: [{
text: 'Nom',
            width: '50%',
            sortable: true,
            dataIndex: 'name',
            editable: false
}, {
text: 'Courriel',
sortable: true,
width: '49%',
            dataIndex: 'mail',
            editable: false
}]
});
```

Le contrôleur associé 1/3

- refs : permet de créer des accesseur en fonction de l'id d'un objet : users-panel deviant accessible via getUsers() par exemple
- me.control : je capture des actions (click, choisir dans la liste, etc.) et associe des méthodes listées à la suite
- par exemple à l'affichage du users-panel, je fabrique les conteneurs de données (stores) et les actions REST associées

```
Ext.define('App.controller.DemoController', {
 extend:'Ext.app.Controller'
,requires:['App.model.Users']
    ,refs: [
         { selector:'#demo-button1'  ,ref:'button1'}
,{ selector:'#users-panel'  ,ref:'users'}
,{ selector:'#search-user'  ,ref:'search'}
         ]
    ,init: function() {
         var me = this;
                me.control({
                        '#users-panel' :{    render:me.setupStores}
                        ,'#demo-button1' :{    click:me.submit }
     ,'#search-user' :{change:me.searchUser, select:me.addUser}
                 });
         }
...
```

Le contrôleur associé 2/3

```
,setupStores: function(render,b,c) {
var me = this
                ,application  = me.getApplication()
   ,gridStore  = Ext.StoreManager.get('users-store')
                ,usersStore  = Ext.StoreManager.get('users-search-store')
                // Creation du Store (conteneur de donnees) pour la fenetre d'auto-completion
   if (!usersStore) {
       proxy=Ext.create('App.proxy.Users');
                   usersStore=Ext.create('Ext.data.Store',{
                               storeId:'users-search-store'
                                ,model:'App.model.Users'
                                ,autoLoad:false
                                ,proxy: proxy
                ,reader: {
                                        root:
'entries'
                                        ,totalProperty: 'totalCount'
                }
});
}
// Creation du Store (conteneur de donnees) de la grille d'affichage des noms et mail
             if (!gridStore) {
                gridStore=Ext.create('Ext.data.Store',{
                            storeId:'users-store'
 ,fields:['name', 'mail']
                });
}
me.getSearch().bindStore(usersStore);
me.getUsers().bindStore(gridStore);
//console.log(me.getSearch());
}
```

Le contrôleur associé 3/3

```
        ,searchUser: function(combo, word, eOpts){
         var me  = this;
         if ((null != word)&&(word.length>2)) {
                 console.log('search',word);
                 console.log('search',combo.getStore());
                 combo.getStore().load({word:word});
         }
//console.log(combo.getStore());
}
,addUser: function(combo, select, eOpts){
         var me  = this;
console.log(select[0].data.cn[0]);
         var rec = [{'name':select[0].data.cn[0],'mail':select[0].data.mail}];
         me.getUsers().getStore().insert(0, rec);
         me.getUsers().getStore().commitChanges();
}
,submit: function(click,target,event) {
alert('click');
        }
});
```

Le Proxy et le Model pour l'auto-complétion

```
Ext.define('App.model.Users', {
         extend: 'Ext.data.Model'
        ,fields: [
                {name: 'uid', type: 'string'}
               ,{name: 'sn', type: 'string'}
               ,{name: 'cn', type: 'auto'} // auto car le service REST renvoie un tableau de noms,
               ,{name: 'mail', type: 'string'}
        ]
});

Ext.define('App.proxy.Users', {
         extend: 'Ext.data.proxy.Ajax'
        ,constructor: function(config) {
                var me     = this;
                me.callParent(config);
                me.url     = '/search/';;
                me.reader = { root:  'entries', totalProperty: 'totalCount'};
        }
       ,buildRequest: function(operation, callback, scope) {
        var me      = this
           ,params  = {}
           ,url     = me.url+Ext.getCmp('search-user').getValue();
        request = Ext.create('Ext.data.Request',{
             params:    params
            ,action:    operation.action
            ,records:   operation.records
            ,operation: operation
            ,url: url
        });
        request.url = me.buildUrl(request);
        operation.request = request;
        return request;
     }
});
```

Démo...

Effet démo....