Développement de clients web cartographiques avec OpenLayers, ExtJS et GeoExt

Nicolas Moyroud
nicolas.moyroud@teledetection.fr
http://libreavous.teledetection.fr

Version du support : 17 mars 2013

Présentation disponible sur le site `http://libreavous.teledetection.fr`

Plan

Qu'est-ce qu'OpenLayers ?

- Librairie javascript utilisable dans une page web
- C'est une API (Application Programming Interface) qui propose des objets pour créer des cartes dynamiques
- Le code d'OpenLayers est entièrement exécuté côté client par le navigateur
- OpenLayers est disponible sous la licence libre FreeBSD
- La librairie est téléchargeable sur le site `http://www.openlayers.org`
- La documentation d'API, un manuel utilisateur et de nombreux exemples sont disponibles sur le site
- Deux livres en anglais sont disponibles : "Openlayers Beginner's Guide" et "Openlayers Cookbook"

Petit historique

- API Google Maps = première API cartographique pour le web sortie en Février 2005 (gratuite dans certains cas mais pas libre car soumise à des conditions et des restrictions d'utilisation)
- La première version stable de l'API OpenLayers sortie en Juin 2006 a été développée par la société MetaCarta
- Le développement est maintenant géré par l'OSGeo
- Version stable actuelle d'Openlayers : 2.12 (sortie en Juin 2012)
- La version 2.12 a apporté de nombreuses améliorations, notamment pour le support des terminaux mobiles (smartphones, tablettes)
- Une réécriture complète est prévue pour la future version 3

Plan

Intégration d'OpenLayers dans une page

- L'archive téléchargée sur le site doit être décompressée dans un répertoire accessible à travers un serveur web
- OpenLayers est appelé grâce à une balise <script> dans l'en-tête

Exemple d'intégration d'OpenLayers dans une page HTML

```
<html>
<head>
    <title>Carte avec OpenLayers</title>
    <script src="OpenLayers-2.12/lib/OpenLayers.js" type="text/javascript">
    </script>
    <script src="carte.js" type="text/javascript">
    </script>
</head>
<body onLoad="init()">
    <div id="map"></div>
</body>
</html>
```

Création d'une carte

- carte.js définit une fonction init() qui contient le code de la carte
- La création de la carte est réalisée par un objet OpenLayers.Map avec en paramètre l'id de la div HTML dans laquelle sera chargée la carte
- Cet objet propose des méthodes pour ajouter des couches, des contrôles, définir le positionnement de la carte, ...

carte.js

```
1 function init() {
2     var map = new OpenLayers.Map('map');
3     var osmLayer = new OpenLayers.Layer.OSM();
4     map.addLayer(osmLayer);
5     map.zoomToMaxExtent();
6     map.addControl(new OpenLayers.Control.OverviewMap());
7 }
```

Positionnement par défaut de la carte

- Au moment du chargement de la page, il faut afficher une zone déterminée sur la carte
- Après avoir chargé au moins une couche de fond sur la carte, il y a plusieurs possibilités pour le positionnement :

Positionnement par défaut de la carte

- Au moment du chargement de la page, il faut afficher une zone déterminée sur la carte
- Après avoir chargé au moins une couche de fond sur la carte, il y a plusieurs possibilités pour le positionnement :
  - centrer sur un point avec un certain niveau de zoom

Positionnement par centrage et zoom

```
var center = new OpenLayers.LonLat(2.70,47.10);
var zoom = 6;
map.setCenter(center, zoom);
```

Positionnement par défaut de la carte

- Au moment du chargement de la page, il faut afficher une zone déterminée sur la carte
- Après avoir chargé au moins une couche de fond sur la carte, il y a plusieurs possibilités pour le positionnement :
  - centrer sur un point avec un certain niveau de zoom
  - afficher l'étendue maximale de toutes les couches chargées

Étendue maximale des couches

```
map.zoomToMaxExtent();
```

Positionnement par défaut de la carte

- Au moment du chargement de la page, il faut afficher une zone déterminée sur la carte
- Après avoir chargé au moins une couche de fond sur la carte, il y a plusieurs possibilités pour le positionnement :
  - centrer sur un point avec un certain niveau de zoom
  - afficher l'étendue maximale de toutes les couches chargées
  - définir le rectangle englobant de la zone affichée

Définir un rectangle englobant

```
var mapBounds = new OpenLayers.Bounds(-5,43,8,50);
map.zoomToExtent(mapBounds);
```

Plan

Les systèmes de projection

- OpenLayers utilise la librairie libre Proj4js pour la gestion des systèmes de projection (portage de proj4 en Javascript)
- Chaque système de projection est connu grâce à l'objet OpenLayers.Projection et son code EPSG

`http://proj4js.org`

`http://spatialreference.org/ref/`

Les systèmes de projection

- OpenLayers utilise la librairie libre Proj4js pour la gestion des systèmes de projection (portage de proj4 en Javascript)
- Chaque système de projection est connu grâce à l'objet OpenLayers.Projection et son code EPSG
- Pour éviter de répéter plusieurs fois l'appel à OpenLayers.Projection, on peut stocker au début du code les systèmes de projection utilisés dans des variables JavaScript

Chargement de deux systèmes de projection

```
var epsg4326 = new OpenLayers.Projection("EPSG:4326");
var epsg900913 = new OpenLayers.Projection("EPSG:900913");
```

Les systèmes de projection

- OpenLayers utilise la librairie libre Proj4js pour la gestion des systèmes de projection (portage de proj4 en Javascript)
- Chaque système de projection est connu grâce à l'objet OpenLayers.Projection et son code EPSG
- Pour éviter de répéter plusieurs fois l'appel à OpenLayers.Projection, on peut stocker au début du code les systèmes de projection utilisés dans des variables JavaScript
- Il est possible d'ajouter soi-même la définition d'autres systèmes de projection

Ajout d'un nouveau système de projection

```
Proj4js.defs["EPSG:32661"] = "+proj=stere
    +lat_0=90 +lat_ts=90 +lon_0=0 +k=0.994 +x_0=2000000 +y_0=2000000
    +ellps=WGS84 +datum=WGS84 +units=m +no_defs";
```

Les méthodes de projection

- La méthode transform permet de reprojeter d'un système vers un autre

```
var epsg4326 = new OpenLayers.Projection("EPSG:4326");
var epsg900913 = new OpenLayers.Projection("EPSG:900913");
var center = new OpenLayers.LonLat(2.70,47.10).transform(
    epsg4326,epsg900913
);
```

Les méthodes de projection

- La méthode transform permet de reprojeter d'un système vers un autre
- La méthode getProjectionObject permet de récupérer le système de projection de la carte

```
var epsg4326 = new OpenLayers.Projection("EPSG:4326");
var center = new OpenLayers.LonLat(2.70,47.10).transform(
    epsg4326,map.getProjectionObject()
);
```

Les méthodes de projection

- La méthode transform permet de reprojeter d'un système vers un autre
- La méthode getProjectionObject permet de récupérer le système de projection de la carte
- displayProjection permet de choisir un projection différente pour l'affichage des coordonnées du pointeur

```
map.addControl(new OpenLayers.Control.MousePosition(
    {displayProjection: epsg4326}
));
```

Plan

Contrôles OpenLayers

- Il existe différents contrôles permettant de manipuler la carte
- Tous les contrôles sont des sous-objets de l'objet OpenLayers.Control
- Des contrôles sont ajoutés par défaut à la création de l'objet map

Exemple de chargement de contrôles

```
1  // chargement d'une carte sans aucun controle
2  var map = new OpenLayers.Map('map', {controls:[]});
3  // deplacement et zoom avec la souris
4  map.addControl(new OpenLayers.Control.Navigation());
5  // grande barre de zoom
6  map.addControl(new OpenLayers.Control.PanZoomBar());
7  // carte de vue globale
8  map.addControl(new OpenLayers.Control.OverviewMap());
9  // coordonnees du pointeur de la souris
10 map.addControl(new OpenLayers.Control.MousePosition());
11 // selecteur de couches
12 map.addControl(new OpenLayers.Control.LayerSwitcher());
```

Plan

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couches OpenStreetMap

```
1 var osmLayer1 = new OpenLayers.Layer.OSM();
2 var osmLayer2 = new OpenLayers.Layer.OSM("OpenMapQuest",
3     ["http://otile1.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png",
4      "http://otile2.mqcdn.com/tiles/1.0.0/osm/${z}/${x}/${y}.png"],
5     {numZoomLevels:19});
6 map.addLayers([osmLayer1,osmLayer2]);
```

http://www.openstreetmap.org

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couches Google

```
1  // couche Google par defaut si on ne precise pas le type
2  var gmap = new OpenLayers.Layer.Google("Google Streets",
3      {numZoomLevels: 20});
4  // autres couches Google
5  var gphy = new OpenLayers.Layer.Google("Google Physical",
6      {type: google.maps.MapTypeId.TERRAIN});
7  var ghyb = new OpenLayers.Layer.Google("Google Hybrid",
8      {type: google.maps.MapTypeId.HYBRID, numZoomLevels: 20});
9  var gsat = new OpenLayers.Layer.Google("Google Satellite",
10     {type: google.maps.MapTypeId.SATELLITE, numZoomLevels: 22});
11 map.addLayers([gmap, gphy, ghyb, gsat]);
```

http://openlayers.org/dev/examples/google-v3.html

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couche WMS

```
1  var contourPays = new OpenLayers.Layer.WMS(
2      "France - Limites admin",
3      "http://localhost/cgi-bin/mapserv?
4       map=/var/www/data/TP2/cartoEtudiants.map&",
5      {
6          projection: epsg4326, layers: "FRA_adm2",
7          transparent: true, format: 'image/png'
8      },{
9          singleTile: true, isBaseLayer : true
10     }
11 );
12 map.addLayer(contourPays);
```

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couche de points issus d'un fichier texte

```
1  var parkings = new OpenLayers.Layer.Text(
2      "Parkings",
3      {
4          location:"markerlist.txt", projection: epsg4326
5      }
6  );
7  map.addLayer(parkings);
8
9  // Extrait du fichier markerlist.txt
10 lon      lat          title       description  icon
11 4.86580 45.78177     Parking 1   500 places   marker_red.png
12 4.87056 45.78462     Parking 2   200 places   marker_blue.png
```

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couche de polygones issus d'un fichier KML

```
1 var regions = new OpenLayers.Layer.Vector("Regions", {
2     protocol: new OpenLayers.Protocol.HTTP({
3         url: "regions.kml",
4         format: new OpenLayers.Format.KML({}),
5         projection: epsg4326
6     }),
7 });
8 map.addLayer(regions);
```

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couche de points issus d'un fichier OSM

```
1 var recyclage = new OpenLayers.Layer.Vector("Points de recyclage", {
2     protocol: new OpenLayers.Protocol.HTTP({
3         url: "recyclage.osm",
4         format: new OpenLayers.Format.OSM({}),
5         projection: epsg4326
6     }),
7 });
8 map.addLayer(recyclage);
```

Chargement de couches

- OpenLayers permet le chargement de différents types de couches grâce aux sous-objets de OpenLayers.Layer

Couche d'objets vecteurs issus d'une réponse GeoJSON

```
1  var lieuxStages = new OpenLayers.Layer.Vector(
2      "Lieux des stages",
3      {
4          protocol: new OpenLayers.Protocol.HTTP({
5              url: "lieux_stages.php?id_promo=2010",
6              format: new OpenLayers.Format.GeoJSON()
7          }),
8          projection: epsg4326,
9          strategies: [new OpenLayers.Strategy.Fixed()]
10     }
11 );
```

Plan

Les données au format GeoJSON

- Le format JSON (Javascript Object Notation) est un format de données textuel qui permet de représenter des objets Javascript
- Les données recupérées par un code JavaScript sont directement utilisables sans conversion
- En PHP, il existe une fonction *json_encode* qui transforme un tableau PHP en JSON
- Le format GeoJSON est une extension de JSON aux objets géographiques
- La fonction PHP ne gère malheureusement pas l'extension GeoJSON, mais la fonction PostGIS ST_AsGeoJSON fait une partie du travail

`http://fr.wikipedia.org/wiki/JavaScript_Object_Notation`

`http://geojson.org/geojson-spec.html`

Les stratégies de chargement

- L'objet OpenLayers.Strategy permet de définir la (ou les) manière(s) de charger les objets dans une couche vecteur
- BBOX : les seuls objets chargés sont ceux contenus dans la zone affichée (rechargement à chaque zoom ou déplacement)
- Fixed : tous les objets de la couche sont chargés dès le départ (il n'y a plus de rechargement ensuite)
- Cluster : les objets spatialement proches sont regroupés
- Paging : les objets sont affichées par bloc avec une pagination

`http://geotribu.net/node/47`
`http://geotribu.net/node/90`

Les stratégies de chargement

Exemple d'utilisation des stratégies

```
1  var lieuxStages = new OpenLayers.Layer.Vector(
2      "Lieux des stages",
3      {
4          protocol: new OpenLayers.Protocol.HTTP({
5              url: "lieux_stages.php?id_promo=2010",
6              format: new OpenLayers.Format.GeoJSON()
7          }),
8          projection: epsg4326,
9          strategies: [
10             new OpenLayers.Strategy.Fixed(),
11             new OpenLayers.Strategy.Cluster()
12         ]
13     }
14 );
```

Les contrôles spécifiques aux couches vecteurs

- Il existe des contrôles spécifiques pour manipuler les objets contenus dans les couches vecteurs
- selectFeature rend sélectionnables les objets d'une couche vecteur

Contrôle de sélection selectFeature

```
1 var lieuxStages = new OpenLayers.Layer.Vector("Lieux des stages");
2 var selectFeatureControl = new OpenLayers.Control.SelectFeature(
3     lieuxStages
4 );
5 map.addControl(selectFeatureControl);
6 // il faut activer le controle manuellement
7 selectFeatureControl.activate();
8 map.addLayer(lieuxStages);
```

Les contrôles spécifiques aux couches vecteurs

- Il existe des contrôles spécifiques pour manipuler les objets contenus dans les couches vecteurs
- selectFeature rend sélectionnables les objets d'une couche vecteur
- drawFeature rend possible la saisie de nouveaux objets géographiques dans la couche

Contrôle d'ajout drawFeature

```
1 // les sous-objets de OpenLayers.Handler permettent de specifier le type
2 // d'objets geographiques ajoutables dans la couche vecteur
3 var drawControl = new OpenLayers.Control.DrawFeature(
4     lieuxStages, OpenLayers.Handler.Polygon
5 );
6 map.addControl(drawControl);
7 drawControl.activate();
```

Les contrôles spécifiques aux couches vecteurs

- Il existe des contrôles spécifiques pour manipuler les objets contenus dans les couches vecteurs
- selectFeature rend sélectionnables les objets d'une couche vecteur
- drawFeature rend possible la saisie de nouveaux objets géographiques dans la couche
- modifyFeature permet de modifier les objets existants dans la couche

Contrôle de modification modifyFeature

```
1 var modifyControl = new OpenLayers.Control.ModifyFeature(
2     lieuxStages
3 );
4 map.addControl(modifyControl);
5 modifyControl.activate();
```

Les styles applicables aux couches vecteurs

- Quand une couche vecteur est chargée, OpenLayers applique un style d'affichage par défaut à ses objets
- Il est possible de définir des styles personnalisés avec des objets Style :
  - pour les objets normaux (default)

Style des objets normaux

```
1 var lieuxStyleDefault = new OpenLayers.Style({
2     'strokeColor': '#0000aa',
3     'strokeWidth': 2,
4     'fillOpacity': 0.80,
5     'fillColor': '#669933',
6     'pointRadius': 5
7 });
```

http://geotribu.net/node/283

Les styles applicables aux couches vecteurs

- Quand une couche vecteur est chargée, OpenLayers applique un style d'affichage par défaut à ses objets
- Il est possible de définir des styles personnalisés avec des objets Style :
  - pour les objets normaux (default)
  - pour les objets sélectionnés (select)

Style des objets sélectionnés

```
1 var lieuxStyleSelected = new OpenLayers.Style({
2     'strokeColor': '#000000',
3     'strokeWidth': 2,
4     'fillOpacity': 0.80,
5     'fillColor': '#aa0000',
6     'pointRadius': 7
7 });
```

http://geotribu.net/node/283

Les styles applicables aux couches vecteurs

- Quand une couche vecteur est chargée, OpenLayers applique un style d'affichage par défaut à ses objets
- Il est possible de définir des styles personnalisés avec des objets Style :
  - pour les objets normaux (default)
  - pour les objets sélectionnés (select)
- Puis de les appliquer à la couche grâce à un objet StyleMap

Applications des styles à la carte

```
1 lieuxStages.styleMap = new OpenLayers.StyleMap({
2     'default': lieuxStyleDefault,
3     'select': lieuxStyleSelected
4 });
```

http://geotribu.net/node/283

Les règles

- Une règle OpenLayers.Rule permet de définir les conditions d'application de styles en fonction de valeurs attributaires des objets vecteurs
- Les conditions sont exprimées grâce à des filtres OpenLayers.Filter
- Le style correspondant est fixé grâce à la propriété symbolizer de la règle
- Dans l'exemple qui suit la source de données GeoJSON renvoie une liste de villes avec un champ attributaire population
- On fixe la couleur et la taille des points en tenant compte du fait que la population est inférieure ou supérieure à 250000

Les règles

Exemple d'utilisation des règles

```
1  var villesLayer = new OpenLayers.Layer.Vector("Villes", {
2      protocol: new OpenLayers.Protocol.HTTP({
3          url: "villes.php",
4          format: new OpenLayers.Format.GeoJSON()
5      }),
6      projection: epsg4326,
7      strategies: [new OpenLayers.Strategy.Fixed()]
8  });
9
10 var villesStyle = new OpenLayers.Style(
11     strokeColor: "green",
12     strokeWidth: 2,
13     strokeOpacity: 0.5,
14     label: '${population}'
15 );
```

Les règles

Exemple d'utilisation des règles - suite

```
16 var rule_pop_low = new OpenLayers.Rule({
17     filter: new OpenLayers.Filter.Comparison({
18         type: OpenLayers.Filter.Comparison.LESS_THAN,
19         property: 'population',
20         value: 250000
21     }),
22     symbolizer: {
23         fillColor: '#0000aa',
24         pointRadius: 8
25     }
26 });
```

Les règles

Exemple d'utilisation des règles - suite

```
27 var rule_pop_high = new OpenLayers.Rule({
28     filter: new OpenLayers.Filter.Comparison({
29         type: OpenLayers.Filter.Comparison.GREATER_THAN_OR_EQUAL_TO,
30         property: 'population',
31         value: 250000
32     }),
33     symbolizer: {
34         fillColor: '#00aa00',
35         pointRadius: 12
36     }
37 });
```

Les règles

Exemple d'utilisation des règles - fin

```
38 villesStyle.addRules([rule_pop_low,rule_pop_high]);
39 var villesStyleMap = new OpenLayers.StyleMap({
40     default: villesStyle
41 });
42
43 villesLayer.styleMap = villesStyleMap;
44 map.addLayer(villeLayer);
```

Plan

ExtJS : une librairie web 2.0

- ExtJS est une librairie JavaScript (non cartographique) téléchargeable sur `http://www.sencha.com/products/extjs/`
- Grâce aux appels AJAX, ExtJS permet de réaliser facilement des interfaces web 2.0 qui se rapproche des applications bureautiques
- Il existe d'autres librairies JavaScript équivalentes : JQuery, Mootools, Scriptaculous, Dojo, ...
- Elle est développée par la société Sencha
- Elle est disponible sous licence libre GNU/GPL si elle utilisée pour développer une application libre et sous licence commerciale pour une application non libre
- La dernière version stable est la 4.0
- Pour le support de l'interaction avec OpenLayers, la dernière version utilisable est la 3.4

Présentation de ExtJS 3.4

- De la même manière que OpenLayers, ExtJS propose une API objet
- La documentation de l'API 3.4 est disponible à l'adresse `http://docs.sencha.com/ext-js/3-4/`
- Une page d'exemples : `http://examples.extjs.eu`
- Il existe différents types d'objets :
  - des blocs pour structurer l'interface des applications (panels)
  - des éléments de formulaires plus avancés que les formulaires HTML (listes liées, contenus de listes provenant d'un appel AJAX)
  - des tableaux de données (grids)
  - des structures arborescentes dépliables (trees)
  - des graphiques (charts)
  - des zones réactives au glisser/déposer (drag&drop)
  - et beaucoup d'autres choses...

Plan

Mise en place dans une page HTML

- Dans l'en-tête de la page HTML il faut intégrer la feuille de style générale de ExtJS
- On peut éventuellement y ajouter la feuille de style d'un thème différent du thème bleu par défaut (ici le thème gris fourni avec la librairie)

Intégration de ExtJS dans une page HTML

```
<link rel="stylesheet" type="text/css"
    href="ext-3.4.0/resources/css/ext-all.css"></link>
<link rel="stylesheet" type="text/css"
    href="ext-3.4.0/resources/css/xtheme-gray.css" /></link>
```

Mise en place dans une page HTML

- Il faut ensuite intégrer dans des balises <script> les deux fichiers Javascript de la librairie : ext-base.js et ext-all.js
- On peut aussi ajouter le fichier de langue française ext-lang-fr.js
- L'utilisation de ExtJS se fait de la même manière qu'avec OpenLayers : dans un fichier JavaScript séparé (ici code.js)

Intégration de ExtJS dans une page HTML

```
<script src="ext-3.4.0/adapter/ext/ext-base.js"
    type="text/javascript"></script>
<script src="ext-3.4.0/ext-all.js"
    type="text/javascript"></script>
<script src="ext-3.4.0/src/locale/ext-lang-fr.js"
    type="text/javascript"></script>
<script src="code.js" type="text/javascript"></script>
```

Structure générale d'une page avec ExtJS

- Dans le script code.js, il faut définir une fonction init() appelée dans le fichier HTML par l'événement onload de la balise <body>
- La création de la page est réalisée par un objet Ext.Viewport
- Cet objet est toujours situé à la fin de la fonction init() car il intègre les panels créés dans le code qui précède

Structure d'une page avec l'objet Viewport

```
1 new Ext.Viewport({
2     layout: "border",
3     defaults: {
4         split: true
5     },
6     items: [mapPanel, westPanel, southPanel]
7 });
```

Plan

Les panels

- Les panels sont des blocs positionnés sur la fenêtre en fonction des points cardinaux (west, east, south, north, center)
- Chaque panel contient d'autres éléments ExtJS
- Sauf pour le panel center, on peut définir la taille d'un panel
- Le panel en position center occupe automatiquement la place laissée par les autres

Un panel ExtJS positionné à gauche (west)

```
1 var westPanel = new Ext.Panel({
2     region  : 'west',
3     border : false,
4     width  : 300,
5     minSize: 275,
6     maxSize: 325,
7     collapsible: true,
8     items : [promosSelectionForm]
9 });
```

Liste déroulante remplie depuis une base de données

- Dans un formulaire ExtJS, on peut créer des listes déroulantes dont le contenu provient d'une base de données
- Un script (par exemple PHP) interroge la base et renvoie les données
- La requête AJAX d'appel du script est lancée par un magasin de données (store) qui charge ensuite les données
- Si les données sont renvoyées au format JSON, on peut utiliser un magasin JsonStore

Magasin de données JSON

```
1 var promosStore = new Ext.data.JsonStore({
2     url : 'liste_promos.php', // script d'acces a la base
3     fields : ['display', 'value'],
4     root : 'rows',
5     autoLoad : true
6 });
```

Liste déroulante remplie depuis une base de données

- La liste est créée grâce à l'objet form.ComboBox
- Elle utilise le magasin de données pour remplir ses valeurs
- valueField et displayField définissent les champs du magasin qui correspondent aux valeurs et aux contenus affichés dans la liste

Création d'une liste

```
1  var promosCombo = new Ext.form.ComboBox({
2      id : 'promosCombo',
3      fieldLabel : "Promotion",
4      triggerAction : 'all',
5      emptyText : "Choisir une promotion",
6      editable : false,
7      store : promosStore,
8      mode : 'local',
9      valueField : 'value',
10     displayField : 'display'
11 });
```

Création d'un formulaire

- Un formulaire est créé grâce à l'objet FormPanel
- items définit ses éléments (listes, champs texte)
- buttons définit ses boutons (envoi, remise à zéro)

Exemple de formulaire

```
1  var promosSelectionForm = new Ext.FormPanel({
2      id : 'promosSelection',
3      title : "Liste des promotions",
4      frame : true,
5      width : '100%',
6      buttonAlign : 'center',
7      labelAlign : 'left',
8      labelWidth : 70,
9      items : [promosCombo],
10     buttons : [submitButton]
11 });
```

Tableaux de données

- Un objet grid.GridPanel permet de représenter des données sous forme tabulaire en lignes et colonnes
- Les données sont chargées grâce à un magasin (par exemple un JsonStore)
- Un objet grid.ColumnModel sert à définir l'ordre, l'apparence (taille, label) et les propriétés des colonnes (triables, masquables)
- dataIndex fixe la correspondance entre les champs du modèle de colonnes et ceux du magasin
- Un objet grid.RowNumberer permet d'ajouter un compteur de lignes

Tableaux de données

Modèle de colonnes ColumnModel

```
1  var lieuxStagesGridModel = new Ext.grid.ColumnModel({
2      columns: [
3          new Ext.grid.RowNumberer(),
4          {
5              id : "id", header : "Id",
6              width : 25, dataIndex : "numstage", hidden: true
7          }, {
8              id : "datedebut", header : "Date de debut",
9              width : 120, dataIndex : "datedebut", sortable: true,
10             renderer: Ext.util.Format.dateRenderer('l d/m/Y')
11         }, {
12             id : "nometud", header : "Nom de l'etudiant",
13             width : 110, dataIndex : "nometud", sortable: true
14         }
15     ]
16 });
```

Tableaux de données

Tableau de données GridPanel

```
17 var gridPanel = new Ext.grid.GridPanel({
18     id : 'tableauAttrib',
19     border: false,
20     stripeRows: true,
21     store : lieuxStagesStore,
22     colModel: lieuxStagesGridModel
23 });
```

Plan

GeoExt : la "librairie-colle" OpenLayers/ExtJS

- GeoExt est une librairie Javascript libre sous licence BSD
- Elle est développée par la société Camptocamp
- La librairie est téléchargeable sur le site officiel `http://geoext.org`
- La documentation d'API et de nombreux exemples sont disponibles sur le site
- Son but est de faciliter le couplage entre OpenLayers et ExtJS
- Elle n'est pas autonome et nécessite l'inclusion préalable de OpenLayers et ExtJS pour pouvoir fonctionner

Exemples d'utilisation de GeoExt

- Afficher une carte OpenLayers dans un panel ExtJS
- Lien entre les lignes d'un gridPanel ExtJS et des objets vecteurs OpenLayers
- Afficher un popup ExtJS au moment du clic sur un objet vecteur OpenLayers
- Structure arborescente ExtJS contenant les couches OpenLayers
- Création de barres (slider) ExtJS pour régler la transparence d'une couche OpenLayers
- Afficher une liste déroulante ExtJS contenant les échelles d'affichage disponibles pour la carte OpenLayers
- etc...

Plan

Mise en place dans une page

- Il faut inclure dans l'en-tête de la page HTML le lien vers la feuille de style de GeoExt
- Il faut également inclure le lien vers le fichier Javascript de la librairie
- Attention de réaliser cette dernière inclusion impérativement **après** celles d'OpenLayers et de ExtJS
- Le code utilisant GeoExt prendra place avec celui de ExtJS et OpenLayers dans la fonction init() de carte.js

Inclusion de la librairie GeoExt

```
<head>
// ... feuilles de style OpenLayers et ExtJS
<link rel="stylesheet" type="text/css"
    href="lib/GeoExt/resources/css/geoext-all.css"></link>
// ... inclusion des librairies OpenLayers et ExtJS
<script src="lib/GeoExt/script/GeoExt.js" type="text/javascript"></script>
<script src="carte.js" type="text/javascript"></script>
</head>
```

Plan

Inclusion d'une carte dans un panel

- Pour ajouter une carte OpenLayers dans une interface ExtJS, il suffit d'utiliser un objet MapPanel de GeoExt
- Ce MapPanel se comportera ensuite comme un panel ExtJS normal

Exemple de MapPanel GeoExt

```
1 var mapPanel = new GeoExt.MapPanel({
2     map: map, // un objet OpenLayers.Map
3     region : 'center',
4     height: 600,
5     width: 800,
6     title: 'Lieux des stages du Master',
7     collapsible: false,
8     extent: mapBounds // un objet OpenLayers.Bounds
9 });
```

Magasin de données géographiques

- GeoExt offre la possibilité de créer un magasin de données géographiques utilisable par un gridPanel de ExtJS
- Il faut utiliser l'objet data.FeatureStore en précisant le format des données récupérées (par exemple GeoJSON)

Exemple de FeatureStore GeoExt

```
1  lieuxStagesStore = new GeoExt.data.FeatureStore({
2      layer: lieuxStages,
3      idProperty: 'numstage',
4      fields: [
5          {name: 'numstage', type: 'int'},
6          {name: 'datedebut', type: 'date', dateFormat: 'Y-m-d'},
7          {name: 'datefin', dateFormat: 'Y-m-d'},
8          {name: 'sujet', type: 'string'},
9          {name: 'nometud', type: 'string'},
10         {name: 'prenometud', type: 'string'}
11     ],
```

Magasin de données géographiques

- GeoExt offre la possibilité de créer un magasin de données géographiques utilisable par un gridPanel de ExtJS
- Il faut utiliser l'objet data.FeatureStore en précisant le format des données récupérées (par exemple GeoJSON)

Exemple de FeatureStore GeoExt - fin

```
12      proxy: new GeoExt.data.ProtocolProxy({
13          protocol: new OpenLayers.Protocol.HTTP({
14              url: "lieux_stages.php",
15              method: 'GET',
16              params: {id_promo: Ext.getCmp('promosCombo').getValue()},
17              format: new OpenLayers.Format.GeoJSON()
18          })
19      })
20  });
```

Lien gridPanel ExtJS / couche vecteur OpenLayers

- L'objet grid.FeatureSelectionModel réalise le lien entre les lignes d'un gridPanel ExtJS et les objets d'une couche vecteur OpenLayers
- L'objet est affecté à la propriété sm d'un gridPanel de ExtJS
- Quand un élément est sélectionné d'un côté, il est automatiquement sélectionné de l'autre

Exemple de modèle de sélection GeoExt

```
1  var gridPanel = new Ext.grid.GridPanel({
2      id : 'tableauAttrib',
3      border: false,
4      stripeRows: true,
5      store : lieuxStagesStore,
6      colModel: lieuxStagesGridModel,
7      sm: new GeoExt.grid.FeatureSelectionModel({
8          selectControl : {boxKey : "shiftKey"}
9      })
10 });
```

Merci de votre attention
Des questions ?