TP 3 : Développement de clients web cartographiques

Objectifs

Utilisation d'OpenLayers pour la réalisation d'une interface cartographique côté client.
Utilisation de ExtJS pour structurer une page et afficher des données attributaires.
Utilisation de GeoExt pour réaliser l'interaction entre OpenLayers et ExtJS.

Liens utiles pendant le TP (et après !)

Tutoriaux geotribu : *http://geotribu.net/node/19*

OpenLayers
API : *http://dev.openlayers.org/releases/OpenLayers-2.11/doc/apidocs/files/OpenLayers-js.html*
Exemples : *http://openlayers.org/dev/examples/*

ExtJS
API : *http://docs.sencha.com/ext-js/3-4/*
Exemples : *http://dev.sencha.com/deploy/ext-3.4.0/examples/*

GeoExt
API : *http://www.geoext.org/lib/index.html*
Exemples : *http://geoext.org/examples.html*

1 Découverte de la librairie cartographique OpenLayers

1.1 Création d'une interface et affichage d'un fond cartographique

Téléchargement et installation de la librairie OpenLayers

- Sur votre disque local, créez dans le répertoire de publication web (/var/www) un nouveau sous-répertoire « cartes ». C'est dans ce répertoire de travail que vous enregistrerez tous les fichiers créés dans la suite.
- Connectez vous avec un navigateur sur le site http://www.openlayers.org
- Cliquez sur le lien pour télécharger la dernière version au format .zip (au moment de la rédaction de ce document la dernière version stable est la 2.11).
- Enregistrez le fichier zip dans votre répertoire de travail.
- Décompressez le fichier zip dans ce même répertoire. Vous devriez obtenir un sous-répertoire « OpenLayers-2.11 » que vous renommerez « OpenLayers ».

Charger OpenLayers dans une page web

- Dans le répertoire de travail, créez un nouveau fichier « carte.html ».
- Ouvrez le fichier avec un éditeur de texte (par exemple avec le bloc-notes).
- Créez la structure HTML de base d'une page web :

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<title>Première carte avec OpenStreetMap</title>
</head>
<body>
</body>
</html>
```

- Dans le bloc <head>, juste après la ligne contenant la balise <title>, ajoutez l'appel à la librairie javascript OpenLayers de la façon suivante :

```
<script type="text/javascript"
src="OpenLayers/lib/OpenLayers.js"></script>
```

- La bibliothèque OpenLayers est maintenant accessible depuis votre page web.

Afficher une première carte avec le fond OpenStreetMap

- Il s'agit maintenant d'afficher une carte dans la page web. Pour cela, vous allez positionner un bloc dans le corps de la page. Entre les balises <body> et </body>, ajoutez ce qui suit :

```
<div id="map"></div>
```

- Dans l'en-tête de la page, vous allez définir un lien vers une feuille de style :

```
<link rel="stylesheet" href="style.css" type="text/css">
```

- Dans le répertoire de travail, créez un fichier « style.css » et définissez y le style associé à la div qui va contenir la carte :

```
#map {
```

```
    width: 800px;
    height: 620px;
    border: 1px solid #000;
}
```

- Le principe d'OpenLayers est d'écrire une fonction javascript qui affichera la carte dans la . Dans l'en-tête de la page « carte.html », juste après l'appel à la bibliothèque OpenLayers, écrivez ce qui suit :

```
<script type="text/javascript" src="carte.js"></script>
```

- Créez avec l'éditeur de texte un nouveau fichier nommé « carte.js » dans le répertoire de travail. Ajoutez ce qui suit dans le nouveau fichier :

```
function init() {
}
```

- Par la suite, tout le code de votre interface cartographique prendra place à l'intérieur de cette fonction.
- Reste maintenant à appeler cette fonction au moment du chargement de la page. Dans le fichier « carte.html », modifiez la balise <body> de la manière suivante :

```
<body onload="init()">
```

- Vous pouvez maintenant afficher un fond cartographique sur votre page. Pour commencer, vous allez intégrer la carte provenant du projet OpenStreetMap.

- Dans le fichier « carte.js », à l'intérieur de la fonction init(), créez une variable contenant une nouvelle carte OpenLayers :

```
var map = new OpenLayers.Map('map');
```

- Notez comme le constructeur de la classe Map attend en paramètre l'id de la dans laquelle doit s'afficher la carte (ici 'map').
- Vous allez maintenant créez une nouvelle couche contenant le fond cartographique libre d'OpenStreetMap :

```
var osmLayer = new OpenLayers.Layer.OSM();
```

- Ajoutez la nouvelle couche à votre carte :

```
map.addLayer(osmLayer);
```

- Pour finir, il vous faut préciser le niveau de zoom pour l'affichage de la carte. Ici vous allez zoomer sur l'étendue complète de la carte, c'est à dire celle de l'unique couche OpenStreetMap affichée (le monde entier) :

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

- Ouvrez la page « carte.html » avec votre navigateur (dans la barre d'adresse tapez « http://localhost/cartes/carte.html »). Félicitations vous venez de réaliser en quelques lignes votre première carte grâce à OpenLayers et OpenStreetMap !
- Testez les différents contrôles par défaut ajoutés automatiquement sur la carte par

OpenLayers : boutons de zoom et de déplacements, zoom à la molette de la souris, déplacement par cliquer/glisser.

Ajouter des contrôles à l'interface cartographique

- Vous allez maintenant choisir vous-même les contrôles affichés sur votre carte. Commencez par désactiver les contrôles par défaut. Pour cela, passez en 2ème paramètre de la classe Map un tableau javascript vide pour les contrôles de la carte :

```
var map = new OpenLayers.Map('map', {controls:[]});
```

- Puis rétablissez les contrôles de la souris :

```
map.addControl(new OpenLayers.Control.MouseDefaults());
```

- Ajoutez une grande barre de zoom :

```
map.addControl(new OpenLayers.Control.PanZoomBar());
```

- Ajoutez une carte d'aperçu général :

```
map.addControl(new OpenLayers.Control.OverviewMap());
```

- Ajoutez la position du curseur de la souris :

```
map.addControl(new OpenLayers.Control.MousePosition());
```

- Ajoutez un sélecteur de couche :

```
map.addControl(new OpenLayers.Control.LayerSwitcher());
```

- Recharger le fichier « carte.html » dans le navigateur. Testez les nouveaux contrôles qui sont présents sur la carte. Remarquez notamment le contrôle MousePosition en bas à droite qui affiche les coordonnées du pointeur de la souris en UTM.

Gérer le positionnement et le zoom initiaux de la carte

- Vous allez définir le positionnement du centre de la carte et son zoom par défaut qui s'afficheront au chargement de la page pour que la zone affichée corresponde à la France métropolitaine. Par défaut avec la couche OpenStreetMap, le système de projection de la carte est un Mercator sphérique qui a défini par Google (EPSG:900913). Les coordonnées sont en UTM. Pour l'affichage de la France, les coordonnées à prendre pour le centre sont 542966 pour la longitude et 5745320 pour la latitude, et le niveau de zoom vaut 6. Ce qui donne :

```
map.setCenter(new OpenLayers.LonLat(310640, 5958419), 6);
```

- Attention ! Cette ligne doit impérativement être placée après l'ajout de la couche osmLayer à la carte. Pensez également à supprimer la ligne zoomToMaxExtent(). Rechargez la page dans votre navigateur et constatez que la zone affichée est bien la France métropolitaine.

Afficher les coordonnées du pointeur de la souris en degrés décimaux

- Les coordonnées du pointeur de la souris sont affichés avec le système de projection UTM de la carte. Pour faciliter la lisibilité vous pouvez basculer l'affichage dans le système de projection WGS84 en degrés décimaux (EPSG:4326). Commencez par ajouter le nouveau système de projection dans votre carte en ajoutant la ligne suivante au début de la fonction init() :

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

- La variable epsg4326 qui contient le système de projection pourra maintenant être utilisée dans toute le code javascript. Utilisez la pour basculer l'affichage des coordonnées du pointeur en modifiant la ligne d'ajout du contrôle MousePosition de la façon suivante :

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

Gérer le positionnement de la carte en utilisant les degrés décimaux

- Pour des raisons pratiques, on peut souhaiter exprimer les coordonnées de positionnement de la carte en degrés décimaux. Pour cela, vous pouvez définir un point en degré décimaux et convertir ses coordonnées vers le système de projection de la carte. La fonction transform permet de réaliser cela de la façon suivante :

```
var center = new OpenLayers.LonLat(2.70,47.10).transform(epsg4326,
map.getProjectionObject());
```

- On a utilisé la variable epsg4326 définie précédemment pour préciser le système de projection des coordonnées fournies. Le système de projection de la carte vers lequel on souhaite transformer les coordonnées est donné par la fonction getProjectionObject() de l'objet map. On aurait aussi pu utiliser à la place une variable définie avec new OpenLayers.Projection("EPSG900913") ;
- On peut ensuite utiliser le point créé pour définir le centre de la carte :

```
map.setCenter(center, 6);
```

- Pensez à supprimer la ligne map.setCenter qui contenait les coordonnées du centre en UTM. Rechargez la page dans votre navigateur et constatez qu'elle est toujours positionnée sur la même zone.

Ajout d'un deuxième fond cartographique

- Vous allez ajouter la possibilité de choisir un deuxième fond cartographique dans votre carte. Pour pouvoir accéder aux couches Google, il est nécessaire d'ajouter l'inclusion de l'API Google dans l'en-tête du fichier « carte.html ». Ajoutez ce qui suit avant l'inclusion de la librairie OpenLayers :

```
<script src="http://maps.google.com/maps/api/js?
v=3.5&amp;sensor=false"></script>
```

- Pour intégrer le fond Google Satellite sur la carte, ajoutez ce qui suit dans la suite de « carte.js » :

```
var gsat = new OpenLayers.Layer.Google(
    "Google Satellite",
    {type: google.maps.MapTypeId.SATELLITE, numZoomLevels: 22,
sphericalMercator: true}
);
```

- Puis chargez le fond dans la carte avec :

```
map.addLayer(gsat);
```

- Utilisez le contrôle LayerSwitcher en haut à droite pour tester l'affichage des deux fonds cartographiques.

1.2 Superposition de couches géographiques locales

Pour cette partie, vous allez utiliser 4 fichiers fournis qui doivent être copiés dans le répertoire de travail : markerlist.txt, zones.kml, marker_blue.png et marker_red.png.

Ajout d'une couche de points issus d'un fichier texte

- Dans un premier temps, vous allez superposer sur votre carte une couche de points contenus dans le fichier texte markerlist.txt. Ouvrez ce fichier avec un éditeur pour en analyser le contenu.
- Il contient en première ligne l'en-tête qui défini l'ordre des colonnes : respectivement la longitude, la latitude, le titre, la description et l'icône du marqueur. Il est important de respecter le nom des champs en anglais sinon cela ne fonctionnera pas correctement. Les colonnes sont séparées par des tabulations et non des espaces. Chacune des lignes suivantes correspond à un point qui sera affiché sur la carte. La dernière colonne « icon » précise l'image à utiliser pour chacun des points.
- Il vous reste à ajouter la couche de points sur votre carte. Juste après le chargement des couches OpenStreetMap et Google, ajoutez ce qui suit :

```
var parkings = new OpenLayers.Layer.Text( "Parkings",
{location:"markerlist.txt", projection: epsg4326});
map.addLayer(parkings) ;
```

- Il est nécessaire de préciser la projection dans laquelle sont données les coordonnées latitude et longitude des points (ici EPSG:4326). Les points s'affichent sur la carte avec les icônes indiqués dans le fichier texte. Sur la carte, on peut cliquer sur chaque point pour afficher son titre et sa description.
- Constatez que la nouvelle couche s'affiche en haut à droite dans le contrôle LayerSwitcher avec le nom « Parkings » que l'on a défini. Vous pouvez la masquer en cliquant sur la case à cocher correspondante.

Ajout d'une couche de polygones issus d'un fichier KML

- Le format KML est un format mis au point par Google pour Google Maps et Google Earth qui est basé sur le formalisme XML. Il est ouvert et normalisé (voir *http://fr.wikipedia.org/wiki/KML*). Un grand nombre d'applications libres ou propriétaires sont capables de lire ou d'écrire dans ce format. Il est donc très intéressant de pouvoir l'utiliser directement avec OpenLayers.
- Vous allez afficher une nouvelle couche qui contiendra des polygones provenant d'un fichier KML « regions.kml ». Vous pouvez ouvrir le fichier avec un éditeur de texte pour analyser son contenu. Il contient deux polygones qui représentent de façon grossière les régions Bretagne et Languedoc-Roussillon.
- Il faut maintenant ajouter une nouvelle couche à la carte pour y intégrer ce fichier KML. Cette couche sera ajoutée juste avant le chargement de la couche contenant les points. C'est toujours la dernière couche chargée qui est affichée au-dessus. Cela permettra donc que les marqueurs soient affichés au-dessus des polygones. Dans le cas contraire, les marqueurs ne seraient plus cliquables. Saisissez le code suivant avant celui des parkings :

```
var regions = new OpenLayers.Layer.GML("Régions", "regions.kml", {
    format: OpenLayers.Format.KML,
    projection : epsg4326,
    styleMap: new OpenLayers.StyleMap({
    "default": {
        fillColor: "#0000AA",
        strokeColor: "#000000",
        strokeWidth: 2,
        fillOpacity: 0.4
       }
    })
});
map.addLayer(regions);
```

- Grâce à la propriété styleMap et un objet OpenLayers.StyleMap , vous avez également défini l'apparence des polygones chargés depuis le fichier KML : couleur de remplissage, couleur du contour, épaisseur du contour et opacité (0.4 pour une opacité de 40%, c'est à dire une transparence de 60%) .
- La nouvelle couche nommée Régions apparaît sur la carte. Comme pour les marqueurs, on peut la masquer temporairement grâce au contrôle LayerSwitcher.

Téléchargement et affichage de points issus d'OpenStreetMap

- Pour commencer, vous allez télécharger les points qui vous intéressent depuis OpenStreetMap en lançant une requête grâce son API étendue (XAPI). Ouvrez un nouvel onglet dans votre navigateur et saisissez l'adresse suivante :

```
http://open.mapquestapi.com/xapi/api/0.6/node[amenity=recycling]
[bbox=4.86465,45.77530,4.89021,45.78700]
```

- Ici vous demandez tous les points (node) qui sont des containers de recyclage (amenity=recycling) et contenus dans un rectangle dont les coordonnées sont précisées (la bbox est toujours obligatoire et ne doit pas être trop grande).
- Sauver le fichier « data.osm » dans le répertoire de travail. Renommez le « recyclage.osm ». Le format OSM est le format natif des données d'OpenStreetMap.
- Les fichiers OSM peuvent être lus par OpenLayers en utilisant le format « OpenLayers.Format.OSM ». Il suffit donc de charger le fichier « recyclage.osm » dans une nouvelle couche en utilisant ce format :

```
var recyclage = new OpenLayers.Layer.GML("Points de recyclage",
"recyclage.osm", {
   format: OpenLayers.Format.OSM,
   projection: epsg4326,
   styleMap:new OpenLayers.StyleMap({
      "default": {
     pointRadius: 7,
     fillColor: "#00DD00",
     strokeColor: "#000000",
     fillOpacity: 1
      }
   })
});
map.addLayer(recyclage);
```

- L'apparence des points a été définie sous forme de cercles de rayon 7 pixels, avec une couleur de fond verte à 100% opaque et un pourtour noir. Dans l'interface, zoomez sur la zone concernée (au nord de Lyon) pour visualiser les différents points.

2 Réalisation d'une interface web complète avec OpenLayers, ExtJS et GeoExt

Vous allez maintenant réaliser une application web complète qui permettra d'afficher des informations attributaires et géographiques concernant les stages du Master.

Téléchargement et installation des librairies

- Dans le répertoire de publication web /var/www, créez un nouveau répertoire « appli_stages ». Créez y un sous-répertoire « lib ». Dans ce dernier copiez le répertoire « OpenLayers » que vous aviez ajouté dans la première partie.
- Pour télécharger la librairie ExtJS, connectez vous sur la page http://www.sencha.com/products/extjs3/download/ puis cliquez sur le lien « Download for Ext JS 3.4.0 ». Décompressez le zip dans le répertoire /var/www/appli_stages/lib.
- Pour télécharger la librairie GeoExt, connectez vous sur la page http://geoext.org/downloads.html puis cliquez sur le lien « GeoExt 1.0 : Source ». Décompressez le zip dans le répertoire /var/www/appli_stages/lib.
- Dans le répertoire /var/www/appli_stages, créez un fichier « stages.html » et un fichier « stages.js ». Dans le fichier stages.html, créez la structure de base d'une page HTML :

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<title>Cartographie des stages</title>
</head>
<body onload="init()">
    <div id="map"></div>
</body>
</html>
```

- Dans le bloc <head>, juste après la ligne contenant la balise <title>, ajoutez les appels aux feuilles de styles de ExtJS et GeoExt :

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

- Ajoutez à la suite les appels aux librairies javascript de la façon suivante (ExtJS et OpenLayers doivent toujours être chargées **avant** GeoExt) :

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

```
type="text/javascript"></script>
<script type="text/javascript" src="stages.js"></script>
```

- Dans le fichier « stages.js », déclarez la fonction init() dans laquelle prendra place tout le code de votre interface :

```
function init() {

}
```

Mise en place de la structure générale de la page avec ExtJS

- Vous allez structurer l'interface en 3 blocs : un bloc à gauche (westPanel) contiendra un formulaire avec une liste déroulante pour sélectionner les promotions du Master, un bloc central (mapPanel) dans lequel s'affichera la carte des lieux de stages de la promotion sélectionnée et un bloc en bas (southPanel) dans lequel s'affichera les informations attributaires de ces stages. Dans la fonction init(), ajoutez le code suivant :

```
new Ext.Viewport({
    layout: "border",
    defaults: {
        split: true
    },
    items: [
       mapPanel,
       westPanel,
       southPanel
    ]
});
```

- Il faudra remplir dans la suite les trois panels que vous venez de déclarer. Attention, tous les codes de « stages.js » qui suivent doivent se situer **avant** cet objet Viewport.

Création du formulaire de sélection des promotions du Master (westPanel)

- Les différentes promotions du Master doivent être chargées depuis la table « etudiant » dans la base de données des stages. Le chargement des données sera réalisé grâce à un script PHP et le résultat renvoyé dans une liste déroulante ExtJS par l'intermédiaire d'un appel AJAX.
- Créez un fichier « liste_promos.php » dans le répertoire « appli_stages » et ajoutez y le code PHP suivant :

```
<?php
$dbconn = pg_connect("host=localhost dbname=TPMapserver user=tp
password=tpbds");
$requete = "SELECT promotion||' ('||COUNT(numetud)||' étudiants)' AS
display, promotion AS value FROM etudiant GROUP BY promotion ORDER BY
promotion DESC;";
$result = pg_query($dbconn,$requete);
while ($row = pg_fetch_object($result)) {
    $json_rows[] = $row;
}
$header = '{success: true, rows: ';
```

```
$footer = '}';
echo $header . json_encode($json_rows) . $footer;
?>
```

- Dans le navigateur ouvrez http://localhost/appli_stages/liste_promos.php et analyser le contenu JSON renvoyé par le script PHP. Dans le script ci-dessus on a enlevé les différents tests d'erreurs pour simplifier le code. Dans le cadre d'un projet en production il faudrait les ajouter (erreur de connexion à la base, erreur de requête, ...)
- Dans le fichier « stages.js », vous allez maintenant créer un formulaire avec une liste déroulante dont le contenu est chargé par le script PHP précédent. Commencez par créer un magasin de données JSON (JsonStore) qui charge les données issues du script PHP :

```
var promosStore = new Ext.data.JsonStore({
    url : 'liste_promos.php',
    fields : ['display', 'value'],
    root : 'rows',
    autoLoad : true
});
```

- Créez maintenant la liste déroulante qui utilise le magasin précédent comme source pour son contenu :

```
var promosCombo = new Ext.form.ComboBox({
    id : 'promosCombo',
    fieldLabel : "Promotion",
    triggerAction : 'all',
    emptyText : "Choisir une promotion",
    editable : false,
    store : promosStore,
    mode : 'local',
    valueField : 'value',
    displayField : 'display'
});
```

- Créez un bouton qui servira à envoyer le formulaire :

```
var submitButton = new Ext.Button({
    text : 'Afficher sur la carte'
});
```

- Créez le formulaire qui contient la liste et le bouton :

```
var promosSelectionForm = new Ext.FormPanel({
    id : 'promosSelection',
    title : "Liste des promotions",
    frame : true,
    width : '100%',
    buttonAlign : 'center',
    labelAlign : 'left',
    labelWidth : 70,
    items : [promosCombo],
    buttons : [submitButton]
});
```

- Pour finir, créez le panel westPanel qui va contenir le formulaire :

```
var westPanel = new Ext.Panel({
```

```
    region  : 'west',
    border : false,
    width  : 300,
    minSize: 275,
    maxSize: 325,
    collapsible: true,
    items : [promosSelectionForm]
});
```

- Dans le bloc Viewport (qui se situe normalement à la fin de votre code), commentez les 2 lignes qui concernent les deux panels que vous n'avez pas encore créés (mapPanel et southPanel), puis dans votre navigateur ouvrez la page http://localhost/appli_stages/stages.html. Vérifiez que la liste contient bien les promotions chargées depuis la base et que le nombre d'étudiants correspondant est correct.

Création de la carte contenant les lieux des stages (mapPanel)

- A la suite du bloc précédent, vous allez maintenant ajouter la carte OpenLayers avec la localisation des stages dans le mapPanel. Créez la structure de base de la carte :

```
var epsg4326 = new OpenLayers.Projection("EPSG:4326");
var options = {
    controls: [],
    projection: epsg4326
};
var map = new OpenLayers.Map('', options);
map.addControl(new OpenLayers.Control.LayerSwitcher());
map.addControl(new OpenLayers.Control.PanZoomBar());
map.addControl(new OpenLayers.Control.Navigation());
var mapBounds = new OpenLayers.Bounds(-5,43,8,50);
```

- Ajoutez comme fond cartographique les limites des départements français issus de votre mapserver local (la couche FRA_adm2) :

```
var contourPays = new OpenLayers.Layer.WMS(
    "France - Limites admin",
    "http://localhost/cgi-bin/mapserv?
     map=/var/www/data/TP2/cartoEtudiants.map&",
    {
     projection: epsg4326,
     layers: "FRA_adm2",
     transparent: true,
     format: 'image/png'
    },
    {
     singleTile: true,
     isBaseLayer : true
    }
);
map.addLayer(contourPays);
```

- Ajoutez une couche vecteur superposée qui va contenir la couche des lieux de stages (points) issus de la base de données. Pour le moment, vous vous contenterez d'afficher les stages de la promotion 2010 sans aucun lien avec la liste déroulante :

```
var lieuxStages = new OpenLayers.Layer.Vector(
    "Lieux des stages",
    {
        protocol: new OpenLayers.Protocol.HTTP({
            url: "lieux_stages.php?id_promo=2010",
            format: new OpenLayers.Format.GeoJSON()
        }),
        projection: epsg4326,
        strategies: [new OpenLayers.Strategy.Fixed()]
    }
);
```

- Activez la possibilité de sélectionner les éléments de la couche vecteur avec un contrôle selectFeature puis ajoutez la couche à la carte :

```
var selectFeatureControl = new
OpenLayers.Control.SelectFeature(lieuxStages);
map.addControl(selectFeatureControl);
selectFeatureControl.activate();
map.addLayer(lieuxStages);
```

- Vous devez maintenant créez le script PHP qui charge les points depuis la base de données en fonction de la promotion et les renvoie au format GeoJSON. Dans un nouveau fichier « lieux_stages.php », saisissez le code suivant :

```
<?php
$dbconn = pg_connect("host=localhost dbname=TPMapserver user=tp
password=tpbds");
$requete = "SELECT numstage, nometud, prenometud, sujet, datedebut,
datefin, ST_ASGeoJSON(localisation) AS lieu FROM stage, etudiant,
entreprise WHERE stage.nument=entreprise.gid AND
stage.numetud=etudiant.numetud AND promotion='" . $_GET['id_promo'] .
"' ORDER BY datefin DESC;";
$result = pg_query($dbconn,$requete);
while ($row = pg_fetch_assoc($result)) {
    $type = '"type": "Feature"';
    $geometry = '"geometry": ' . $row['lieu'];
    unset($row['lieu']);
    $properties = '"properties": ' . json_encode($row);
    $feature[] = '{' . $type . ', ' . $geometry . ', ' .
$properties . '}';
}
$header = '{"type": "FeatureCollection", "features": [';
$footer = ']}';
echo $header . implode(', ', $feature) . $footer;
?>
```

- Vous pouvez enfin créer un MapPanel GeoExt en lui affectant la carte OpenLayers. Pour l'instant vous n'aviez pas encore utilisé la variable mapBounds pour définir l'étendue par défaut de la carte. Cela peut être fait au moment de la création du MapPanel de la façon suivante :

```
var mapPanel = new GeoExt.MapPanel({
    map: map,
    region : 'center',
    height: 600,
    width: 800,
    title: 'Lieux des stages du Master',
```

```
    collapsible: false,
    extent: mapBounds
});
```

- Décommentez la ligne qui concerne le mapPanel dans le Viewport, puis testez l'interface avec votre navigateur. Sélectionnez des points de la couche vecteur et remarquez le changement d'apparence du point sélectionné.
- Vous allez changer l'apparence des points de la couche vecteur : les points non sélectionnés apparaîtront en vert entouré de bleu, tandis que les points sélectionnés seront un peu plus gros et en rouge entouré de noir. Ajoutez le code suivant juste **avant** l'ajout de la couche « lieuxStages » à la carte :

```
var lieuxStyleDefault = new OpenLayers.Style({
    'strokeColor': '#0000aa',
    'strokeWidth': 2,
    'fillOpacity': 0.80,
    'fillColor': '#669933',
    'pointRadius': 5
});
var lieuxStyleSelected = new OpenLayers.Style({
    'strokeColor': '#000000',
    'strokeWidth': 2,
    'fillOpacity': 0.80,
    'fillColor': '#aa0000',
    'pointRadius': 7
});
lieuxStages.styleMap = new OpenLayers.StyleMap({
    'default': lieuxStyleDefault,
    'select': lieuxStyleSelected
});
```

- Visualisez à nouveau l'interface et sélectionnez des points pour voir le nouveau style de sélection.

Utilisation de la liste pour charger les données de la carte et création du tableau contenant les informations attributaires des stages (southPanel)

- Pour pouvoir faire le lien entre la carte et les autres blocs de ExtJS, le chargement des données dans la couche vecteur de la carte doit maintenant être géré par GeoExt. Vous devez donc changer le bloc de création de la couche vecteur. Remplacez le code précédent par celui qui suit :

```
var lieuxStages = new OpenLayers.Layer.Vector("Lieux des stages");
```

- Pour afficher les données GeoJSON renvoyées par le script PHP, vous devez créer un gridPanel avec ExtJS. Ce gridPanel est un tableau qui s'appuie sur deux objets différents : un magasin de données (store) qui charge les données GeoJSON et un modèle de colonnes (colModel) qui définit l'affichage du tableau (titre, largeur, tri des colonnes). Le gridPanel permettra ensuite de gérer le lien entre les valeurs attributaires et la carte grâce à un modèle de sélection issu de GeoExt. Commencez par créer un magasin de données GeoExt de la façon suivante :

```
lieuxStagesStore = new GeoExt.data.FeatureStore({
    layer: lieuxStages,
    idProperty: 'numstage',
    fields: [
        {name: 'numstage', type: 'int'},
```

```
            {name: 'datedebut', type: 'date', dateFormat: 'Y-m-d'},
            {name: 'datefin', dateFormat: 'Y-m-d'},
            {name: 'sujet', type: 'string'},
            {name: 'nometud', type: 'string'},
            {name: 'prenometud', type: 'string'}
        ],
        proxy: new GeoExt.data.ProtocolProxy({
            protocol: new OpenLayers.Protocol.HTTP({
                url: "lieux_stages.php",
                method: 'GET',
                params: {id_promo: Ext.getCmp('promosCombo').getValue()},
                format: new OpenLayers.Format.GeoJSON()
            })
        })
    });
```

- Dans le code précédent, vous avez défini les différents champs à récupérer depuis les données GeoJSON grâce à la propriété « fields ». Avec la propriété « layer », vous avez également indiqué à GeoExt la couche OpenLayers concernée sur la carte. Grâce à la propriété « proxy », vous précisez le script PHP qui est la source des données et vous lui ajoutez le paramètre « id_promo » en précisant que sa valeur provient de l'objet ExtJS « promosCombo », c'est à dire la liste déroulante du westPanel.
- Vous devez maintenant créez le modèle de colonnes du gridPanel. Ici pas besoin de GeoExt, un ColumnModel standard de ExtJS fera l'affaire :

```
var lieuxStagesGridModel = new Ext.grid.ColumnModel({
    columns: [
        new Ext.grid.RowNumberer(),
        {
            id : "id",
            header : "Id",
            width : 25,
            dataIndex : "numstage",
            hidden: true
        }, {
            id : "datedebut",
            header : "Date de début",
            width : 120,
            dataIndex : "datedebut",
            sortable: true,
            renderer: Ext.util.Format.dateRenderer('l d/m/Y')
        }, {
            id : "datefin",
            header : "Date de fin",
            width : 120,
            dataIndex : "datefin",
            sortable: true,
            renderer: Ext.util.Format.dateRenderer('l d/m/Y')
        }, {
            id : "nometud",
            header : "Nom de l'étudiant",
            width : 110,
            dataIndex : "nometud",
            sortable: true
        }, {
            id : "prenometud",
```

```
            header : "Prénom de l'étudiant",
            width : 110,
            dataIndex : "prenometud",
            sortable: true
        }, {
            id : "sujet",
            header : "Sujet du stage",
            width : 700,
            dataIndex : "sujet",
            sortable: true
        }
    ]
});
```

- Le modèle de colonnes consiste simplement à déclarer chacune des colonnes qui vont apparaître dans le gridPanel. La propriété « header » est le texte qui apparaîtra dans l'en-tête des colonnes. La propriété dataIndex fait le lien avec le « name » déclaré dans le magasin de données. D'autres options peuvent être définies (ici vous avez géré la largeur et la possibilité de trier sur les colonnes). La première colonne Ext.grid.RowNumberer() permet d'afficher un compteur de lignes.
- Créez le gridPanel à partir du magasin et du modèle de colonnes en prenant soin de préciser un FeatureSelectionModel de GeoExt comme modèle pour la sélection des enregistrements (propriété « sm ») :

```
var gridPanel = new Ext.grid.GridPanel({
    id : 'tableauAttrib',
    border: false,
    stripeRows: true,
    store : lieuxStagesStore,
    colModel: lieuxStagesGridModel,
    sm: new GeoExt.grid.FeatureSelectionModel({
        selectControl : {boxKey : "shiftKey"}
    })
});
```

- Positionnez votre gridPanel dans le southPanel que vous aviez prévu au départ :

```
var southPanel = new Ext.Panel({
    title   : 'Liste des stages',
    region : 'south',
    layout : 'fit',
    collapsible: true,
    width  : '100%',
    height : 200,
    items  : [gridPanel]
});
```

- Pensez à décommenter la ligne concernant le southPanel dans le Viewport, puis afficher l'interface dans votre navigateur. Vous verrez apparaître un nouveau panel en bas de votre interface. Par contre, si vous sélectionnez une promotion dans la liste déroulante, vous constaterez que rien ne se passe ni sur la carte, ni dans le gridPanel. C'est normal car vous n'avez rien défini au moment du clic sur le bouton « Afficher sur la carte » du formulaire. Reprenez la définition du « submitButton » et transformez là en :

```
var submitButton = new Ext.Button({
```

```
    text : 'Afficher sur la carte',
    handler : function() {
        var datasetId = Ext.getCmp('promosCombo').getValue();
        if (datasetId != '') {
            Ext.getCmp('tableauAttrib').store.removeAll();
            Ext.getCmp('tableauAttrib').store.proxy.protocol.params.i
d_promo = datasetId;
            Ext.getCmp('tableauAttrib').store.load();
        }
        else {
            Ext.MessageBox.alert("Erreur !","Veuillez d'abord
sélectionner une promotion à afficher.");
        }
    }
});
```

- La propriété « handler » gère l'exécution d'une fonction au moment de l'activation (un click pour un bouton). Dans la fonction exécutée, vous récupérez la valeur de la promotion sélectionnée dans la liste « promosCombo ». Si rien n'a été sélectionné, vous affichez un message d'erreur dans une boîte d'information. Si une promotion a été sélectionnée, vous videz le magasin de données du gridPanel « tableauAttrib », puis vous le rechargez en utilisant la valeur sélectionnée.
- Ouvrez à nouveau l'interface dans votre navigateur et testez que le chargement des données fonctionne correctement sur la carte et dans le panel du bas. Remarquez le mode de sélection bi-directionnel entre la couche vecteur sur la carte et les lignes dans le gridPanel. Testez également l'ordre de tri (petite flèche à droite de l'en-tête de chaque colonne) et le redimensionnement des colonnes.

Félicitations vous venez de terminer votre première interface web cartographique avec OpenLayers, ExtJS et GeoExt !