

**M2 pro Sigma : Module 653  
Cartographie et SIG interactifs en ligne**

**SÉANCE 6 : LE WEBMAPPING CÔTÉ CLIENT  
L'API OPENLAYERS**

**Webographie :**

- Site officiel, documentation (attention, deux parties) et exemples (très utiles pour essayer une fonctionnalité et la personnaliser par la suite) : <http://www.openlayers.org/>
- Tutoriaux GéoTribu : <http://www.geotribu.net/node/21>
- Framework GéoEXT : <http://www.geoext.org/>

**Introduction**

OpenLayers est une bibliothèque de fonctions JavaScript développée au départ par la société MetaCarta pour ses besoins propres, puis versée dans le domaine public et retenue comme projet open source par l'OSGEO.

Cette bibliothèque permet, côté client donc, d'afficher des données géospatiales, de les explorer (zoom, pan, interrogations), et d'offrir à l'utilisateur le moyen d'ajouter ses informations au clic. Pour le programmeur, ces données géospatiales deviennent des objets JS manipulables et interactifs (reprojection, styles d'affichage, réaction au clic et même dessin). OpenLayers offre donc un ensemble de fonctions utiles pour visualiser des données spatiales et permettre des interactions avancées dans le cadre d'une interface graphique complète.

OpenLayers est capable de se connecter à un grand nombre de sources de données spatiales, notamment les webservices aux normes OGC, mais aussi de lire des fichiers KML et des fichiers images calés. OpenLayers a été sélectionné par l'IGN pour son API d'accès au GéoPortail, qui en est une extension.

Exemples avancés d'utilisation (avec la participation d'étudiants de Sigma) :

- Carte interactive du laboratoire Terrae : <http://w3.terrae.univ-tlse2.fr/carte/carte.php>
- Atelier cartographique d'Obediences.net : [http://www.obediencies.net/MapFish-1.1/client/examples/php/fichier\\_lieux.php](http://www.obediencies.net/MapFish-1.1/client/examples/php/fichier_lieux.php)

**1°- Principes de l'utilisation d'OpenLayers et première carte interactive**

On peut tester l'utilisation d'OpenLayers en local, mais il faut bien sûr que les fichiers soient servis en http (par Apache par exemple). Une fois téléchargée, la bibliothèque est disponible sous deux formes : un seul fichier dont le code JS est compacté, ou un fichier appelant de nombreux sous-fichiers, dont le code JS est lisible, décompacté, ce qui est intéressant pour le débogage (avec le plugin FireBug de FireFox par exemple).

On appelle l'un ou l'autre de ces fichiers selon que l'on est en mode de test (débogage facilité) ou en mode de production (rapidité apportée par la version « compressée »). Comme il s'agit d'une bibliothèque de fonctions JavaScript, pour l'utiliser il faut ajouter un lien dans une balise <script> du bloc <head> de votre fichier HTML.

Par exemple, si on place le répertoire OpenLayers dans un sous-répertoire nommé « OpenLayers » du répertoire contenant le fichier HTML l'appel sera de la forme suivante :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Exemple d'utilisation d'OpenLayers</title>
    <script src="./OpenLayers/OpenLayers.js" type="text/javascript">
    </script>
  </head>
  ...
```

Une fois la bibliothèque chargée, ses fonctions deviennent disponibles à la programmation en JavaScript. Par exemple, commençons par faire afficher une carte OpenLayers dans un <div> de notre page HTML, carte qui sera issue des données WMS GéoSignal, la couche raster 1000K.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Exemple d'utilisation d'OpenLayers</title>
    <script src="/OpenLayers-2.7/OpenLayers.js" type="text/javascript"></script>
    <script type="text/javascript">
      function init() {
        var map, layer_wms;
        map = new OpenLayers.Map('map');
        var layer_wms = new OpenLayers.Layer.WMS(
          "GeoSignal WMS",
          "http://www.geosignal.org/cgi-bin/wmsmap?",
          {layers: 'RASTER1000k'}
        );
        map.addLayers([layer_wms]);

        //Centrage et zoom
        var posgps = new OpenLayers.LonLat(1, 44); //Position lat/long en
        coordonnées classiques
        var posg = posgps.transform(
          new OpenLayers.Projection("EPSG:4326"),
          new OpenLayers.Projection("EPSG:27582")); //Conversion en
        coordonnées L2C
        map.setCenter(posg, 8);
      }
    </script>
  </head>
  <body onload="init()" >
    <h2>Exemple d'utilisation d'OpenLayers pour consommer un WMS (GéoSignal
    Raster 1000K)</h2>
    <div id="map" style="width: 512px; height: 256px; border: 1px solid
    #bbb;"></div>
  </body>
</html>
```

Cet exemple est visible en ligne : <http://www.geotests.net/apis/ol/index.html>

Explication :

- Lors du chargement de la page, la fonction init() est appelée.
- Cette fonction contient les commandes OL pour créer la carte, c'est à dire :
  - initialiser les objets map et layer\_wms
  - créer un objet carte OL dans le div dont l'id est "map"
  - créer un layer de type WMS en allant consommer le WMS Geosignal
  - ajouter ce layer à l'objet map
  - créer un objet point de coordonnées correspondant à la Haute-Garonne, en lat/long
  - convertir ce point en coordonnées L2C, qui est le système utilisé par le WMS Géosignal
  - centrer la carte sur ce point et zoomer dessus au niveau 8

Exercice : Modifiez cet exemple pour afficher le fond OpenStreetMaps sur Toulouse. Il faut utiliser un layer de type OSM, et convertir les coordonnées de Toulouse dans le système Google Mercator.

Cf. : <http://www.geotests.net/apis/ol/indexosm.html>

## 2°- Ajout de contrôles utilisateur et multiples couches WMS

Pour aller un peu plus loin et ajouter un peu d'interactivité, nous allons utiliser les fonctionnalités offertes par OpenLayers. Un grand nombre de fonctionnalités d'interaction utilisateur sont en effet disponibles en utilisant des objets OpenLayers simples.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
  a <title>Exemple d'utilisation d'OpenLayers : 2</title>
  <script src="/OpenLayers-2.8-rc1/OpenLayers.js"
type="text/javascript"></script>
  <script type="text/javascript">
    function init() {
      var map;
      map = new OpenLayers.Map('map', {
        controls: [
          new OpenLayers.Control.Navigation(),
          new OpenLayers.Control.PanZoomBar(),
          new OpenLayers.Control.LayerSwitcher({'ascending':false}),
          new OpenLayers.Control.Permalink(),
          new OpenLayers.Control.ScaleLine(),
          new OpenLayers.Control.Permalink('permalink'),
          new OpenLayers.Control.MousePosition(),
        ],
        maxExtent: new OpenLayers.Bounds(-1, 40, 1, 45),
        allOverlays: true
      });

      var osm_europe = new OpenLayers.Layer.WMS('Europe',
        'http://osm.wherogroup.com/cgi-bin/osm_basic.xml?',
        {layers: 'Grenzen', transparent: 'true', isBaseLayer: 'true',
attribution:'<span style="background-color:#fff">data by <a
href="http://openstreetmap.org">openstreetmap</a>'}
```

```

    );

    var osm_autoroute = new OpenLayers.Layer.WMS('Autoroute',
        'http://osm.wherogroup.com/cgi-bin/osm_basic.xml?',
        {layers: 'Autobahn', transparent: 'true'}
    );

    var osm_villes = new OpenLayers.Layer.WMS('Ville',
        'http://osm.wherogroup.com/cgi-bin/osm_basic.xml?',
        {layers: 'Grossstadt', transparent: 'true'}
    );

    var osm_hydro = new OpenLayers.Layer.WMS('Hydro',
        'http://osm.wherogroup.com/cgi-bin/osm_basic.xml?',
        {layers: 'Fluesse', transparent: 'true'}
    );

    var vectors = new OpenLayers.Layer.Vector("Vector Layer");

    map.addControl(new OpenLayers.Control.EditingToolbar(vectors));
    map.addLayers([osm_europe, osm_autoroute, osm_hydro, osm_villes]);

    //Centrage et zoom
    var posgps = new OpenLayers.LonLat(1, 44); //Position lat/long en
    coordonnées classiques
    map.setCenter(posgps, 8);
}
</script>
</head>
<body onload="init()" >
    <h2>Exemple 2 : Layers multiples et contrôles utilisateur</h2>
    <div id="map" style="width: 512px; height: 400px; border: 1px solid
#bbb;"></div>
</body>
</html>

```

Fichier visible en ligne ici : <http://www.geotests.net/apis/ol/index2.html>

On utilise la dernière version disponible (2.10) d'OL car elle permet de mieux gérer les couches WMS de grande taille. De même, pour réduire la charge serveur on définit l'option MaxExtent de l'objet map pour réduire l'extension des requêtes WMS qui sont lancées.

Le serveur utilisé est celui de la société allemande WhereGroup, qui diffuse en WMS les données OpenStreetMap. Nous allons consommer quelques layers de ce serveur : Autoroutes, Villes, Rivières principales et le fond des frontières européennes. L'ensemble des couches disponibles est visible dans une application web basée sur MapBender : <http://osm.wherogroup.com/basic>

Les contrôles utilisateur OL présents sur cette page sont :

- Navigation (gestion de la souris : drag et clics)
- PanZoomBar (flèches de panning et curseur de zoom)
- LayerSwitcher (Liste des couches et leur visibilité)
- Permalink (lien pour retrouver une carte identique)
- ScaleLine (barre d'échelle)
- MousePosition (affichage des coordonnées de la position de la souris)
- EditingToolbar (Barre d'outils d'édition et de dessin)

Ces couches étant non projetées (lat/long WGS84, EPSG:4326), nous n'avons pas besoin de reprojeter notre point de centrage+zoom.

### 3°- Ajout de marqueurs et d'objets vectoriels

Enfin, nous allons voir comment ajouter un marqueur (puce) cliquable sur une carte, statiquement (dans le code). Un exemple de génération dynamique de marqueurs cliquable est visible dans le code source de la carte interactive du laboratoire Terrae (lien cité).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script src="/OpenLayers-2.8-rc1/OpenLayers.js"
type="text/javascript"></script>
    <script type="text/javascript">
      //Inspiré d'un exemple de GéoTribu
      function init() {
        var map, popup;
        map = new OpenLayers.Map('map');

        var layer_wms = new OpenLayers.Layer.WMS( "OpenLayers
WMS", "http://labs.metacarta.com/wms/vmap0?", {layers: 'basic'});
        map.addLayer(layer_wms);
        map.setCenter(new OpenLayers.LonLat(1, 41), 2);
        map.addControl(new OpenLayers.Control.LayerSwitcher());

        markers = new OpenLayers.Layer.Markers("Marker");
        map.addLayer(markers);

        //Creation du Marker
        feature = new OpenLayers.Feature(layer_wms, new OpenLayers.LonLat(0,45));
        marker = feature.createMarker();
        markers.addMarker(marker);
        marker.events.register("mousedown", marker, mousedown);

        //Gestion des evenements
        function mousedown(evt) {
          //Je verifie qu'aucun popup n'existe deja
          if (popup == null) {
            popup = feature.createPopup(true);
            popup.setContentHTML("Exemple de PopUp sur un marqueur");
            popup.setBackgroundColor("white");
            markers.map.addPopup(popup);
          } else {
            popup.toggle();
          }
          OpenLayers.Event.stop(evt);
        }
      }
    </script>
  </head>
  <body onload="init()" >
    <h2>Exemple de marqueur cliquable</h2>
    <div id="map" style="width: 512px; height: 256px; border: 1px solid
#bbb;"></div>
  </body>
</html>
```

Ce fichier est testable en ligne ici : <http://www.geotests.net/apis/ol/index3.html>

Description du code source :

- Une carte basée sur une seule couche WMS (MetaCarta basic layer)
- on ajoute un layer de type Marker
- On crée un objet de type *feature*, à des coordonnées fournies, sur le layer WMS

- On crée un objet de type marker, dans le layer Markers, localisé sur cet objet feature
- On lie un événement au clic sur ce marqueur, en indiquant la fonction à lancer
- On définit cette fonction de réaction au clic : elle créera un objet popup sur l'objet feature dans la couche Markers (sauf si le popup existe déjà, alors on le rend invisible s'il était visible, et inversement)

#### 4°- Création d'entités spatiales au format GeoJSON

Le format JSON correspond à la notation {paramètre : valeur, paramètre : valeur} de JavaScript, qui a l'avantage de ne pas nécessiter de conversion (« *parsing* »), puisqu'il est directement intégré par le moteur d'exécution JavaScript du navigateur, à la différence des formats plus courants comme le KML, le WKT, et les protocoles d'échanges d'entités vectorielles comme le WFS.

L'intégration d'entités spatiales au format JSON a donc été rapidement utilisée, et OpenLayers sait lire ce format, devenu GeoJSON.

Un exemple d'utilisation est disponible sur la page des exemples de développement d'OpenLayers : <http://www.openlayers.org/dev/examples/>

#### 5°- Utilisation de l'API GéoPortail

Le GéoPortail de l'IGN propose depuis avril 2008, en version bêta, l'accès à ses données par le biais d'une API JavaScript basée sur la bibliothèque de fonctions OpenLayers. C'est toujours la version 2.7 d'OL qui est utilisée actuellement, avec l'ajout de fonctions supplémentaires par surcharge.

La gestion des projections est réalisée par la version JavaScript de PROJ4, et les effets graphiques le sont par la bibliothèque script.aculo.us (concurrente d'EXT, autre bibliothèque graphique JS connue, utilisée dans MapFish et GeoEXT).

Dans cette version bêta, le fichier de l'API pèse 370ko compacté, ce qui est assez important (ce fichier doit être téléchargé par le navigateur client pour être exécuté, au premier chargement de la page).

Pour l'instant l'accès au service est gratuit, dans les limites précisées par une licence d'utilisation (qui fixe notamment un nombre maximal de requêtes par jour). Lorsque la version définitive sera disponible, l'utilisation gratuite de l'API ne permettra de visualiser que le contenu classique du GéoPortail : cartes scannées et orthophotos.

Il est aussi possible d'accéder aux données du GéoPortail par les services WMS/WMS-C et WFS, avec un système d'identification et de gestion des accès (payant).

Affichage simple d'une fenêtre GéoPortail avec les paramètres par défaut :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```

<head>
<title>API Geoportail - votre carte personnelle</title>
<script src="http://api.ign.fr/api?v=1.0beta3&key=XXXXXXX&instance=map"></script>

<script type="text/javascript">

function initGeoportalMap() {

    geoportalLoadmap("GeoportalMapDiv", "normal", "FXX");
    if(map.allowedGeoportalLayers){
        map.addGeoportalLayers(map.allowedGeoportalLayers);
    }
}
</script>
</head>
<body>
<div id="GeoportalMapDiv" style="width:400px;height:400px;"></div>
</body>
</html>

```

## Personnalisation

- Ne charger que la couche cartes :

```

function initGeoportalMap() {

    geoportalLoadmap("GeoportalMapDiv", "normal", "FXX");

    if(map.allowedGeoportalLayers){
        for (var i= 0; i<map.allowedGeoportalLayers.length; i++) {
            var overloaded_options= null;
            switch (map.allowedGeoportalLayers[i]) {
                case 'GEOGRAPHICALGRIDSYSTEMS.MAPS': // cartes
                    overloaded_options= {
                                            opacity: 1.0
                                        };
                    // On ajoute la couche
                    map.addGeoportalLayer(map.allowedGeoportalLayers[i]
                    ,overloaded_options);
                    break;
                case 'ORTHOIMAGERY.ORTHOPHOTOS'      : // ortho-photos
                    // On ne fait rien
                    break;
                default                                :
                    break;
            }
        }
    }
}

```

- Modification de l'apparence des couches

Pour modifier l'apparence des couches affichées par l'API GéoPortail, il faut rechercher dans le DOM de la page les objets qui représentent les couches, pour ensuite pouvoir modifier leurs paramètres d'affichage.

Exemple : Rendre invisible couche orthophotos  
<http://www.geotests.net/apis/geoportail/ex1.html>

```

for(var i = 1; i < map.olMap.layers.length; i++){
    var layer= map.olMap.layers[i];
    if(layer.name == 'ORTHOIMAGERY.ORTHOPHOTOS')
    {
        layer.setVisibility(false);
    }
}

```

On passe en revue les couches (tableau layers de l'objet olMap) avec une boucle for. Si le layer en cours possède un paramètre nommé « ORTHOIMAGERY.ORTHOPHOTOS », on le rend invisible.

#### - Ajout de données KML

On utilise simplement la fonction OpenLayers correspondant :

```

function initGeoportalMap() {
    geoportalLoadmap("GeoportalMapDiv", "normal", "FXX");

    if(map.allowedGeoportalLayers){
        map.addGeoportalLayers(map.allowedGeoportalLayers);
    }

    map.addLayer("KML",
        {
            'terrasse.kml.name':
            {
                'fr':"Terrasse Carto",
            }
        },
        "../google/carto.kml",
        {
            minZoomLevel:10,
            maxZoomLevel:17
        }
    );
    map.setCenterAtLonLat(1.40220, 43.577531, 18);
    map.setZoom();
}

```

Exemple : <http://www.geotests.net/apis/geoportail/ex2.html>