


ATELIER	THEMATIQUE 2 : INTRODUCTION AU DEVELOPPEMENT WEBMAPPING AVEC HTML/JAVASCRIPT
	 <p>The image displays four logos arranged in a diamond shape. At the top are the logos for HTML (orange shield), JS (yellow shield), and CSS (blue shield). Below them is the OpenLayers logo, which consists of a blue cube icon and the text 'OpenLayers'.</p>
01/09/2021	Document de formation
	<p>Auteur: Riadh Tébourbi – Supcom 2021</p>

I- INTRODUCTION

La cartographie Web est le processus de conception, mise en œuvre et le déploiement d'applications Web qui utilisent des données spatiales. Pour mettre en œuvre des applications de cartographie Web de nombreuses API JavaScript existent et avec la combinaison avec d'autres technologies web comme HTML5, Bootstrap, JQuery, etc., vous pouvez créer des applications très riches et très performantes. Openlayers est une puissante API JavaScript de développement Web Mapping, soutenue par la communauté et open source. Cet atelier vous permettra d'apprendre à développer des applications de cartographie Web utilisant Openlayers à partir de zéro. Il vous guidera à travers l'API OpenLayers de la manière la plus simple et la plus efficace possible.

II - PRESENTATION DE L'API OPENLAYERS

OpenLayers est une bibliothèque de fonctions JavaScript permettant le développement d'applications de cartographie en ligne.

OpenLayers peut afficher des données tuilées provenant de serveurs cartographiques en WMS ou sous forme de formes géométriques à partir d'une grande variété de sources de données.

Les classes principales de OpenLayers

Map

La carte est le composant principal de OpenLayers. Pour créer et afficher une carte nous avons besoin d'une vue, d'une ou plusieurs couches spatiales provenant de sources de données, et d'un conteneur cible qui est un contrôle déclaré au niveau du code html.

View

La vue détermine la façon avec laquelle la carte est rendue. Elle est utilisée pour définir la résolution, le centre de la carte, le zoom, l'étendue, etc.

Layers

Les couches spatiales peuvent être ajoutées à la carte dans un ordre empilé. Les couches inférieures sont rendues avant couches supérieures. Les couches peuvent être des couches raster (images), ou des couches vectorielles (points / lignes / polygones).

Source

Chaque couche de données possède une source attachée, une sorte de lien qui indique comment charger le contenu de la couche. Ça peut être une source provenant d'un serveur WMS ou tout simplement une source de données vectorielle dont les données sont dans un format particulier (par exemple GeoJSON ou KML), dans ce cas la couche spatiale est remplie d'objet géométriques (Features).

Features

Ce sont des objets géométriques stockés dans des couches et qui représentent le monde réel. Elles peuvent être rendues en utilisant des géométries de base (comme point, ligne ou polygone) et en utilisant un style donné (épaisseur de la ligne, la couleur de remplissage, etc.).

Une des composantes importantes pour pouvoir développer autour de OpenLayers est la documentation. N'hésitez pas donc à visiter la documentation de OpenLayers sur le site officiel:



<https://openlayers.org/en/latest/apidoc/>

Et voir les exemples sur le site:



<https://openlayers.org/en/latest/examples/>

III - PREPARATION DE L'ENVIRONNEMENT DE DEVELOPPEMENT

Dans cette formation nous allons utiliser un éditeur de texte qui intègre un environnement de développement avancé: WebStrom. Nous allons commencer par créer un projet dans lequel nous allons utiliser ce code html de base (squelette):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Un squelette d'application OL</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width">

  <!-- OpenLayers CSS -->
  <link rel="stylesheet" href="css/ol.css" type="text/css">
  <!-- Autres styles de librairies -->

  <!-- nos styles personnalisés-->
  <style>

  </style>
</head>
<body>
  <!-- Notre code HTML -->
  <!--déclaration de l'objet map-->
  <div id="map">
  </div>

  <!-- OpenLayers JS-->
  <script src="resources/ol.js" type="text/javascript"></script>

  <!-- Autres librairies JS-->

  <!-- Notre code JavaScript-->
  <script>
  </script>
```

```
</body>  
</html>
```

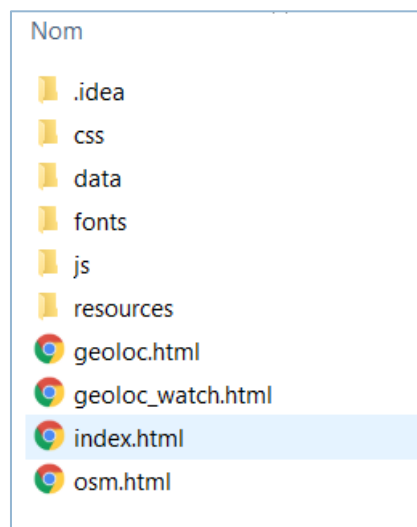
Dans ce code html remarque qu'on a fait appel à la bibliothèque OpenLayers (fichier css et js).

Nous avons ensuite déclaré un composant « map » qui servira à afficher le contenu d'un objet Map qui sera créé dans le code Javascript.

De plus, et afin de développer nos applications de webmapping riches, nous disposons d'une panoplie de bibliothèques JavaScript: jQuery, Bootstrap, etc.

Le répertoire de notre projet sera organisé de la façon suivante:

- Les fichiers html sont stockés dans le répertoire racine,
- Les fichiers de style css seront stockés dans le sous répertoire "css",
- Les librairies JavaScript seront stockées dans le sous répertoire "resources",
- Notre code JavaScript sera mis dans le répertoire "js".



Dans "Autres librairies JavaScript" chargez la librairie JQuery ("jquery.js").

IV - CREER UNE PREMIERE CARTE

1) Afficher une cartographie OSM

Nous allons compléter notre code "squelette" en affichant sur notre page web une carte provenant de Open Street Map.

Pour cela nous allons commencer par créer dans notre projet un fichier HTML "osm.html" dans le répertoire racine du projet et un fichier de code JavaScript "osm.js" dans le répertoire "/js" du projet.

Insérez ensuite le code "squelette" du paragraphe III dans le fichier "osm.html" et faites appel dans ce fichier à votre script "osm.js":

```
<script src="./js/osm.js"></script>
```

Dans le fichier "osm.js" écrivez le code suivant:

```
var lyr_osm = new ol.layer.Tile({
  title: 'OSM',
  type: 'base',
  visible: true,
  source: new ol.source.OSM()
});

var mapView = new ol.View({
  center: ol.proj.transform([0, 0], 'EPSG:4326', 'EPSG:3857'),
  zoom: 3
});

var layersList = [lyr_osm];

var map = new ol.Map({
  target: 'map',
  layers: layersList,
  view: mapView
});
```

Dans ce code JavaScript :

- Nous avons déclaré un objet `lyr_osm` qui est une instance de la classe « layer » de OpenLayers. Il est de type « Tile » (tuiles) et a comme source de données le service OSM (`source: new ol.source.OSM()`)
- Nous avons déclaré un objet « `mapView` » de la classe View: la vue est un composant qui sert à afficher un rendu de la carte. La propriété « center » permet de center la View donc la carte à un point géographique donné.
- Nous avons déclaré un tableau `layersList` qui contient un seul élément `lyr_osm` (`var layersList = [lyr_osm]`).

L'objectif maintenant c'est d'utiliser ces objets en les reliant à un objet Map qu'on va créer.

Nous avons alors déclaré un objet "map" qui est une instance de la classe Map de OpenLayers. Cet objet contient des propriétés à définir: *target*, *layers* et *view* qui pointent vers les objets déclarés auparavant :

target: Nous avons "mappé" l'objet "map" au contrôle (container html) "map" de notre code html: **target: 'map'**. Ceci permet de lier un composant créé dans HTML avec un objet créé dans le code Javascript.

layers: Sert à définir les **sources** des couches spatiales à afficher. Nous avons utilisé comme source Open Street Map.

view: un objet qui sert à afficher un rendu de la carte

Vous avez remarqué la référence vers deux systèmes de projections dans mapView: "EPSG: 4326" et "EPSG:3857". La ligne:

```
ol.proj.transform([0, 0], 'EPSG:4326', 'EPSG:3857')
```

indique que nous voulons transformer un point [0,0] dans le système de coordonnées 4326 vers le système 3857.

- EPSG: 3857: projection sphérique de Mercator. Cette projection fournit un système de coordonnées projetées utilisée par de nombreux fournisseurs de données libres, comme OpenStreetMap, et des services Web, tels que Google Maps ou Microsoft Virtual Earth. Cela signifie donc que lorsqu'on affiche une cartographie de OSM on est forcément dans le système de coordonnées EPSG 3875.

- EPSG: 4326: il s'agit ici du système de coordonnées non projetées (latitude / longitude) WGS84.

Le site <http://dogeo.fr/apps/projection/> vous donne les coordonnées géographiques dans les différents systèmes de projections.

Dans la propriété "Center" de "View" nous voulions centrer la carte sur le point de coordonnées géographiques en WGS84 (latitude, longitude) [0,0] (EPSG: 4326), or OSM travaille dans le système de coordonnées EPSG: 3857. Il était donc nécessaire de transformer ces coordonnées de EPSG: 4326 vers EPSG: 3857 à l'aide de la fonction `ol.proj.transform`.



Initialisez la carte affichée en zoomant sur Abidjan.



Personnaliser le style de la carte: vous pouvez personnaliser le style css du composants "map" déclaré dans le fichier "osm.html". Pour cela nous allons créer un fichier de style "osm.css" dans le répertoire "./css" du projet et y insérer le code suivant:

```
#map {
  width:100%;
  height:50%;
  position:absolute;
  right:0px;
  top:50px;
  bottom:0;
  overflow:hidden;
  background-color: #E9E9E9;
}
```

Il faudra ensuite faire référence à ce fichier "osm.css" dans le fichier html "osm.html":

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

2) Afficher une cartographie provenant d'un serveur cartographique

Nous allons maintenant apprendre à afficher des couches spatiales publiées sur le serveur GeoServer. Commencez d'abord par publier les couches spatiales se trouvant dans le répertoire « formation_sig\data » sur GeoServer si vous ne l'avez pas déjà fait lors de la thématique de formation précédente. On suppose donc que vous disposez des services cartographiques suivants sur Geoserver :

Couches

Gérer les couches publiées via GeoServer

➕ Ajouter une nouvelle ressource

➖ Retirer les ressources sélectionnées

<< < 1 > >> Résultats 1 à 8 (sur les 8 correspondances des 13 articles)

formation_gs Clear

<input type="checkbox"/>	Type	Titre	Nom de la couche	Entrepôt	Activée ?	SRC natif
<input type="checkbox"/>		Abidjan_HR_ext	formation_gs:Abidjan_HR_ext	Abidjan_HR	✓	EPSG:4326
<input type="checkbox"/>		civ_adm1	formation_gs:civ_adm1	adm_ci	✓	EPSG:4326
<input type="checkbox"/>		civ_adm2	formation_gs:civ_adm2	adm_ci	✓	EPSG:4326
<input type="checkbox"/>		civ_adm3	formation_gs:civ_adm3	adm_ci	✓	EPSG:4326
<input type="checkbox"/>		gis_osm_landuse	formation_gs:gis_osm_landuse	vecteur_osm	✓	EPSG:4326
<input type="checkbox"/>		gis_osm_places	formation_gs:gis_osm_places	vecteur_osm	✓	EPSG:4326
<input type="checkbox"/>		gis_osm_pois	formation_gs:gis_osm_pois	vecteur_osm	✓	EPSG:4326
<input type="checkbox"/>		gis_osm_roads	formation_gs:gis_osm_roads	vecteur_osm	✓	EPSG:4326

Nous allons maintenant utiliser OpenLayers pour visualiser ces couches publiées sur le WEB. Nous allons commencer par afficher les couches :

formation_gs:gis_osm_landuse

formation_gs:gis_osm_roads

formation_gs:gis_osm_pois

formation_gs:gis_osm_places

Commencez par créer le fichier html "geoserver.html" à partir du code html "squelette".

Créez ensuite le fichier de code JavaScript "geoserver.js" dans le répertoire "/js" et insérez le code suivant:

```
//URL Geoserver
var url_geoserver = "http://localhost:8080/geoserver/wms";

//noms des couches
var name_layer_landuse = "formation_gs:gis_osm_landuse";
var name_layer_roads = "formation_gs:gis_osm_roads";
var name_layer_pois = "formation_gs:gis_osm_pois";
var name_layer_places = "formation_gs:gis_osm_places";

//déclaration des couches openlayers
var lyr_landuse = new ol.layer.Tile({
  source: new ol.source.TileWMS({
    url: url_geoserver,
    params: {"LAYERS": name_layer_landuse, "TILED": "true"}
  }),
  title: "Occupation du sol"
});
```

```
var lyr_roads = new ol.layer.Tile({
  source: new ol.source.TileWMS(({
    url: url_geoserver,
    params: {"LAYERS": name_layer_roads, "TILED": "true"}
  })),
  title: "Routes"
});

var lyr_pois = new ol.layer.Tile({
  source: new ol.source.TileWMS(({
    url: url_geoserver,
    params: {"LAYERS": name_layer_pois, "TILED": "true"}
  })),
  title: "POIs"
});

var lyr_places = new ol.layer.Tile({
  source: new ol.source.TileWMS(({
    url: url_geoserver,
    params: {"LAYERS": name_layer_places, "TILED": "true"}
  })),
  title: "Lieux"
});

//visibilité par défaut des couches au chargement de la carte
lyr_landuse.setVisible(true);
lyr_roads.setVisible(true);
lyr_pois.setVisible(true);
lyr_places.setVisible(true);

//déclaration de la liste des couches à afficher dans un ordre précis
var layersList = [lyr_landuse, lyr_roads, lyr_pois, lyr_places];

var mapView = new ol.View({
  projection: 'EPSG:4326',
  center: [-5.690183, 7.786829],
  zoom: 7
});
var map = new ol.Map({
  target: 'map',
  layers: layersList,
  view: mapView
});
```

Le code commence par définir l'url du serveur cartographique. Celui-ci servira les couches selon le protocole WMS.

Ensuite nous avons définie l'url de la première couche à visualiser avec OpenLayers et publiée sur Geoserver. Le format de cette url est le suivant: **"espace de travail:nom de la couche"** :

```
var name_layer_landuse = "formation_gs:gis_osm_landuse";
```

Vient ensuite la déclaration de l'objet couche (Layer) de Openlayers. Dans cette déclaration nous avons fourni comme paramètres l'url de GéoServer et l'url de la couche concernée. Ici le type de la source de données est « TileWMS » :


```
var lyr_landuse = new ol.layer.Tile({
  source: new ol.source.TileWMS({
    url: url_geoserver,
    params: {"LAYERS": name_layer_landuse, "TILED": "true"}
  })),
  title: "Occupation du sol"
});
```

Dans la déclaration de l'objet "map" nous avons alors fourni comme paramètres la liste des couches à visualiser définie dans le tableau "layersList". La projection utilisée ici est EPSG:4326 c'est à dire la projection géographique (latitude, longitude) WGS84. Nous avons forcé OpenLayers à utiliser cette projection car toutes nos données sur Geoserver ont pour système de coordonnées origine EPSG :4326.

Géoserver peut très bien fournir les couches soit dans le système de projection d'origine ou dans le système de coordonnées géographique WGS84 (EPSG:4326) ou même dans le système EPSG: 3857.

Si vous voulez utiliser EPSG :3857 dans votre application vous pouvez très bien changer la déclaration de l'objet MapView par :

```
var mapView = new ol.View({
  projection: 'EPSG:3857',
  center: new ol.geom.Point([-5.690183, 7.786829]).transform('EPSG:4326',
'EPSG:3857').getCoordinates(),
  zoom: 7
});
```

ceci produira le même résultat au niveau de l'application, sauf que Openlayers utilisera EPSG :3857 et Geoserver devra fournir les données non plus en EPSG :4326 mais en EPSG :3857.



Complétez le fichier geoserver.js afin de visualiser les couches publiées sur GeoServer :

Les limites administratives

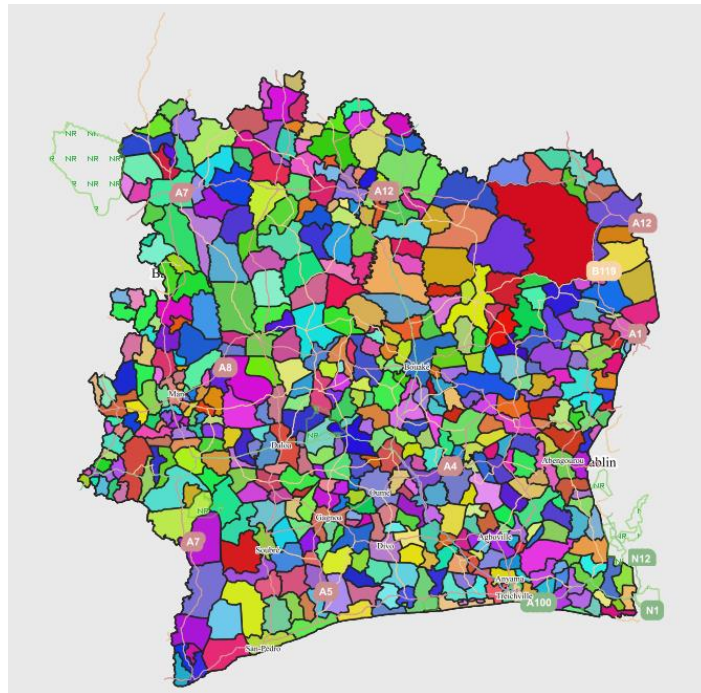
formation_gs:civ_adm1 (districts)

formation_gs:civ_adm2 (régions)

formation_gs:civ_adm3 (départements)

L'image HR formation_gs:Abidjan_HR_ext

2-1) LayerSwitcher: Quand vous avez rajouté toutes les couches publiées sur GéoServer dans votre application vous avez certainement remarqué qu'il serait pratique de disposer d'un outil permettant de gérer l'affichage de chaque couche.



L'extension LayerSwitcher pour Openlayers est là pour ça. Nous allons alors utiliser ce composant dans l'objet "map" après sa création:

```
var layerSwitcher = new ol.control.LayerSwitcher({
  tipLabel: 'Légende'
});
map.addControl(layerSwitcher);
```

Avant que votre code puisse fonctionner vous devez d'abord faire appel aux deux fichiers `ol-layerswitcher.css` et `ol-layerswitcher.js` dans votre projet. Ces deux fichiers constituent une extension pour OpenLayers dans laquelle est définie la classe `LayerSwitcher`.

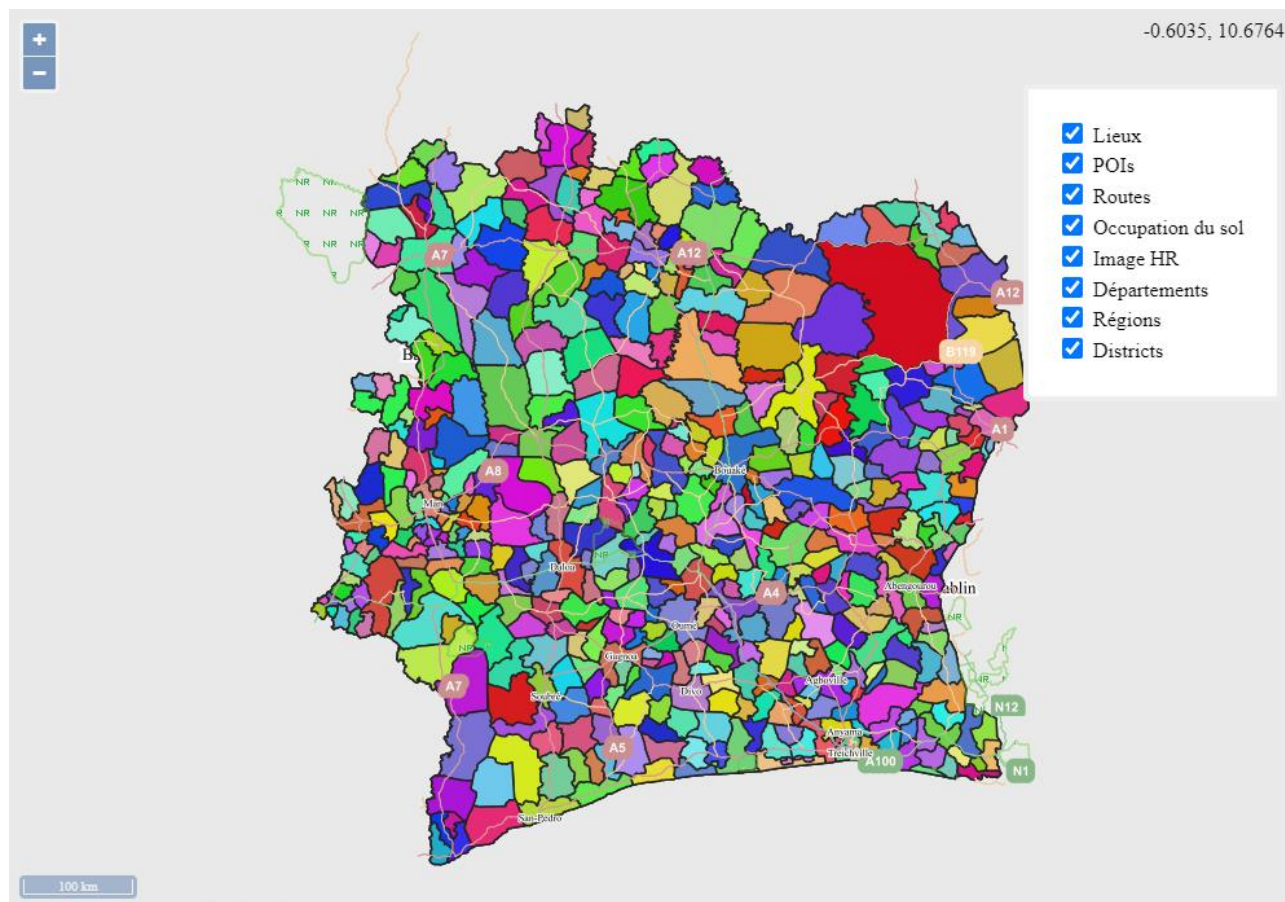
2-2) C'est quoi les "Controls" ?

Chaque composant que vous utilisez dans OpenLayers est appelé "Control". Il existe plusieurs composants qu'on peut utiliser dans OpenLayers et nous allons en découvrir plusieurs plus tard. Cependant, nous allons tout de suite utiliser deux composants: le premier permet d'afficher une barre d'échelle sur la carte: il s'agit du composant `ol.control.ScaleLine()`. Le deuxième permet d'afficher les coordonnées géographiques de la souris:

```
var MousePosition = new ol.control.MousePosition({
  coordinateFormat: ol.coordinate.createStringXY(4),
  projection: 'EPSG:4326'
});
```



Rajoutez ces deux composants dans la liste des composants à utiliser dans l'objet "map". Vous devriez obtenir le résultat suivant :



La couleur de fond de la carte: vous pouvez modifier la couleur de fond de plan de la carte affichée en créant un fichier de style "geoserver.css" comme dans la section précédente mais en y rajoutant la propriété: background-color: #E9E9E9;

3) Superposer les données publiées sur Géoserver sur OSM



Vous allez maintenant créer une nouvelle application en créant les fichiers "osm_gs.html" et "osm_gs.js" qui superpose les données publiées sur Geoserver sur la carte provenant de OSM. Le fichier "osm_gs.js" est presque identique à "geoserver.js" à une exception près: OSM utilise le système de projection EPSG:3857. Dans cette application vous devez :

- Mettre OSM comme fond de plan
- Rajouter les couches de Geoserver : limites administratives et image HR

V - INTERAGIR AVEC LA CARTE

Dans cette section vous allez apprendre à interagir avec la carte. Pour cela nous allons commencer par répondre à l'évènement liés au clic et au mouvement de la souris.

1) Mouse move:

On aimerait d'abord pouvoir afficher les coordonnées de la souris dans les deux systèmes de coordonnées EPSG:3857 et EPSG:4326.

Pour cela modifiez le fichier "osm_gs.html" en y ajoutant deux labels:

```
<p>Coordonnées géographiques de la souris dans les différents systèmes de projection:</p>
<p>
  EPSG:3857 > <span id="mouse3857" class="label">0 / 0</span>
  EPSG:4326 > <span id="mouse4326" class="label">0 / 0</span>
</p>
```

Le premier label est référencé dans le code html par son id="mouse3857", le deuxième par son id="4326". Nous allons modifier dynamiquement le contenu de ces deux labels dans "osm_gs.js" par les valeurs des coordonnées de la souris:

```
map.on('pointermove', function(event) {
  var coord3857 = event.coordinate;
  var coord4326 = ol.proj.transform(coord3857, 'EPSG:3857', 'EPSG:4326');

  $('#mouse3857').text(ol.coordinate.toStringXY(coord3857, 2));
  $('#mouse4326').text(ol.coordinate.toStringXY(coord4326, 5));
});
```

Nous venons de faire appel à l'évènement "pointermove" de OpenLayers. La variable "event" contient les coordonnées de la souris (event.coordinate). Par défaut les coordonnées sont dans le système EPSG:3857 (puisque l'on utilise OSM), il suffit donc de les transformer en EPSG:4326 pour récupérer les coordonnées en WGS84.

Ces coordonnées sont ensuite affichées dans les labels correspondants. Remarquez l'utilisation du "\$" de JQuery permettant de faire appel, dans un code JavaScript, à un composant déclaré dans une balise html. Ceci nous évite d'utiliser à chaque fois l'instruction document.getElementById(...) pour faire appel à un composant html et nous offre donc plus de souplesse dans notre code. Vous devez donc ajouter une référence vers JQuery dans votre code HTML (Autres librairies JS) :

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

Vous devez aussi modifier le fichier de css pour mettre la propriété : « top:90px; » du composant #map afin de pouvoir afficher les coordonnées dans une zone de 90 pixels au-dessus de la carte.

Coordonnées géographiques de la souris dans les différents systèmes de projection:

EPSG:3857 > -488503.67, 922095.30 EPSG:4326 > -4.38830, 8.25462



2) Mouse clic:

On veut maintenant détecter le clic sur la carte à l'aide de la souris et récupérer les coordonnées de la souris du point cliqué. Pour cela nous allons utiliser l'événement "singleclick" de OpenLayers:

```
map.on('singleclick', function(evt) {
    onSingleClick(evt);
});
```

La fonction JavaScript "onSingleClick" contient le code à exécuter lorsque l'utilisateur clic sur la carte:

```
var onSingleClick = function(evt) {
    var coord = evt.coordinate;
    console.log(coord);
}
```

Comme vous le voyez on se contente d'afficher ces coordonnées sur la console, nous allons plus tard utiliser cet événement pour faire une tâche plus intéressante: afficher des informations sur une couche spatiale provenant de Géoserver au point cliqué.



Les coordonnées affichées sur la console sont en EPSG:3857, modifier le code pour afficher les coordonnées en EPSG:4326.

3) Des "Popups" pour afficher des informations:

Ca serait maintenant intéressant d'utiliser une info-bulle (Popup) afin d'afficher les coordonnées au point cliqué.

Dans le fichier osm_gs.html commencez par déclarer les composants html qui permettront d'afficher les informations sous la déclaration du composant "map":

Donc remplacer :

Par :

```
<!--déclaration de l'objet map-->
<div id="map">
  <div id="popup" class="ol-popup">
    <a href="#" id="popup-closer" class="ol-popup-closer"></a>
    <div id="popup-content"></div>
  </div>
</div>
```

Dans ce code il y'a déclaration de trois composants:

- "popup": fait référence au container de l'info-bulle (une sorte de "frame" ou "container") qui contiendra les deux autres composants.
- "popup-closer": qui est cliquable puisqu'il contient une balise "href" et qui va servir à fermer la "Popup".
- "popup-content": c'est la partie qui contiendra le texte à afficher.

Nous allons ensuite déclarer les mêmes composants dans le code JavaScript afin de pouvoir les remplir dynamiquement en fonction du clic de l'utilisateur:

```
//Definition des popups pour affichage des infos
var container = document.getElementById('popup');
var content = document.getElementById('popup-content');
var closer = document.getElementById('popup-closer');

closer.onclick = function() {
  container.style.display = 'none';
  closer.blur();
  return false;
};
```

Dans ce code nous avons également définie une fonction permettant de répondre à l'événement clic sur le "popup-closer", ce code fait disparaître le popup lorsqu'on clic sur le bouton "fermer" du popup.

Pour pouvoir superposer (overlay) ce "popup", défini par la variable "container", à notre "map" nous allons utiliser un objet de la classe Overlay de OpenLayers:

```
var overlayPopup = new ol.Overlay({
  element: container
});
```

Nous venons de déclarer un objet overlayPopup de la classe Overlay de OpenLayers et nous avons spécifié que cet objet contiendra comme élément ("element") notre "container".

En résumé nous avons créer un container html qui sera superposé à notre "map" grâce à un objet "Overlay" de OpenLayers.

Il suffit maintenant de rajouter cet Overlay à la liste des "overlays" de l'objet maps (propriété "overlays"):

```
var map = new ol.Map({
  target: 'map',
  overlays: [overlayPopup],
```

```

    layers: layersList,
    view: mapView
  });

```

Nous allons attaquer maintenant la partie du code qui consiste à modifier le contenu (content) de ce "container" d'une façon dynamique, il suffit donc de modifier la fonction "onSingleClick":

```

var onSingleClick = function(evt) {
  var coord = evt.coordinate;
  console.log(coord);
  var coord_wgs84 = ol.proj.toLonLat(coord);
  var str = ol.coordinate.toStringXY(coord_wgs84, 6);
  if(str) {
    str = '<p>' + str + '</p>';
    overlayPopup.setPosition(coord);
    content.innerHTML = str;
    container.style.display = 'block';
  }
  else{
    container.style.display = 'none';
    closer.blur();
  }
}

```



Modifiez le style du Popup grâce à une définition de syle dans le fichier "osm_gs.css":

```

.ol-popup {
  position: absolute;
  background-color: white;
  box-shadow: 0 1px 4px rgba(0,0,0,0.2);
  padding: 15px;
  border-radius: 10px;
  border: 1px solid #cccccc;
  bottom: 12px;
  left: -50px;
  min-width: 280px;
}
.ol-popup:after, .ol-popup:before {
  top: 100%;
  border: solid transparent;
  content: " ";
  height: 0;
  width: 0;
  position: absolute;
  pointer-events: none;
}
.ol-popup:after {
  border-top-color: white;
  border-width: 10px;
  left: 48px;
  margin-left: -10px;
}
.ol-popup:before {

```



```

border-top-color: #cccccc;
border-width: 11px;
left: 48px;
margin-left: -11px;
}
.ol-popup-closer {
text-decoration: none;
position: absolute;
top: 2px;
right: 8px;
}
.ol-popup-closer:after {
content: "X";
}

```

Nous allons plus tard utiliser ce Popup pour afficher les informations sur une couche spatiale provenant de Géoserver au point cliqué.

4) Interroger les couches spatiales:

Nous voudrions maintenant afficher dans notre info-bulle des informations sur les couches spatiales au point cliqué. Ces informations proviennent du serveur cartographique en WMS et sont stockées dans les tables des attributs des couches publiées.

Pour comprendre cela je vous invite d'abord à voir la documentation sur Geoserver sur cette page:



<http://docs.geoserver.org/latest/en/user/services/wms/reference.html>

Dans cette documentation nous voyant les possibilités qu'offre GeoServer concernant le protocole WMS dans le tableau suivant:

Operations	
WMS requests can perform the following operations:	
Operation	Description
Exceptions	If an exception occur
GetCapabilities	Retrieves metadata about the service, including supported operations and parameters, and a list of the available layers
GetMap	Retrieves a map image for a specified area and content
GetFeatureInfo (optional)	Retrieves the underlying data, including geometry and attribute values, for a pixel location on a map
DescribeLayer (optional)	Indicates the WFS or WCS to retrieve additional information about the layer.
GetLegendGraphic (optional)	Retrieves a generated legend for a map

Nous voyons donc que parmi les opérations qu'on pourrait faire nous en avons une qui nous intéresse: "GetFeatureInfo". Le principe est le même que celui d'aller chercher une "map" (getMap): tout se base sur une URL Request vers le serveur cartographique. Il suffit donc de générer et d'envoyer une url qui

demande à Geoserver de nous fournir les valeurs des attributs de couches spatiales en un point géographique donné.

Par exemple l'url suivante permet de retrouver les informations sur la couche des Adm1 à partir de GéoServer:

http://localhost:8080/geoserver/formation_gs/wms?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetFeatureInfo&FORMAT=image/png&TRANSPARENT=true&QUERY_LAYERS=formation_gs:civ_adm1&STYLES&LAYERS=formation_gs:civ_adm1&INFO_FORMAT=text/html&FEATURE_COUNT=50&X=50&Y=50&SRS=EPSG:4326&WIDTH=101&HEIGHT=101&BBOX=-5.136108398437502,5.5810546875,-4.026489257812502,6.690673828125

Parmi les paramètres fournis dans cette requête, il existe un paramètre intéressant: FORMAT. Ce paramètre indique à Geoserver le format des données dans lequel nous voulons les récupérer. Parmi les autres possibilités (que vous pouvez tester) nous trouvons aussi:

- ✓ **text/plain**: format texte simple
- ✓ **text/xml**: format xml
- ✓ **text/html**: format html
- ✓ **application/json**: format json
- ✓ **text/javascript**: format JavaScript ou Jsonp (Json padding).

Le dernier format est celui qu'on va utiliser car il est le plus flexible. En JSONP, vous ne renvoyez pas juste du JSON mais un appel de fonction avec du JSON comme seul et unique paramètre. Ceci est très pratique lors d'échanges asynchrones entre une application cliente et un serveur. Grâce à jQuery nous pouvons traiter du Jsonp très facilement. En effet, à la fin de son chargement, le script sera interprété, la fonction en question exécutée et jQuery se chargera d'appeler vos callbacks en leur transmettant les données reçues: c'est la magie d'ajax.



<http://api.jquery.com/jquery.ajax/#jQuery-ajax-settings>

Testez la requête avec comme format de retour jsonp (**text/javascript**) et remarquez le nom du script à appeler pour traiter les données retournées: "parseResponse".

```
parseResponse({"type":"FeatureCollection","features":[{"type":"Feature","id":"civ_adm1.1","geometry":{"type":"MultiPolygon","coordinates":[[[[-5.0933,6.2872],[
-5.0722,6.2865],[
-5.0699,6.2879],[
-5.0662,6.2884],[
-5.063,6.2888],[
-5.0611,6.2899],[
-5.0591,6.2911],[
-5.0573,6.2922],[
-5.0542,6.2931],[
-5.0523,6.2938],[
-5.050
-5.0311,6.2922],[
-5.0278,6.2905],[
-5.0253,6.2891],[
-5.0244,6.2873],[
-5.0251,6.2834],[
-5.0263,6.28],[
-5.0286,6.277],[
-5.0304,6.2749],[
-5.0321,6.2729],[
-5.0337,
-5.0431,6.2595],[
-5.0423,6.2574],[
-5.0413,6.2552],[
-5.0404,6.2529],[
-5.0395,6.2509],[
-5.0392,6.2478],[
-5.0398,6.2456],[
-5.0402,6.2429],[
-5.0392,6.2407],[
-5.03
-5.0399,6.2371],[
-5.0389,6.2381],[
-5.0381,6.2391],[
-5.0373,6.2391],[
-5.0365,6.2391],[
-5.0358,6.2391],[
-5.0351,6.2391],[
-5.0344,6.2391],[
-5.0337,6.2391],[
-5.033,6.2391],[
-5.0323,6.2391],[
-5.0316,6.2391],[
-5.0309,6.2391],[
-5.0302,6.2391],[
-5.0295,6.2391],[
-5.0288,6.2391],[
-5.0281,6.2391],[
-5.0274,6.2391],[
-5.0267,6.2391],[
-5.026,6.2391],[
-5.0253,6.2391],[
-5.0246,6.2391],[
-5.0239,6.2391],[
-5.0232,6.2391],[
-5.0225,6.2391],[
-5.0218,6.2391],[
-5.0211,6.2391],[
-5.0204,6.2391],[
-5.0197,6.2391],[
-5.019,6.2391],[
-5.0183,6.2391],[
-5.0176,6.2391],[
-5.0169,6.2391],[
-5.0162,6.2391],[
-5.0155,6.2391],[
-5.0148,6.2391],[
-5.0141,6.2391],[
-5.0134,6.2391],[
-5.0127,6.2391],[
-5.012,6.2391],[
-5.0113,6.2391],[
-5.0106,6.2391],[
-5.0099,6.2391],[
-5.0092,6.2391],[
-5.0085,6.2391],[
-5.0078,6.2391],[
-5.0071,6.2391],[
-5.0064,6.2391],[
-5.0057,6.2391],[
-5.005,6.2391],[
-5.0043,6.2391],[
-5.0036,6.2391],[
-5.0029,6.2391],[
-5.0022,6.2391],[
-5.0015,6.2391],[
-5.0008,6.2391],[
-5.0001,6.2391],[
-4.9994,6.2391],[
-4.9987,6.2391],[
-4.998,6.2391],[
-4.9973,6.2391],[
-4.9966,6.2391],[
-4.9959,6.2391],[
-4.9952,6.2391],[
-4.9945,6.2391],[
-4.9938,6.2391],[
-4.9931,6.2391],[
-4.9924,6.2391],[
-4.9917,6.2391],[
-4.991,6.2391],[
-4.9903,6.2391],[
-4.9896,6.2391],[
-4.9889,6.2391],[
-4.9882,6.2391],[
-4.9875,6.2391],[
-4.9868,6.2391],[
-4.9861,6.2391],[
-4.9854,6.2391],[
-4.9847,6.2391],[
-4.984,6.2391],[
-4.9833,6.2391],[
-4.9826,6.2391],[
-4.9819,6.2391],[
-4.9812,6.2391],[
-4.9805,6.2391],[
-4.9798,6.2391],[
-4.9791,6.2391],[
-4.9784,6.2391],[
-4.9777,6.2391],[
-4.977,6.2391],[
-4.9763,6.2391],[
-4.9756,6.2391],[
-4.9749,6.2391],[
-4.9742,6.2391],[
-4.9735,6.2391],[
-4.9728,6.2391],[
-4.9721,6.2391],[
-4.9714,6.2391],[
-4.9707,6.2391],[
-4.97,6.2391],[
-4.9693,6.2391],[
-4.9686,6.2391],[
-4.9679,6.2391],[
-4.9672,6.2391],[
-4.9665,6.2391],[
-4.9658,6.2391],[
-4.9651,6.2391],[
-4.9644,6.2391],[
-4.9637,6.2391],[
-4.963,6.2391],[
-4.9623,6.2391],[
-4.9616,6.2391],[
-4.9609,6.2391],[
-4.9602,6.2391],[
-4.9595,6.2391],[
-4.9588,6.2391],[
-4.9581,6.2391],[
-4.9574,6.2391],[
-4.9567,6.2391],[
-4.956,6.2391],[
-4.9553,6.2391],[
-4.9546,6.2391],[
-4.9539,6.2391],[
-4.9532,6.2391],[
-4.9525,6.2391],[
-4.9518,6.2391],[
-4.9511,6.2391],[
-4.9504,6.2391],[
-4.9497,6.2391],[
-4.949,6.2391],[
-4.9483,6.2391],[
-4.9476,6.2391],[
-4.9469,6.2391],[
-4.9462,6.2391],[
-4.9455,6.2391],[
-4.9448,6.2391],[
-4.9441,6.2391],[
-4.9434,6.2391],[
-4.9427,6.2391],[
-4.942,6.2391],[
-4.9413,6.2391],[
-4.9406,6.2391],[
-4.9399,6.2391],[
-4.9392,6.2391],[
-4.9385,6.2391],[
-4.9378,6.2391],[
-4.9371,6.2391],[
-4.9364,6.2391],[
-4.9357,6.2391],[
-4.935,6.2391],[
-4.9343,6.2391],[
-4.9336,6.2391],[
-4.9329,6.2391],[
-4.9322,6.2391],[
-4.9315,6.2391],[
-4.9308,6.2391],[
-4.9301,6.2391],[
-4.9294,6.2391],[
-4.9287,6.2391],[
-4.928,6.2391],[
-4.9273,6.2391],[
-4.9266,6.2391],[
-4.9259,6.2391],[
-4.9252,6.2391],[
-4.9245,6.2391],[
-4.9238,6.2391],[
-4.9231,6.2391],[
-4.9224,6.2391],[
-4.9217,6.2391],[
-4.921,6.2391],[
-4.9203,6.2391],[
-4.9196,6.2391],[
-4.9189,6.2391],[
-4.9182,6.2391],[
-4.9175,6.2391],[
-4.9168,6.2391],[
-4.9161,6.2391],[
-4.9154,6.2391],[
-4.9147,6.2391],[
-4.914,6.2391],[
-4.9133,6.2391],[
-4.9126,6.2391],[
-4.9119,6.2391],[
-4.9112,6.2391],[
-4.9105,6.2391],[
-4.9098,6.2391],[
-4.9091,6.2391],[
-4.9084,6.2391],[
-4.9077,6.2391],[
-4.907,6.2391],[
-4.9063,6.2391],[
-4.9056,6.2391],[
-4.9049,6.2391],[
-4.9042,6.2391],[
-4.9035,6.2391],[
-4.9028,6.2391],[
-4.9021,6.2391],[
-4.9014,6.2391],[
-4.9007,6.2391],[
-4.9,6.2391],[
-4.8993,6.2391],[
-4.8986,6.2391],[
-4.8979,6.2391],[
-4.8972,6.2391],[
-4.8965,6.2391],[
-4.8958,6.2391],[
-4.8951,6.2391],[
-4.8944,6.2391],[
-4.8937,6.2391],[
-4.893,6.2391],[
-4.8923,6.2391],[
-4.8916,6.2391],[
-4.8909,6.2391],[
-4.8902,6.2391],[
-4.8895,6.2391],[
-4.8888,6.2391],[
-4.8881,6.2391],[
-4.8874,6.2391],[
-4.8867,6.2391],[
-4.886,6.2391],[
-4.8853,6.2391],[
-4.8846,6.2391],[
-4.8839,6.2391],[
-4.8832,6.2391],[
-4.8825,6.2391],[
-4.8818,6.2391],[
-4.8811,6.2391],[
-4.8804,6.2391],[
-4.8797,6.2391],[
-4.879,6.2391],[
-4.8783,6.2391],[
-4.8776,6.2391],[
-4.8769,6.2391],[
-4.8762,6.2391],[
-4.8755,6.2391],[
-4.8748,6.2391],[
-4.8741,6.2391],[
-4.8734,6.2391],[
-4.8727,6.2391],[
-4.872,6.2391],[
-4.8713,6.2391],[
-4.8706,6.2391],[
-4.8699,6.2391],[
-4.8692,6.2391],[
-4.8685,6.2391],[
-4.8678,6.2391],[
-4.8671,6.2391],[
-4.8664,6.2391],[
-4.8657,6.2391],[
-4.865,6.2391],[
-4.8643,6.2391],[
-4.8636,6.2391],[
-4.8629,6.2391],[
-4.8622,6.2391],[
-4.8615,6.2391],[
-4.8608,6.2391],[
-4.8601,6.2391],[
-4.8594,6.2391],[
-4.8587,6.2391],[
-4.858,6.2391],[
-4.8573,6.2391],[
-4.8566,6.2391],[
-4.8559,6.2391],[
-4.8552,6.2391],[
-4.8545,6.2391],[
-4.8538,6.2391],[
-4.8531,6.2391],[
-4.8524,6.2391],[
-4.8517,6.2391],[
-4.851,6.2391],[
-4.8503,6.2391],[
-4.8496,6.2391],[
-4.8489,6.2391],[
-4.8482,6.2391],[
-4.8475,6.2391],[
-4.8468,6.2391],[
-4.8461,6.2391],[
-4.8454,6.2391],[
-4.8447,6.2391],[
-4.844,6.2391],[
-4.8433,6.2391],[
-4.8426,6.2391],[
-4.8419,6.2391],[
-4.8412,6.2391],[
-4.8405,6.2391],[
-4.8398,6.2391],[
-4.8391,6.2391],[
-4.8384,6.2391],[
-4.8377,6.2391],[
-4.837,6.2391],[
-4.8363,6.2391],[
-4.8356,6.2391],[
-4.8349,6.2391],[
-4.8342,6.2391],[
-4.8335,6.2391],[
-4.8328,6.2391],[
-4.8321,6.2391],[
-4.8314,6.2391],[
-4.8307,6.2391],[
-4.83,6.2391],[
-4.8293,6.2391],[
-4.8286,6.2391],[
-4.8279,6.2391],[
-4.8272,6.2391],[
-4.8265,6.2391],[
-4.8258,6.2391],[
-4.8251,6.2391],[
-4.8244,6.2391],[
-4.8237,6.2391],[
-4.823,6.2391],[
-4.8223,6.2391],[
-4.8216,6.2391],[
-4.8209,6.2391],[
-4.8202,6.2391],[
-4.8195,6.2391],[
-4.8188,6.2391],[
-4.8181,6.2391],[
-4.8174,6.2391],[
-4.8167,6.2391],[
-4.816,6.2391],[
-4.8153,6.2391],[
-4.8146,6.2391],[
-4.8139,6.2391],[
-4.8132,6.2391],[
-4.8125,6.2391],[
-4.8118,6.2391],[
-4.8111,6.2391],[
-4.8104,6.2391],[
-4.8097,6.2391],[
-4.809,6.2391],[
-4.8083,6.2391],[
-4.8076,6.2391],[
-4.8069,6.2391],[
-4.8062,6.2391],[
-4.8055,6.2391],[
-4.8048,6.2391],[
-4.8041,6.2391],[
-4.8034,6.2391],[
-4.8027,6.2391],[
-4.802,6.2391],[
-4.8013,6.2391],[
-4.8006,6.2391],[
-4.8,6.2391],[
-4.7993,6.2391],[
-4.7986,6.2391],[
-4.7979,6.2391],[
-4.7972,6.2391],[
-4.7965,6.2391],[
-4.7958,6.2391],[
-4.7951,6.2391],[
-4.7944,6.2391],[
-4.7937,6.2391],[
-4.793,6.2391],[
-4.7923,6.2391],[
-4.7916,6.2391],[
-4.7909,6.2391],[
-4.7902,6.2391],[
-4.7895,6.2391],[
-4.7888,6.2391],[
-4.7881,6.2391],[
-4.7874,6.2391],[
-4.7867,6.2391],[
-4.786,6.2391],[
-4.7853,6.2391],[
-4.7846,6.2391],[
-4.7839,6.2391],[
-4.7832,6.2391],[
-4.7825,6.2391],[
-4.7818,6.2391],[
-4.7811,6.2391],[
-4.7804,6.2391],[
-4.7797,6.2391],[
-4.779,6.2391],[
-4.7783,6.2391],[
-4.7776,6.2391],[
-4.7769,6.2391],[
-4.7762,6.2391],[
-4.7755,6.2391],[
-4.7748,6.2391],[
-4.7741,6.2391],[
-4.7734,6.2391],[
-4.7727,6.2391],[
-4.772,6.2391],[
-4.7713,6.2391],[
-4.7706,6.2391],[
-4.7699,6.2391],[
-4.7692,6.2391],[
-4.7685,6.2391],[
-4.7678,6.2391],[
-4.7671,6.2391],[
-4.7664,6.2391],[
-4.7657,6.2391],[
-4.765,6.2391],[
-4.7643,6.2391],[
-4.7636,6.2391],[
-4.7629,6.2391],[
-4.7622,6.2391],[
-4.7615,6.2391],[
-4.7608,6.2391],[
-4.7601,6.2391],[
-4.7594,6.2391],[
-4.7587,6.2391],[
-4.758,6.2391],[
-4.7573,6.2391],[
-4.7566,6.2391],[
-4.7559,6.2391],[
-4.7552,6.2391],[
-4.7545,6.2391],[
-4.7538,6.2391],[
-4.7531,6.2391],[
-4.7524,6.2391],[
-4.7517,6.2391],[
-4.751,6.2391],[
-4.7503,6.2391],[
-4.7496,6.2391],[
-4.7489,6.2391],[
-4.7482,6.2391],[
-4.7475,6.2391],[
-4.7468,6.2391],[
-4.7461,6.2391],[
-4.7454,6.2391],[
-4.7447,6.2391],[
-4.744,6.2391],[
-4.7433,6.2391],[
-4.7426,6.2391],[
-4.7419,6.2391],[
-4.7412,6.2391],[
-4.7405,6.2391],[
-4.7398,6.2391],[
-4.7391,6.2391],[
-4.7384,6.2391],[
-4.7377,6.2391],[
-4.737,6.2391],[
-4.7363,6.2391],[
-4.7356,6.2391],[
-4.7349,6.2391],[
-4.7342,6.2391],[
-4.7335,6.2391],[
-4.7328,6.2391],[
-4.7321,6.2391],[
-4.7314,6.2391],[
-4.7307,6.2391],[
-4.73,6.2391],[
-4.7293,6.2391],[
-4.7286,6.2391],[
-4.7279,6.2391],[
-4.7272,6.2391],[
-4.7265,6.2391],[
-4.7258,6.2391],[
-4.7251,6.2391],[
-4.7244,6.2391],[
-4.7237,6.2391],[
-4.723,6.2391],[
-4.7223,6.2391],[
-4.7216,6.2391],[
-4.7209,6.2391],[
-4.7202,6.2391],[
-4.7195,6.2391],[
-4.7188,6.2391],[
-4.7181,6.2391],[
-4.7174,6.2391],[
-4.7167,6.2391],[
-4.716,6.2391],[
-4.7153,6.2391],[
-4.7146,6.2391],[
-4.7139,6.2391],[
-4.7132,6.2391],[
-4.7125,6.2391],[
-4.7118,6.2391],[
-4.7111,6.2391],[
-4.7104,6.2391],[
-4.7097,6.2391],[
-4.709,6.2391],[
-4.7083,6.2391],[
-4.7076,6.2391],[
-4.7069,6.2391],[
-4.7062,6.2391],[
-4.7055,6.2391],[
-4.7048,6.2391],[
-4.7041,6.2391],[
-4.7034,6.2391],[
-4.7027,6.2391],[
-4.702,6.2391],[
-4.7013,6.2391],[
-4.7006,6.2391],[
-4.7,6.2391],[
-4.6993,6.2391],[
-4.6986,6.2391],[
-4.6979,6.2391],[
-4.6972,6.2391],[
-4.6965,6.2391],[
-4.6958,6.2391],[
-4.6951,6.2391],[
-4.6944,6.2391],[
-4.6937,6.2391],[
-4.693,6.2391],[
-4.6923,6.2391],[
-4.6916,6.2391],[
-4.6909,6.2391],[
-4.6902,6.2391],[
-4.6895,6.2391],[
-4.6888,6.2391],[
-4.6881,6.2391],[
-4.6874,6.2391],[
-4.6867,6.2391],[
-4.686,6.2391],[
-4.6853,6.2391],[
-4.6846,6.2391],[
-4.6839,6.2391],[
-4.6832,6.2391],[
-4.6825,6.2391],[
-4.6818,6.2391],[
-4.6811,6.2391],[
-4.6804,6.2391],[
-4.6797,6.2391],[
-4.679,6.2391],[
-4.6783,6.2391],[
-4.6776,6.2391],[
-4.6769,6.2391],[
-4.6762,6.2391],[
-4.6755,6.2391],[
-4.6748,6.2391],[
-4.6741,6.2391],[
-4.6734,6.2391],[
-4.6727,6.2391],[
-4.672,6.2391],[
-4.6713,6.2391],[
-4.6706,6.2391],[
-4.6699,6.2391],[
-4.6692,6.2391],[
-4.6685,6.2391],[
-4.6678,6.2391],[
-4.6671,6.2391],[
-4.6664,6.2391],[
-4.6657,6.2391],[
-4.665,6.2391],[
-4.6643,6.2391],[
-4.6636,6.2391],[
-4.6629,6.2391],[
-4.6622,6.2391],[
-4.6615,6.2391],[
-4.6608,6.2391],[
-4.6601,6.2391],[
-4.6594,6.2391],[
-4.6587,6.2391],[
-4.658,6.2391],[
-4.6573,6.2391],[
-4.6566,6.2391],[
-4.6559,6.2391],[
-4.6552,6.2391],[
-4.6545,6.2391],[
-4.6538,6.2391],[
-4.6531,6.2391],[
-4.6524,6.2391],[
-4.6517,6.2391],[
-4.651,6.2391],[
-4.6503,6.2391],[
-4.6496,6.2391],[
-4.6489,6.2391],[
-4.6482,6.2391],[
-4.6475,6.2391],[
-4.6468,6.2391],[
-4.6461,6.2391],[
-4.6454,6.2391],[
-4.6447,6.2391],[
-4.644,6.2391],[
-4.6433,6.2391],[
-4.6426,6.2391],[
-4.6419,6.2391],[
-4.6412,6.2391],[
-4.6405,6.2391],[
-4.6398,6.2391],[
-4.6391,6.2391],[
-4.6384,6.2391],[
-4.6377,6.2391],[
-4.637,6.2391],[
-4.6363,6.2391],[
-4.6356,6.2391],[
-4.6349,6.2391],[
-4.6342,6.2391],[
-4.6335,6.2391],[
-4.6328,6.2391],[
-4.6321,6.2391],[
-4.6314,6.2391],[
-4.6307,6.2391],[
-4.63,6.2391],[
-4.6293,6.2391],[
-4.6286,6.2391],[
-4.6279,6.2391],[
-4.6272,6.2391],[
-4.6265,6.2391],[
-4.6258,6.2391],[
-4.6251,6.2391],[
-4.6244,6.2391],[
-4.6237,6.2391],[
-4.623,6.2391],[
-4.6223,6.2391],[
-4.6216,6.2391],[
-4.6209,6.2391],[
-4.6202,6.2391],[
-4.6195,6.2391],[
-4.6188,6.2391],[
-4.6181,6.2391],[
-4.6174,6.2391],[
-4.6167,6.2391],[
-4.616,6.2391],[
-4.6153,6.2391],[
-4.6146,6.2391],[
-4.6139,6.2391],[
-4.6132,6.2391],[
-4.6125,6.2391],[
-4.6118,6.2391],[
-4.6111,6.2391],[
-4.6104,6.2391],[
-4.6097,6.2391],[
-4.609,6.2391],[
-4.6083,6.2391],[
-4.6076,6.2391],[
-4.6069,6.2391],[
-4.6062,6.2391],[
-4.6055,6.2391],[
-4.6048,6.2391],[
-4.6041,6.2391],[
-4.6034,6.2391],[
-4.6027,6.2391],[
-4.602,6.2391],[
-4.6013,6.2391],[
-4.6006,6.2391],[
-4.6,6.2391],[
-4.5993,6.2391],[
-4.5986,6.2391],[
-4.5979,6.2391],[
-4.5972,6.2391],[
-4.5965,6.2391],[
-4.5958,6.2391],[
-4.5951,6.2391],[
-4.5944,6.2391],[
-4.5937,6.2391],[
-4.593,6.2391],[
-4.5923,6.2391],[
-4.5916,6.2391],[
-4.5909,6.2391],[
-4.5902,6.2391],[
-4.5895,6.2391],[
-4.5888,6.2391],[
-4.5881,6.2391],[
-4.5874,6.2391],[
-4.5867,6.2391],[
-4.586,6.2391],[
-4.5853,6.2391],[
-4.5846,6.2391],[
-4.5839,6.2391],[
-4.5832,6.2391],[
-4.5825,6.2391],[
-4.5818,6.2391],[
-4.5811,6.2391],[
-4.5804,6.2391],[
-4.5797,6.2391],[
-4.579,6.2391],[
-4.5783,6.2391],[
-4.5776,6.2391],[
-4.5769,6.2391],[
-4.5762,6.2391],[
-4.5755,6.2391],[
-4.5748,6.2391],[
-4.5741,6.2391],[
-4.5734,6.2391],[
-4.5727,6.2391],[
-4.572,6.2391],[
-4.5713,6.2391],[
-4.5706,6.2391],[
-4.5699,6.2391],[
-4.5692,6.2391],[
-4.56
```

```

var clicked_coord;
var onSingleClick = function(evt) {
    var coord = evt.coordinate;
    console.log(coord);

    var source1 = name_layer_adm1;
    var source2 = name_layer_adm2;
    var source3 = name_layer_adm3;
    var layers_list = source3 + ',' + source2 + ',' + source1;
    var wmslyr_adm1 = new ol.source.TileWMS({
        url: url_geoserver,
        params: {'LAYERS': name_layer_adm1, 'TILED': true},
        serverType: 'geoserver',
        crossOrigin: 'anonymous'
    });
    var view = map.getView();
    var viewResolution = view.getResolution();
    var url=wmslyr_adm1.getFeatureInfoUrl(
        evt.coordinate, viewResolution, view.getProjection(),
        { 'INFO_FORMAT': 'text/javascript',
          'FEATURE_COUNT': 20,
          'LAYERS': layers_list,
          'QUERY_LAYERS': layers_list
        });
    console.log(url);
    if (url) { //call parseResponse(data)
        clicked_coord = coord;
        $.ajax(url,
            {dataType: 'jsonp'}
        ).done(function (data) {
            });
    }
}

```

Nous avons commencé par définir les sources de données qui nous intéressent pour le GetInfo, ici il s'agit des couches limites administratives: nous voulons afficher le District, Région et Département au point cliqué.

Nous avons ensuite déclaré une source de données de type WMS provenant de Geoserver `wmslyr_adm1 = new ol.source.TileWMS` qui va nous permettre d'interroger la couche.

Vient ensuite l'appel à la méthode "getFeatureInfoUrl" appliqué à la source "wmslyr_adm1" mais qui prend en paramètre la liste d'autres couches à interroger : "layers_list".

Si l'url est bien formée nous faisons appel à la fonction Ajax de jQuery pour interroger Geoserver avec cette url d'une façon asynchrone.

Quand le serveur cartographique répond une fonction "callback" est appelée automatiquement par jQuery il s'agit de la méthode "parseResponse " qu'il faudra donc implémenter:

```

function parseResponse(data) {
    var vectorSource = new ol.source.Vector({
        features: (new ol.format.GeoJSON()).readFeatures(data)
    });
    console.log((new ol.format.GeoJSON()).readFeatures(data));
    var features = vectorSource.getFeatures();
    var str="";

```

```

var district = "";
var region = "";
var departement = "";

for(x in features) {
    var id = features[x].getId();
    console.log(id);
    var props = features[x].getProperties();
    if(id.indexOf("adm1")>-1) district = props["ADM1_FR"];
    if(id.indexOf("adm2")>-1) region = props["ADM2_FR"];
    if(id.indexOf("adm3")>-1) departement = props["ADM3_FR"];
}

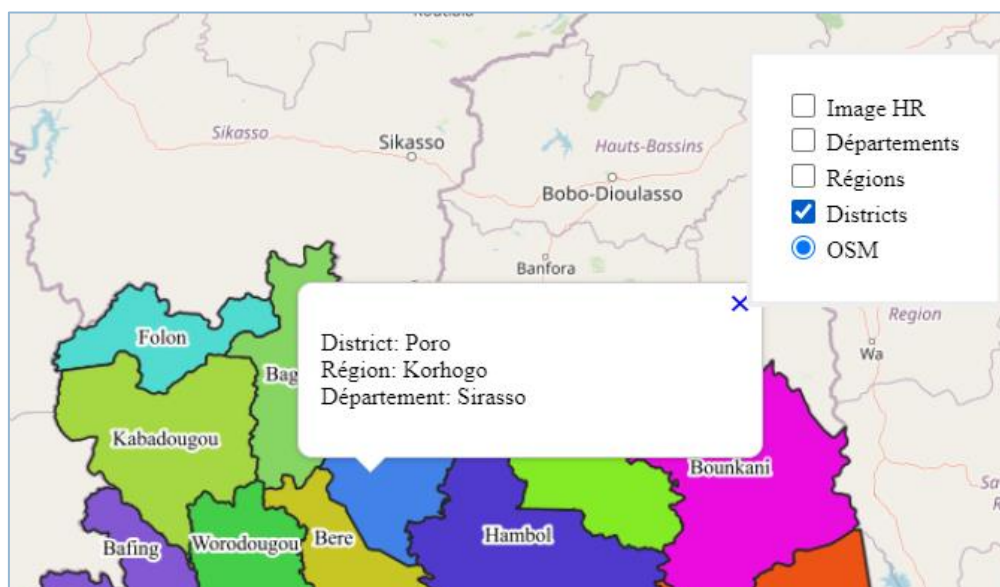
str = str + "District: " + district+ '<br/>';
str = str + "Région: " + region+ '<br/>';
str = str + "Département: " + departement+ '<br/>';
if(str) {
    str = '<p>' + str + '</p>';
    overlayPopup.setPosition(clicked_coord);
    content.innerHTML = str;
    container.style.display = 'block';
}
else{
    container.style.display = 'none';
    closer.blur();
}
}

```

C'est dans cette fonction que sera utilisé notre "Popup" défini plus haut pour afficher les attributs des objets interrogés.

Nous commençons par créer une source de donnée vecteur à partir des données json récupérées. Nous parcourons ensuite les objets créés de cette source de données et nous récupérons les attributs des limites administratives (s'ils existent). Nous apprendrons dans le chapitre suivant à manipuler des sources de données vecteurs d'une façon dynamique.

Nous remplissons enfin notre info-bulle avec du html contenant les informations en question.



VI - MANIPULER DES SOURCES DE DONNEES VECTEURS

Dans ce chapitre nous allons apprendre à créer, à intégrer et à afficher des données géographiques d'une façon dynamique et par programmation. Ces données peuvent être par exemple :

- Des objets géographiques créés à la volée,
- Des points provenant d'un fichier XML, kml ou GeoJson,
- Des points, des lignes ou des polygones dessinés par l'utilisateur,

1) Créer dynamiquement des objets géographiques

Dans un premier temps nous allons ajouter par programmation deux objets géométriques à notre carte: un point et un cercle. Nous devons donc suivre 4 étapes:

- Créer les objets géométriques (ou formes géométriques),
- Créer une source de données vecteur à partir de ces objets,
- Créer une couche et l'associer à cette source,
- Ajouter la couche créée à la carte.

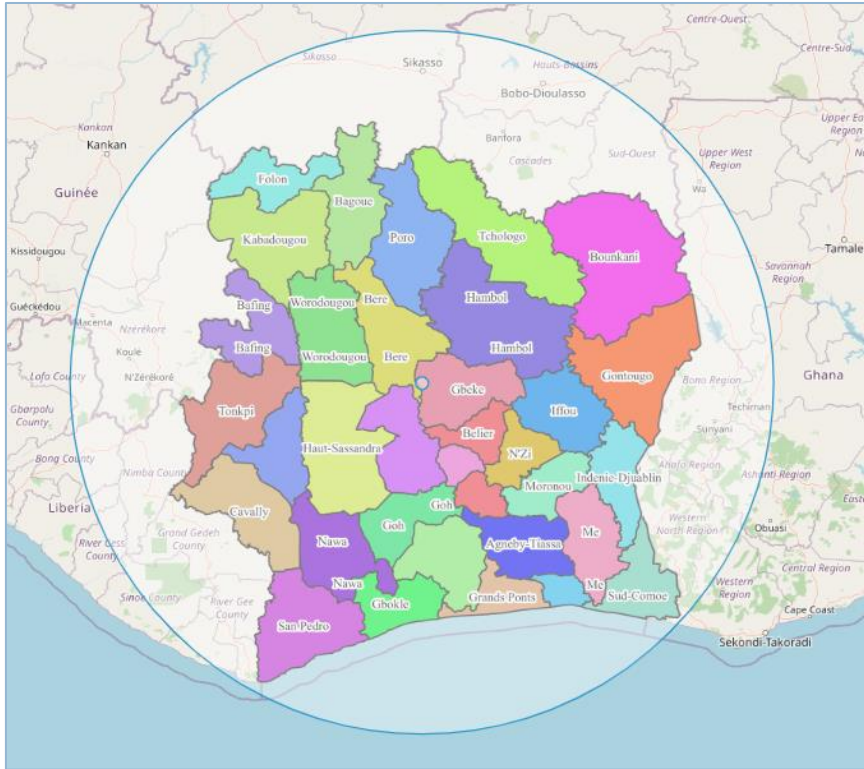
```
// Define Geometries
var point = new ol.geom.Point(
  ol.proj.transform([-5.690183, 7.786829], 'EPSG:4326', 'EPSG:3857')
);
var circle = new ol.geom.Circle(
  ol.proj.transform([-5.690183, 7.786829], 'EPSG:4326', 'EPSG:3857'),
  450000
);

// Features
var pointFeature = new ol.Feature(point);
var circleFeature = new ol.Feature(circle);

// Source
var vectorSource = new ol.source.Vector({
  projection: 'EPSG:4326'
});
vectorSource.addFeatures([pointFeature, circleFeature]);

// vector layer
var vectorLayer = new ol.layer.Vector({
  source: vectorSource
});

//add layer to the map
map.addLayer(vectorLayer);
```



2) Changeons les styles

Vous pouvez changer le style de "dessin" des objets géographiques créés précédemment grâce à la classe "Style" de OpenLayers. Le principe est simple: nous allons définir un style de dessin que nous allons ensuite attribuer à notre couche vecteur:

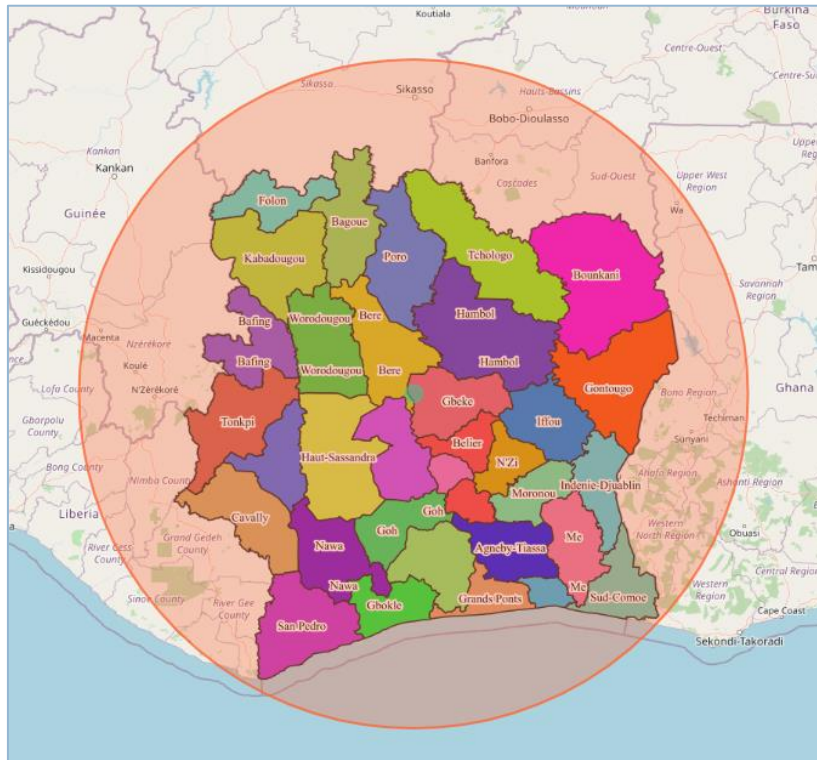
```
var style = new ol.style.Style({
  fill: new ol.style.Fill({
    color: 'rgba(255, 100, 50, 0.3)'
  }),
  stroke: new ol.style.Stroke({
    width: 2,
    color: 'rgba(255, 100, 50, 0.8)'
  }),
  image: new ol.style.Circle({
    fill: new ol.style.Fill({
      color: 'rgba(55, 200, 150, 0.5)'
    }),
    stroke: new ol.style.Stroke({
      width: 1,
      color: 'rgba(55, 200, 150, 0.8)'
    }),
    radius: 7
  })
});
```

```
// vector layer with the style
var vectorLayer = new ol.layer.Vector({
  source: vectorSource,
```

```

    style: style
  });

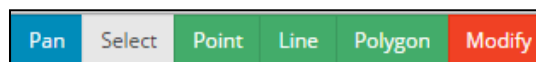
```



3) Et si on dessinait

Au lieu de rajouter des objets géométriques dans le code, nous allons permettre à l'utilisateur de dessiner les objets qu'il veut grâce à des outils de dessin. Ces objets peuvent être des points des lignes et des polygones.

Pour cela nous allons d'abord modifier notre application afin d'offrir à l'utilisateur la possibilité de choisir quel objet il veut dessiner. On veut créer des boutons poussoirs ou une barre d'outils qui contient les choix suivant:



Pan: Déplacer la carte,

Select: Sélectionner un objet dessiné,

Point: Dessiner un point,

Line: Dessiner une ligne,

Polygon: Dessiner un polygone,

Modify: Modifier une forme géométrique.

Commençons donc par créer ces boutons dans notre code html:

```
<!-- Notre code HTML -->
<div class="btn-group btn-group-sm" role="group" aria-label="Draw">
  <button id="pan" type="button" class="btn btn-primary">Pan</button>
  <button id="select" type="button" class="btn btn-default">Select</button>

  <button id="point" type="button" class="btn btn-success">Point</button>
  <button id="line" type="button" class="btn btn-success">Line</button>
  <button id="polygon" type="button" class="btn btn-success">Polygon</button>
  <button id="modify" type="button" class="btn btn-danger">Modify</button>
</div>
<!--déclaration de l'objet map-->
```

Et pour avoir de jolis boutons comme sur la figure nous allons utiliser les bibliothèques Bootstrap et jQuery-ui:

au niveau des styles:

```
<!-- Autres styles de librairies -->
<!-- bootswatch yeti theme et jqueryy-ui-->
<link rel="stylesheet" type="text/css" href="./css/jquery-ui.css">
<link href="//maxcdn.bootstrapcdn.com/bootswatch/3.2.0/yeti/bootstrap.min.css"
rel="stylesheet">
```

et au niveau des APIs JavaScript:

```
<!-- Autres librairies JS-->
<script type="text/javascript" src="./resources/jquery.js"></script>
<script type="text/javascript" src="./resources/jquery-ui.js"></script>
<script type="text/javascript" src="./resources/bootstrap.min.js"></script>
```

Maintenant que nous avons préparé notre interface IHM pour répondre à notre besoin, il est temps maintenant de s'occuper de la partie Métier dans le code JavaScript:

```
var vectorSource = new ol.source.Vector({
  projection: 'EPSG:4326'
});

// vector layer
var vectorLayer = new ol.layer.Vector({
  source: vectorSource,
  style: style
});
//add layer to the map
map.addLayer(vectorLayer);

var button = $('#pan').button('toggle');
var interaction;
$('#div.btn-group button').on('click', function(event) {
  var id = event.target.id;

  // Toggle buttons
  button.button('toggle');
  button = $('#'+id).button('toggle');
  // Remove previous interaction
  map.removeInteraction(interaction);
  // Update active interaction
```

```
switch(event.target.id) {
  case "select":
    interaction = new ol.interaction.Select();
    map.addInteraction(interaction);
    break;
  case "point":
    interaction = new ol.interaction.Draw({
      type: 'Point',
      source: vectorLayer.getSource()
    });
    map.addInteraction(interaction);
    break;
  case "line":
    interaction = new ol.interaction.Draw({
      type: 'LineString',
      source: vectorLayer.getSource()
    });
    map.addInteraction(interaction);
    break;
  case "polygon":
    interaction = new ol.interaction.Draw({
      type: 'Polygon',
      source: vectorLayer.getSource()
    });
    map.addInteraction(interaction);
    break;
  case "modify":
    interaction = new ol.interaction.Modify({
      features: new ol.Collection(vectorLayer.getSource().getFeatures())
    });
    map.addInteraction(interaction);
    break;
  default:
    break;
}
});
```

Le code se base sur la classe "interaction" de OpenLayers. Cette classe apporte au programmeur des fonctionnalités d'interaction avec la carte. Il existe plusieurs types d'interactions possibles avec la carte qu'OpenLayers offre à l'utilisateur:



https://openlayers.org/en/latest/apidoc/module-ol_interaction.html

Comme vous le voyez dessiner c'est aussi interagir avec la carte. Avant d'utiliser une interaction nous d'abord supprimer l'interaction ajoutée précédemment avec "removeInteraction" puis ajouter l'interaction désirée avec "addInteraction". Mais avant d'ajouter une interaction nous devons d'abord la créer. Par exemple pour créer une interaction de type "Draw":

```
interaction = new ol.interaction.Draw({
  type: 'Point',
  source: vectorLayer.getSource()
});
```


Nous avons fait appel au constructeur de l'interaction "Draw" et comme paramètres nous avons choisi "Point" comme type de géométrie à dessiner et nous avons spécifier que les objets dessinés seront stockés dans notre couche vecteur "VectorLayer".

VII - GEOLOCALISATION

Dans ce chapitre nous allons apprendre à afficher la position géographique de l'utilisateur. Cette position sera mise à jour automatiquement et suivra le déplacement de l'utilisateur.

La géolocalisation nécessite donc de recentrer la carte sur la position de l'utilisateur et de mettre celle-ci en évidence avec un marker. Une icône sera alors affichée à la position de l'utilisateur. Pour cela, nous allons utiliser une icône de la bibliothèque d'icônes "Font-awesome":



<https://fontawesome.github.io/Font-Awesome/icons/>

C'est une bibliothèque très riche et très utilisée par la communauté des développeur web.

Nous commençons par faire référence à cette bibliothèque dans notre fichier html:

```
<link rel="stylesheet" type="text/css" href="./css/font-awesome.css">
```

Ce fichier de style contient la liste des styles des différents icones. Les icones elles-mêmes se trouvent dans le répertoire "Fonts" à copier dans la racine du projet.

Il faudra ensuite déclarer un contrôle dans notre code html qui contiendra notre icône:

```
<div id="location" class="marker"><i class="fa fa-street-view fa-lg" style="color:lawngreen"></i></div>
```

Notez que nous avons référencé ce contrôle par un id="location".

Vient ensuite la partie Métier qui affichera la position actuelle de l'utilisateur en utilisant cette icône:

```
//Geolocation
var geolocation = new ol.Geolocation({
  projection: map.getView().getProjection(),
  tracking: true
});
var marker = new ol.Overlay({
  element: document.getElementById('location'),
  positioning: 'center-center'
});
map.addOverlay(marker);

geolocation.on('change:position', function() { //real time tracking
  map.getView().setCenter(geolocation.getPosition());
  map.getView().setZoom(15);
  marker.setPosition(geolocation.getPosition());
});
```

Nous utilisons ici la classe "geolocation" de OpenLayers:



https://openlayers.org/en/latest/apidoc/module-ol_Geolocation-Geolocation.html

Une instance de la classe "geolocation" est d'abord créé. Comme paramètre il est important de spécifier le système de projection dans lequel nous voulons travailler, ici on indique que l'on veut travailler avec la projection utilisée dans la "map". Le deuxième paramètre indique si oui ou non on veut mettre à jour cette position lorsque celle-ci change.

Vous pouvez aussi utiliser un événement de "geolocation" celui de "change:position". Vous pouvez par exemple, si l'événement se réalise, zoomer sur la position actuelle de l'utilisateur: vous êtes alors en train de Tracker l'utilisateur : `geolocation.on('change:position'...`

Afin d'afficher notre icône à la position de l'utilisateur, nous avons créé un objet "marker" (qui n'est autre qu'un simple "overlay") qui pointe vers l'icône référencée dans le code html par id="location".

On ajoute ensuite notre "marker" à notre carte et nous changeons la position du marker (`marker.setPosition`) à chaque fois que la position change (dans la fonction `geolocation.on('change:position')`)

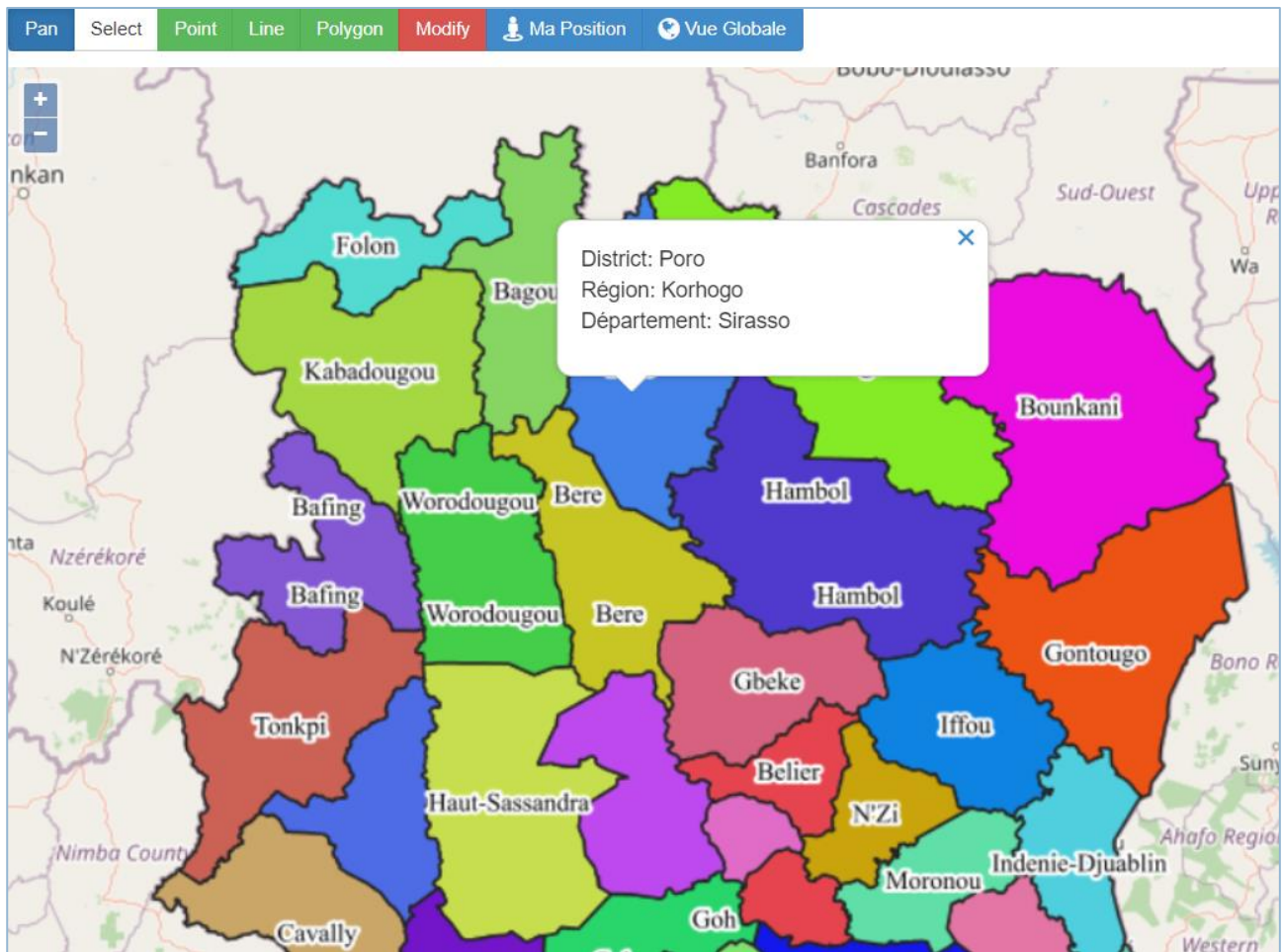


utilisez les méthodes "geolocation.getPosition()", "map.getView().setCenter(..)" et "map.getView().setZoom(...)" et ajouter un bouton permettant de zoomer sur la position de l'utilisateur.

Utilisez la méthode "map.getView().fit(...)" et "map.getView().setZoom(...)" pour ajouter un bouton pour zoomer sur l'étendue géographique. Pour déclarer les boutons :

```
<button id="button_pos" type="button" class="btn btn-primary"
onclick="zoomToMyPosition()"><i class="fa fa-street-view fa-lg"></i> Ma
Position</button>
<button type="button" class="btn btn-primary" onclick="goToFullExtent()"><i
class="fa fa-globe fa-lg"></i> Vue Globale</button>
```

Vous devez alors implémenter les deux fonctions `zoomToMyPosition` et `goToFullExtent()`



VIII - AMELIORATION DE L'INTERFACE DE L'APPLICATION

Dans cette section nous allons améliorer le design de notre application en la rendant responsive en utilisant un thème Bootstrap. Le thème que nous allons utiliser peut être trouvé sur le site "BootstrapZero" sur lequel il existe des dizaines de "templates" gratuits:

<http://www.bootstrapzero.com/bootstrap-template/complex-navbar>



Téléchargez le thème "complex-navbar". Modifier votre application "osm_gs" afin qu'elle utilise le thème Bootstrap téléchargé. Votre application devra avoir l'allure suivante:

