Smartdata Documentation

Version 1.0

GrandLyon

20 May 2014

Table des matières

Bienvenue dans la documentation de SmartData. Cette documentation comporte plusieurs parties. La page services vous donnera des informations générales sur les services disponibles et leur utilisation. Par la suite, plusieurs exemples et extraits de code sont fournis dans la section exemples. ... à compléter...

CHAPITRE 1

Guide du développeur

Les différentes section de la documentation sont accessibles directement.

1.1 Les services offerts par GrandLyon Smart Data

La plateforme Smart Data vous permet d'accéder à différents services de consultation des données OpenData. Les services sont de deux types : des services de visualisation qui vous permettent d'afficher des images (cartes) et des services de téléchargement qui vous permettent d'accéder directement à la donnée, selon des modalités et des formats variés.

Lors de la mise en oeuvre d'une application cartographique, on ne peut pas traiter toutes les données de la même manière. Il y a en effet des données qui habillent (fond de carte par ex.), d'autres qui renseignent sur le contexte (points d'intérêt divers : mairie, gare...) et d'autres enfin qui portent l'information importante, celle que l'on souhaite vraiment mettre en avant et exploiter dans l'application (disponibilité des vélos, localisation des bus en temps réel...).

Le fond de plan sera toujours proposé au format image, non interactif, et très souvent issu d'un système de cache tuilé permettant la récupération rapide sous forme de petites tuiles (256 x 256 pixels) de son contenu. Ce sera toujours la couche la plus basse dans l'application, celle en dessous de toutes les autres, pour des raisons évidentes de visibilité.

Les informations cruciales, importantes, porteuses de la valeur ajoutée de l'application seront les plus interactives possibles, pour que d'un survol une info-bulle donne accès à l'essentiel, qu'un clic ouvre un fiche complète, qu'un changement de style (taille, couleur) dynamique permette de souligner la sélection, l'association, la relation avec un autre élément. Pour ce faire il faut donc utiliser un service de téléchargement, seul à même de transmettre les données brutes permettant la mise en oeuvre d'une couche vectorielle dans l'application cartographique ou de donner accès à tous les attributs (informations associées à l'objet géographique) directement.

Pour les autres couches de données il faut faire des arbitrages. On ne peut tout charger en vectoriel pour des raisons de lisibilité (trop de points/lignes qui clignotent, grossissent ou s'agitent en même temps rendent la carte inutilisable) et de performance (chaque point est inscrit dans le DOM de la page. Plusieurs millieurs de points deviennent très lourds à gérer pour le navigateur). Donc on transige. On utilise les services de visualisation (WMS, format image) pour les données dont l'emplacement l'information la plus importante (par ex : stationnement handicapé, il n'y a rien à mettre dans une info-bulle, l'important est que la place soit là où elle est, et il suffit donc de l'afficher) ou dont l'emprise spatiale a un intérêt particulier (contours d'un parc, d'une zone règlementée...). Et on choisit le format vectoriel pour quelques informations certes secondaires par rapport à l'objet principal de l'application, mais dont les caractérisques sont importantes à connaître (stations vélos, événement...)

1.1.1 Service WMS

Le service WMS est le service de visualisation par excellence. Il sert à "voir" la donnée géographique avec une mise en forme prédéfinie (couleurs, styles, symboles...). C'est le service à privilégier pour intégrer des jeux de données au

format image.

Il est accessible à partir de l'URL

https ://download.data.grandlyon.com/wms/[nom_du_service]

Généralement on fait appel à son opération getCapabilities pour connaître son contenu (liste des couches) :

https ://download.data.grandlyon.com/wms/grandlyon ?SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.3.0

renvoie ainsi un document XML listant (entre autres) les couches mises à disposition par le service, dont vous obtiendrez le contenu à l'aide d'une requête GetMap de ce type :

https ://download.data.grandlyon.com/wms/grandlyon ?LAYERS=adr_voie_lieu.adrcommune&FORMAT=image%2Fpng&EXCEPTIO

Ca peut vous sembler un peu compliqué et fastidieux... Mais les librairies cartographiques sont là pour vous aider. Leaflet et OpenLayers implémentent des classes WMS qui feront tout ça pour vous, en adaptant les requêtes selon les manipulations faites sur la carte. Il vous suffit généralement de renseigner l'URL de base du service et d'indiquer le nom de la couche que vous souhaitez utiliser comme l'illustrent nos *Exemples et extraits de code*.

1.1.2 Service WFS

Les services WFS permettent de récupérer la donnée brute telle qu'elle est enregistrée dans la base de données. Il s'agit d'un service de téléchargement, même si ce téléchargement (récupération de la donnée) n'est pas forcément complet, c'est-à-dire qu'on peut l'effectuer sur une partie seulement du territoire, notamment pour les données les plus volumineuses.

Il est accessible à partir de l'URL

https ://download.data.grandlyon.com/wfs/[nom_du_service]

Généralement on fait appel à son opération getCapabilities pour connaître son contenu (liste des couches) :

https ://download.data.grandlyon.com/wfs/grandlyon ?SERVICE=WFS&REQUEST=GetCapabilities&VERSION=1.1.0

renvoie ainsi un document XML listant (entre autres) les couches mises à disposition par le service, dont vous obtiendrez le contenu à l'aide d'une requête GetFeatures de ce type :

https ://download.data.grandlyon.com/wfs/grandlyon ?SERVICE=WFS&REQUEST=GetFeature&typename=pvo_patrimoine_voirie.pv

Mais là encore, rassurez-vous, les librairies cartographiques disposent des classes nécessaires à une utilisation simple de ce type de service.

Le format généralement utilisé en WFS est le GML (Geographic Markup Language) qui est un dérivé du XML. Toutefois, ce format n'est pas forcément le plus simple à utiliser dans le cadre d'une application web. Aussi, tout en utilisant le WFS, il est possible de recevoir un flux au format GeoJSON en ajoutant le paramètre OUTPUTFORMAT=geojson :

https ://download.data.grandlyon.com/wfs/grandlyon ?SERVICE=WFS&REQUEST=GetFeature&typename=pvo_patrimoine_voirie.pv

1.1.3 Services REST (en JSON)

Les services JSON de notre infrastructure permettent une navigation facile et rapide entre les différents jeux de données mis à disposition. Chaque service possède un point d'entrée dédié :

https ://download.data.grandlyon.com/ws/grandlyon/all.json

et

https ://download.data.grandlyon.com/ws/smartdata/all.json

Ces documents listent l'ensemble des tables disponibles en consultation/téléchargement. Certaines peuvent avoir un accès restreint en fonction de vos droits.

De liens en lien, vous pouvez alors naviguer vers la description des tables (par ex. https ://download.data.grandlyon.com/ws/grandlyon/fpc_fond_plan_communaut.fpcplandeau.json), les différentes valeurs présentes dans un champ particulier (par ex. les essences des arbres de la métropole : https ://download.data.grandlyon.com/ws/grandlyon/abr_arbres_alignement.abrarbre/essencefrancais.json). Ce dernier mode dispose d'options particulières :
– compact : si false, décrit la valeur pour chacun des enregistrements, sinon liste les différentes valeurs trouvées dans la table. True par défaut.
– maxfeatures : indique le nombre maximal d'enregistrement à faire remonter par le service. 1000 par défaut.
– start : indique l'index de départ, afin de pouvoir paginer les résultats. 1 par défaut.
On peut ainsi demander au service les essences de 50 arbres à partir du 100e dans la base :

https ://download.data.grandlyon.com/ws/grandlyon/abr_arbres_alignement.abrarbre/essencefrancais.json ?compact=false&maxfeatures

On peut également accéder à la totalité du contenu de la table (ou paginer ce contenu) en utilisant une URL du type :

https ://secure.grandlyon.webmapping.fr/ws/smartdata/jcd_jcdecaux.jcdvelov/all.json ?compact=false

pour consulter l'intégralité des enregistrements.

Les services REST-JSON sont ainsi particulièrement adaptés à la constition de listes de valeurs, de tableaux et de grilles paginés, d'interface de navigation au sein des données.

1.1.4 Service OSM (OpenStreetMap)

La plateforme SmartData propose un service de fond de carte tuilé construit à partir des données OpenStreetMap de la région Rhône-Alpes. Il est utilisable à partir de l'URL :

http ://openstreetmap.data.grandlyon.com

Le nom de couche à utiliser est tout simplement osm_grandlyon. La couche est disponibles dans les projections suivantes :
– ESPG :3857 et EPSG :900913 (Mercator Sphérique)
– EPSG :4326 (WGS84)
– EPSG :4171 (RGF93)
Veuillez noter que ces deux derniers systèmes sont définis en degrés et non en mètres, et que leur utilisation pour faire une carte (et non lire les données) aboutit à un résultat visuel un peu écrasé qui est tout à fait normal (puisque vous projetez de fait des coordonnées géographiques sphériques sur un plan, le fichier ou l'écran. Cette projection est nommée plate-carrée).

1.2 Authentification

1.2.1 Principes

Certaines des données publiées par les services SmartData nécessitent une autorisation. Afin d'en obtenir une, vous devez ouvrir un compte sur http ://smartdata.grandlyon.com/creation-de-compte/ et spécifier les différents jeux de données et modalités d'accès que vous souhaitez.

Une fois ces opération réalisées, vous aurez un identifiant (généralement l'adresse email utilisée lors de la création du compte) et un mot de passe. Ceux-ci vous sont personnels, et leur utilisation dans le contexte du développement d'application pour les tiers doit donc être fait avec certaines précautions.

Mais voyons d'abord comment utiliser ces éléments d'identification pour accéder à des données protégées.

La méthode d'authentification utilisée est le Basic Auth HTTP Lors de l'accès à chacun des types de services, il est donc nécessaire d'utiliser le header HTTP nommé 'Authorization' dans lequel seront insérés login et mot de passe, séparés par deux points (" :") et encodé en base64. Le header ressemble alors à ceci :

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Mais ne vous y méprenez pas. L'encodage en base64 n'est pas un cryptage ! Le procédé est réversible et on peut donc retrouver les valeurs encodés très facilement. Couplé à un flux HTTPS, qui crypte les informations transmises par et vers le serveur, ils ne sont pas récupérables, mais écrits tels quels dans le code ils le sont.

1.2.2 Exemple avec cURL et WGET

L'utilisation du header authorization avec cURL est très simple. Imaginons un utilisateur doté des identifiants suivants :
– login : demo
– password : demo4dev
L'instruction cURL à utiliser pour accéder à la donnée "demo.demovelov" sur le service smartdata serait alors :

```
cURL -u demo:demo4dev curl https://download.data.grandlyon.com/ws/smartdata/demo.demovelov/all.json?c
```

sauf erreur, vous devriez alors recevoir un flux json.

L'instruction WGET à utiliser est comparable :

```
wget --http-user=demo --http-password=demo4dev https://download.data.grandlyon.com/ws/smartdata/demo.
```

1.2.3 Exemples avec PHP ou Python

Que ce soit en PHP ou en Python, les librairies utilisées pour émettre des requêtes HTTP intègrent toutes les possibilité de déclarer le Header Authorization.

Pour Python et urllib2 nous aurons :

```python
import urllib2, base64

# set basic information
username = 'demo'
password = 'demo4dev'
url = 'https://download.data.grandlyon.com/ws/smartdata/demo.demovelov/all.json'

# prepare the request Object
request = urllib2.Request(url)

# encode the username / password couple into a single base64 string
base64string = base64.encodestring('%s:%s' % (username, password)))

# then add this string into the Authorization header
request.add_header("Authorization", "Basic %s" % base64string)

# and open the url
result = urllib2.urlopen(request)

# then handle the result the way you like
```

En PHP, nous utiliserons la librairie cURL intégrée :

```php
<?php

// set basic information
$username='demo';
$password='demo4dev';
```

```php
$URL='https://download.data.grandlyon.com/ws/smartdata/demo.demovelov/all.json';

// instantiate a new cUrl object
$ch = curl_init();

// Everything is an option with PHPCurl !
curl_setopt($ch, CURLOPT_URL,$URL);

// set RETURNTRANSFER to true to be able to handle the result
curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);

// set this option for enabling Basic Auth HTTP rules
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);

// the previous setting will help here to encode the username/password into the correct format
curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");

// and lift off...
$result=curl_exec ($ch);

// then handle the result the way you like

?>
```

1.3 Bonnes Pratiques

1.3.1 Principes

Que vous souhaitiez développer une application web ou embarquée dans un smartphone, vous avez sans doute le souci de garantir son bon fonctionnement et sa disponibilité. Voici quelques conseils destinés à vous faire utiliser notre plateforme de manière optimale.

1.3.2 Accès direct aux données protégées

Comme vous l'avez sans doute remarqué, certaines de nos données ne sont utilisables que si vous disposez d'un compte vous y autorisant. Dès lors que vous implémentez ces informations dans votre code, il vous appartient de veiller à leur sécurité, leur non-divulgation et leur facilité d'emploi tout au long de leur cycle de vie (mise à jour, suppression...)

En Javascript/Ajax

Une des particularité du Javascript est d'être chargé et de s'exécuter côté client. Si vous intégrez vos identifiants dans votre code javascript, ils seront donc forcément accessibles à l'utilisateur de votre site. Employer ruses et contournements ne fera que rendre plus difficile leur récupération, mais à coup sûr ils apparaîtront tôt ou tard dans les requêtes que vous allez émettre vers nos services. Pour tout développement en Javascript/Ajax, il faut donc utiliser les technique de Proxyfication des requêtes décrites plus bas.

En PHP, Python, Java

Pas de problèmes avec ces languages qui s'exécutent côté serveur. Il suffit d'implémenter les *Exemples et extraits de code* proposés.

Sur plateforme mobile

Les développements effectués sur plateforme mobile sont relativement sûrs dans la mesure où ils sont compilés et sont donc distribués à vos utilisateurs sous une forme qui rend extrêmement difficile la récupération des informations qui s'y trouvent. Par contre, il vous faut considérer un aspect pratique lié au cycle de vie de vos identifiant. Pour peu que vous changiez de mot de passe, ou même de compte, comment va se passer la diffusion du nouveau couple login/mot de passe sur les terminaux de vos utilisateurs ? Devront-ils faire une mise à jour ? Pourrez-vous provoquer cette mise à jour obligatoire ? Cette réflexion devra avoir été menée avant de diffuser votre solution.

Performances

Une autre question importante est celle des performances de notre plateforme. Elle n'est pas dimensionnée pour tenir la même charge que www.google.fr. Si votre application a du succès, si un tweet la propulse vers des sommets de popularité très rapidement, êtes vous sûrs que nos infrastructures tiendront la charge et permettront à vos utilisateurs d'avoir une bonne expérience avec votre application ?

Conclusion

Pour chacune des situations ci-dessus, il vous faut veiller à la mise en oeuvre des pratiques conseillées. Le plus simple est parfois de récupérer directement la donnée.

1.3.3 Récupération des données

Vos droits d'accès vous permettent d'accéder à la donnée et de la stocker sur vos serveurs. Leur fréquence de récupération doit être adaptée à leur fréquence de mise à jour. Une fois intégrée à votre infrastructure, la résolution des problèmes évoqués précédemment est beaucoup plus simple :

– Sur plateforme mobile, les utilisateurs accèdent directement à votre version des données. Si vos login/mot de passe change, cela affecte votre script de récupération des données uniquement et non l'application déployée chez les clients, qui de son côté peut utiliser un système d'authentification/autorisation qui vous est propre.
– Pour les performance : la récupération ponctuelle, même si elle est fréquente, affecte peu les performances de notre plateforme. Pour peu que votre infrastructure soit dimensionnée pour absorber la charge générée par votre application, vous n'aurez pas de problème à craindre de notre côté en vous y prenant ainsi.

1.3.4 Proxyfication

Que les données soient installées au sein de votre infrastructure ou que votre application accède directement aux données sur la nôtre, il est des cas d'utilisation ou le recours à une proxyfication va néanmoins être indispensable. C'est surtout le cas en environnement Javascript avec lequel vous avez deux obstacles :

– L'impossiblité de cacher les login/mot de passe à transmettre pour accéder aux données. Vous pouvez néanmoins avoir mis en oeuvre, après avoir récupéré les données, un mécanisme d'authentification qui est propre à vos utilisateurs, auquel cas vous réglez une partie du problème
– L'accès à un domaine tiers en Ajax. Ceci est strictement interdit par "Same Origine Policy" qui impose au contenu appelé par une requête Ajax d'être situé sur le même domaine et le même port que la page principale dans laquelle se situe la requête. Cette règle ne vaut pas pour les images, ce qui explique que l'on puisse utiliser les flux WMS externes directement, mais vaut pour tous nos autres types de services : WFS, JSON ou KML.

Pour contourner ces restrictions, vous pouvez employer l'astuce de la proxyfication. Le principe est de transmettre les requêtes AJAX à un script situé sur votre propre serveur (donc mêmes domaine et port que la page principale), qui transmettra les requêtes au service externe, sans être soumis aux mêmes contraintes d'un client web. Cela poeut également vous servir pour distinguer les comptes d'accès à votre serveur de celui utilisé pour vous connecter à notre infrastructure.

CLIENT WEB ——> SCRIPT PROXY ——> SERVICE EXTERNE
user :toto.....user :my_account.....>

Implémenter un script de proxyfication n'est pas très compliqué, mais il faut respecter certaines règles. Ne pas le laisser ouvert à tous vents par exemple, car sinon n'importe qui pourrait venir s'en servir pour naviguer sur internet de manière anonyme. Ne pas le laisser faire n'importe quoi non plus, afin qu'un utilisateur ne puisse pas envoyer n'importe quelle requête à votre serveur.

Implémentation en Python

Le script que nous allons utiliser ici est fourni avec OpenLayers sous le nom de proxy.cgi.

Il s'agit d'un script écrit en Python, parfaitement adapté au WFS. Pour l'utiliser, il faut le déposer dans un répertoire exécutable du serveur web (généralement le cgi-bin), en régler les droits pour le rendre exécutable (chmod a+x proxy.cgi par exemple) et indiquer à OpenLayers sa présence pour qu'il l'utilise (car OpenLayers est malin mais pas devin...) avec la directive :

```
OpenLayers.ProxyHost = "/cgi-bin/proxy.cgi?url=";
```

Pour toute information complémentaire concernant OpenLayers et un Proxy, veuillez vous référer à la page de FAQ http ://trac.osgeo.org/openlayers/wiki/FrequentlyAskedQuestions#ProxyHost

```
#!/usr/local/bin/python


"""This is a blind proxy that we use to get around browser
restrictions that prevent the Javascript from loading pages not on the
same server as the Javascript.  This has several problems: it's less
efficient, it might break some sites, and it's a security risk because
people can use this proxy to browse the web and possibly do bad stuff
with it.  It only loads pages via http and https, but it can load any
content type. It supports GET and POST requests."""

import urllib2
import cgi
import sys, os

# Designed to prevent Open Proxy type stuff.
# replace 'my_target_server' by the external domain you are aiming to
allowedHosts = ['localhost','my_target_server']

method = os.environ["REQUEST_METHOD"]

if method == "POST":
    qs = os.environ["QUERY_STRING"]
    d = cgi.parse_qs(qs)

        # checks if a url parameter exists in the POST request. If not, go to hell.
    if d.has_key("url"):
        url = d["url"][0]
    else:
        url = "http://www.openlayers.org"
else:
    fs = cgi.FieldStorage()
        # checks if a url parameter exists in the GET request. If not, go to hell.
    url = fs.getvalue('url', "http://www.openlayers.org")

try:
```

```
    host = url.split("/")[2]

        # reply with HTTP 502 code if the host is not allowed
    if allowedHosts and not host in allowedHosts:
        print "Status: 502 Bad Gateway"
        print "Content-Type: text/plain"
        print
        print "This proxy does not allow you to access that location (%s)." % (host,)
        print
        print os.environ
    # checks if the request is a http or https request
    elif url.startswith("http://") or url.startswith("https://"):

        if method == "POST":
            length = int(os.environ["CONTENT_LENGTH"])
            headers = {"Content-Type": os.environ["CONTENT_TYPE"]}
            body = sys.stdin.read(length)
            r = urllib2.Request(url, body, headers)
            y = urllib2.urlopen(r)
        else:
            y = urllib2.urlopen(url)

        # print content type header
        i = y.info()
        if i.has_key("Content-Type"):
            print "Content-Type: %s" % (i["Content-Type"])
        else:
            print "Content-Type: text/plain"
        print

        print y.read()

        y.close()
    else:
        print "Content-Type: text/plain"
        print
        print "Illegal request."

except Exception, E:
    print "Status: 500 Unexpected Error"
    print "Content-Type: text/plain"
    print
    print "Some unexpected error occurred. Error text was:", E
```

Ce script PHP fait la même chose :

```
<?php
            /*
            License: LGPL as per: http://www.gnu.org/copyleft/lesser.html
            $Id: proxy.php 3650 2007-11-28 00:26:06Z rdewit $
            $Name$
            */

            ////////////////////////////////////////////////////////////////////////////
            // Description:
            // Script to redirect the request http://host/proxy.php?url=http://someUrl
            // to http://someUrl .
            //
```

```
// This script can be used to circumvent javascript's security requirements
// which prevent a URL from an external web site being called.
//
// Author: Nedjo Rogers
////////////////////////////////////////////////////////////////////////////////

// define alowed hosts
$aAllowedDomains = array('localhost','my_target_server')

// read in the variables

if(array_key_exists('HTTP_SERVERURL', $_SERVER)){
        $onlineresource=$_SERVER['HTTP_SERVERURL'];
}else{
        $onlineresource=$_REQUEST['url'];
}
$parsed = parse_url($onlineresource);
$host = @$parsed["host"];
$path = @$parsed["path"] . "?" . @$parsed["query"];
if(empty($host)) {
        $host = "localhost";
}

if(is_array($aAllowedDomains)) {
        if(!in_array($host, $aAllowedDomains)) {
                die("le domaine '$host' n'est pas autorisé. contactez l'administrateur.")
        }
}

$port = @$parsed['port'];
if(empty($port)){
        $port="80";
}
$contenttype = @$_REQUEST['contenttype'];
if(empty($contenttype)) {
        $contenttype = "text/html; charset=ISO-8859-1";
}
$data = @$GLOBALS["HTTP_RAW_POST_DATA"];
// define content type
header("Content-type: " . $contenttype);

if(empty($data)) {
        $result = send_request();
}
else {
        // post XML
        $posting = new HTTP_Client($host, $port, $data);
        $posting->set_path($path);
        echo $result = $posting->send_request();
}

// strip leading text from result and output result
$len=strlen($result);
$pos = strpos($result, "<");
if($pos > 1) {
        $result = substr($result, $pos, $len);
}
//$result = str_replace("xlink:","",$result);
```

```
echo $result;

// define class with functions to open socket and post XML
// from http://www.phpbuilder.com/annotate/message.php3?id=1013274 by Richard Hundt

class HTTP_Client {
        var $host;
        var $path;
        var $port;
        var $data;
        var $socket;
        var $errno;
        var $errstr;
        var $timeout;
        var $buf;
        var $result;
        var $agent_name = "MyAgent";
        //Constructor, timeout 30s
        function HTTP_Client($host, $port, $data, $timeout = 30) {
                $this->host = $host;
                $this->port = $port;
                $this->data = $data;
                $this->timeout = $timeout;
        }

        //Opens a connection
        function connect() {
                $this->socket = fsockopen($this->host,
                $this->port,
                $this->errno,
                $this->errstr,
                $this->timeout
        );
        if(!$this->socket)
                return false;
        else
                return true;
        }

        //Set the path
        function set_path($path) {
                $this->path = $path;
        }

        //Send request and clean up
        function send_request() {
                if(!$this->connect()) {
                        return false;
                }
                else {
                        $this->result = $this->request($this->data);
                        return $this->result;
                }
        }

        function request($data) {
                $this->buf = "";
                fwrite($this->socket,
```

```
                        "POST $this->path HTTP/1.0\r\n".
                        "Host:$this->host\r\n".
                        "Basic: ".base64_encode("guillaume:catch22")."\r\n".
                        "User-Agent: $this->agent_name\r\n".
                        "Content-Type: application/xml\r\n".
                        "Content-Length: ".strlen($data).
                        "\r\n".
                        "\r\n".$data.
                        "\r\n"
                );

                while(!feof($this->socket))
                        $this->buf .= fgets($this->socket, 2048);
                        $this->close();
                        return $this->buf;
                }


                function close() {
                        fclose($this->socket);
                }
        }



        function send_request() {
                global $onlineresource;
                $ch = curl_init();
                $timeout = 5; // set to zero for no timeout

                // fix to allow HTTPS connections with incorrect certificates
                curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
                curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 1);

                //curl_setopt($ch, CURLOPT_USERPWD, 'guillaume:catch22');
                //curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);

                curl_setopt($ch, CURLOPT_URL,$onlineresource);
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
                curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
                curl_setopt($ch, CURLOPT_ENCODING , "gzip, deflate");

                if( ! $file_contents = curl_exec($ch)){
                        trigger_error(curl_error($ch));
                }
                curl_close($ch);
                $lines = array();
                $lines = explode("\n", $file_contents);
                if(!($response = $lines)) {
                        echo "Unable to retrieve file '$service_request'";
                }
                $response = implode("",$response);
                return utf8_decode($response);
        }
?>
```

1.3.5 Synthèse

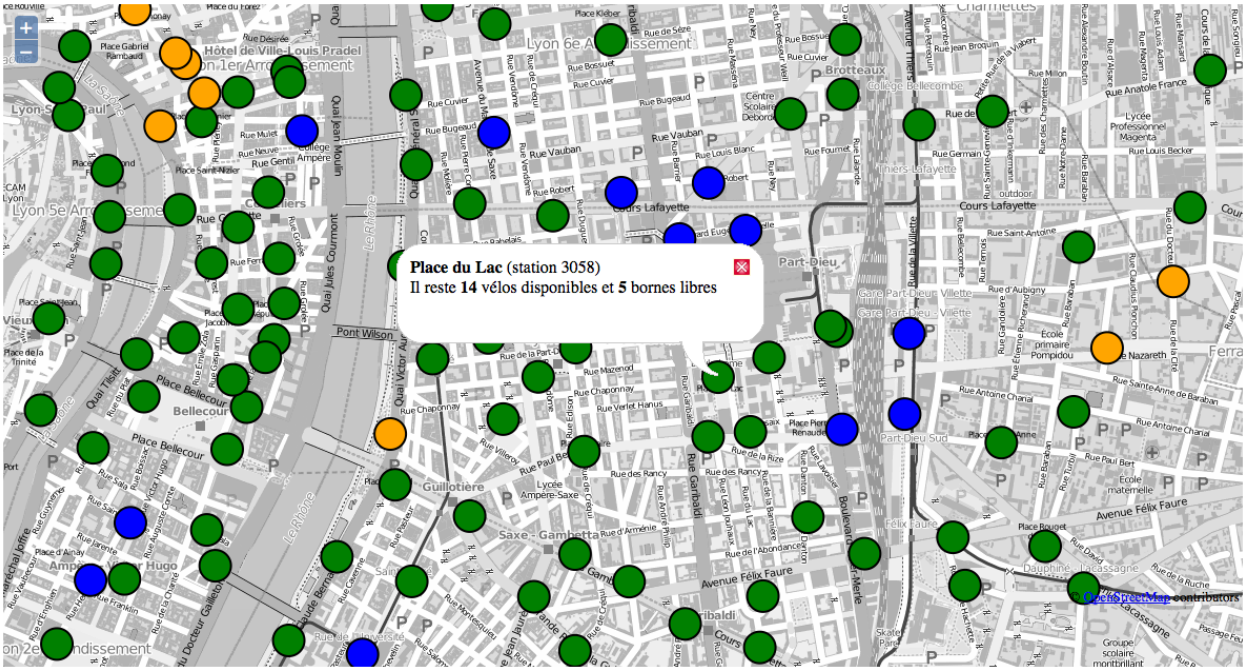
Comme nous l'avons vu, il y a différentes stratégies à utiliser selon les flux que vous utilisez et la plateforme pour laquelle vous développez. Faire du WMS dans une application web sera plus simple que traiter du WFS volumineux dans une application iPhone. On peut cependant distinguer les approches les plus intéressantes :

– Pour de l'image simple, sans authentification, utilisez un flux direct vers notre plateforme
– pour les gros volumes texte (WFS, JSON...) récupérez la donnée à intervalles réguliers et servez là depuis votre serveur. Ca peut aussi vous éviter le recours à un script proxy.
– Pour les applications nomades sur Smartphone, privilégiez l'autonomie de l'application par rapport aux modalités d'accès aux données. Récupérez les données, et implémentez un service listant les données disponibles, de sorte que vous pourrez intégrer de nouvelles couches de données à votre application sans la mettre à jour.

1.4 Exemples et extraits de code

1.4.1 WMS avec OpenLayers

Cet exemple montre l'utilisation du service WFS avec les bornes vélo'v en temps réel à travers la bibliothèque OpenLayers.



Code source correspondant :

```
<html>
  <head>
    <title>Utilisation des services GrandLyon Smart Data : OpenLayers</title>
    <script src="http://openlayers.org/api/OpenLayers.js"></script>
  </head>
    <body>
      <div style="width:100%; height:100%" id="map"></div>
      <script defer="defer" type="text/javascript">
        var map = new OpenLayers.Map('map');
        var osm = new OpenLayers.Layer.OSM('Simple OSM Map', null, {
          //conversion en valeurs de gris à la volée
```

```
  eventListeners: {
    tileloaded: function(evt) {
      var ctx = evt.tile.getCanvasContext();
      if (ctx) {
        var imgd = ctx.getImageData(0, 0, evt.tile.size.w, evt.tile.size.h);
        var pix = imgd.data;
        for (var i = 0, n = pix.length; i < n; i += 4) {
          pix[i] = pix[i + 1] = pix[i + 2] = (3 * pix[i] + 4 * pix[i + 1] + pix[i + 2]) / 8;
        }
        ctx.putImageData(imgd, 0, 0);
        evt.tile.imgDiv.removeAttribute("crossorigin");
        evt.tile.imgDiv.src = ctx.canvas.toDataURL();
      }
    }
  }
});
//URL du service smartdata
var smartdata_url = "https://download.data.grandlyon.com/wfs/smartdata?";
//Définition du proxy pour gérer le cross-domain. Voir le paragraphe Bonne Pratiques -> Proxy
OpenLayers.ProxyHost = "/cgi-bin/proxy.cgi?url=";

//Styles pour le rendu
var colors = ["green", "blue", "orange", "grey"];
var context = {
    getColor: function(feature) {
        return colors[feature.data.availabilitycode - 1];
    }
}
var template = {
  pointRadius: 15,
  fillColor: "${getColor}" // using context.getColor(feature)
};
var style = new OpenLayers.Style(template, {context: context});

//Définition du layer WFS
var wfs = new OpenLayers.Layer.Vector("WFS Smartdata", {
    strategies: [new OpenLayers.Strategy.BBOX()],
    protocol: new OpenLayers.Protocol.WFS({
        version: "1.1.0",
        srsName: "EPSG:4326",
        url: smartdata_url,
        featurePrefix : 'ms',
        featureType: "jcd_jcdecaux.jcdvelov",
        geometryName: "msGeometry",
        formatOptions: {
          xy: false
        }
    }),
    styleMap: new OpenLayers.StyleMap(style),
    renderers: OpenLayers.Layer.Vector.prototype.renderers
});

//gestion du click sur les markers
var selectControl = new OpenLayers.Control.SelectFeature(wfs);
map.addControl(selectControl);
selectControl.activate();

wfs.events.on({
```

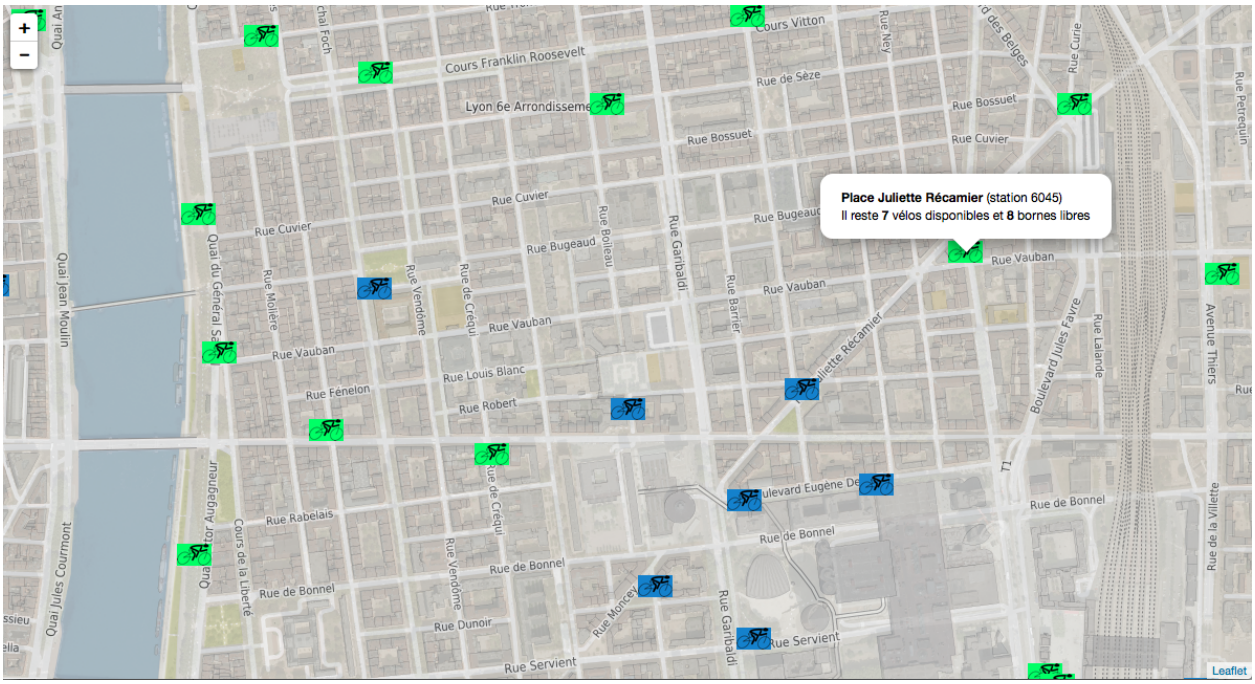
```
            featureselected: function(event) {
              var feature = event.feature;
              feature.popup = new OpenLayers.Popup.FramedCloud("box",
                  feature.geometry.getBounds().getCenterLonLat(),
                  null,
                  '<div><b>'+feature.data.name + '</b> (station '+feature.data.number+')<br/>'
                  + 'Il reste <b>' + feature.data.available_bikes + '</b> v&eacute;los disponibles et '
                  + '<b>' + feature.data.available_bike_stands + ' </b>bornes libres</div>',
                  null,
                  true
              );
              while( map.popups.length ) {
                  map.removePopup( map.popups[0] );
              }
              map.addPopup(feature.popup);
              }
          });

          //Config de la map
          map.addLayers([osm, wfs]);
          var zoom = 15;
          var lonLat = new OpenLayers.LonLat(4.85,45.76);
          map.setCenter(
              lonLat.transform(
                  new OpenLayers.Projection("EPSG:4326"),
                  map.getProjectionObject()
              ), zoom
          );

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

1.4.2 WFS avec Leaflet

Cet exemple montre l'utilisation du service WFS avec les bornes vélo'v en temps réel à travers la bibliothèque LeafLet.



Code source correspondant :

```
<html>
  <head>
    <title>Utilisation des services GrandLyon Smart Data : Leaflet</title>
    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <script src="leaflet.js"></script>
    <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>

    <link rel="stylesheet" href="leaflet.css" />
    <style>
        body {
                padding: 0;
                margin: 0;
        }
        html, body, #map {
                height: 100%;
        }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
        //Initialisation de la map
        var map = L.map('map').setView([45.76, 4.85], 14);
        //Layer WMS sur une orthophoto
        L.tileLayer.wms("https://download.data.grandlyon.com/wms/grandlyon",{
                layers: '1840_5175_16_CC46',
                format: 'image/png',
                transparent: true,
                opacity: 0.6
        }).addTo(map);
```

```
//Layer WMS openstreetmap
L.tileLayer.wms("http://openstreetmap.wms.data.grandlyon.com/default",{
        layers: 'default',
        format: 'image/png',
        transparent: true,
        opacity: 0.7
}).addTo(map);

//Définition du proxy pour le WFS (cross domain). Voir le paragraphe Bonne Pratiques -> Proxy
var proxy = "proxy.php?url=";
var smartdata_url = "https://secure.grandlyon.webmapping.fr/wfs/smartdata";
var params = '?SERVICE=WFS
    &REQUEST=GetFeature
    &VERSION=1.1.0
    &TYPENAME=jcd_jcdecaux.jcdvelov
    &outputformat=geojson';

var VertIcon = L.icon({
    iconUrl: 'images/cycling_Vert.png',
    iconSize:     [33, 21]
});
var OrangeIcon = L.icon({
    iconUrl: 'images/cycling_Orange.png',
    iconSize:     [33, 21]
});
var BleuIcon = L.icon({
    iconUrl: 'images/cycling_Bleu.png',
    iconSize:     [33, 21]
});
var GrisIcon = L.icon({
    iconUrl: 'images/cycling_Gris.png',
    iconSize:     [33, 21]
});

$.get(proxy + encodeURIComponent(smartdata_url + params), function(json){
    var obj = $.parseJSON(json);
    // Add markers
    for(i=0;i<obj.features.length;i++) {
        //create feature from json
        var ftr = obj.features[i];
        // set marker options from properties
        var options = {
            gid: ftr.properties.gid,
            number: ftr.properties.number,
            name: ftr.properties.name,
            available_bikes: ftr.properties.available_bikes,
            available_bike_stands: ftr.properties.available_bike_stands
        };
        //set marker icon from availability
        switch(ftr.properties.availability){
            case 'Vert':
                options.icon = VertIcon;
                break;
            case 'Orange':
                options.icon = OrangeIcon;
                break;
            case 'Bleu' :
                options.icon = BleuIcon;
```

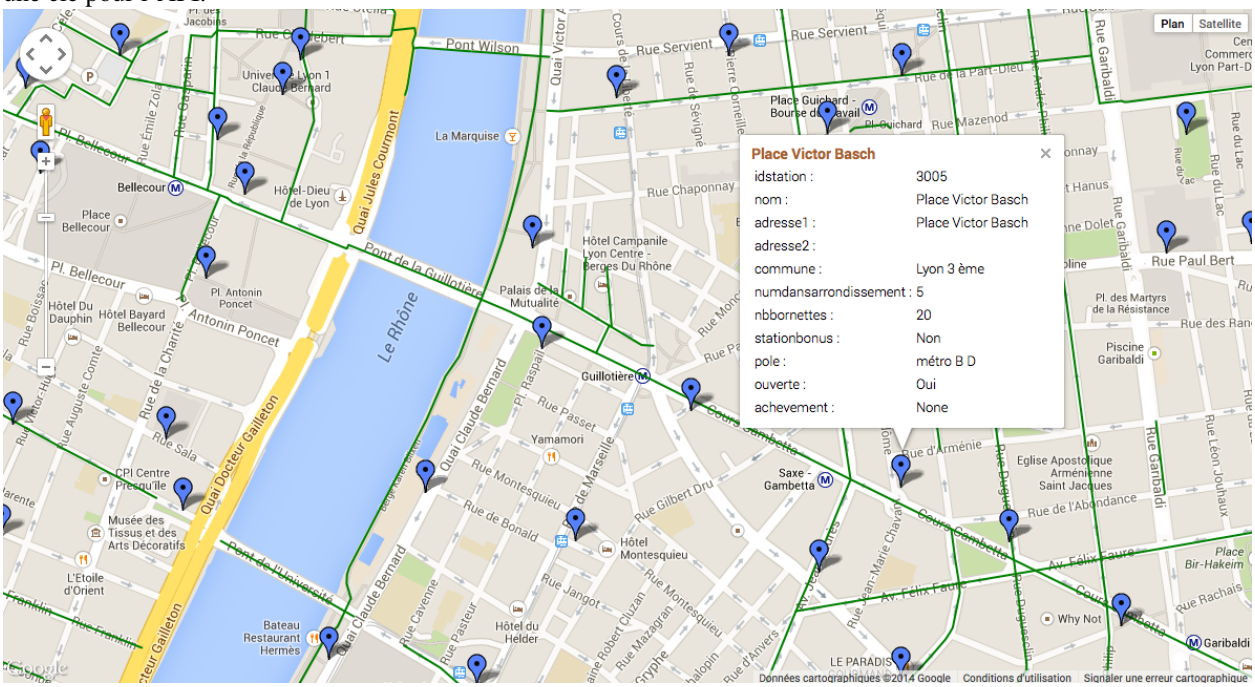
```
                    break;
                default :
                    options.icon = GrisIcon;
            }
            //ajout du marker à la map
            var point = L.marker(
                [ftr.geometry.coordinates[1],ftr.geometry.coordinates[0]],
                options
            ).addTo(map);
            //définition de la popup sur le click
            point.bindPopup(
                '<b>'+ point.options.name + '</b> (station '+point.options.number+')<br/>'
                + 'Il reste <b>' + point.options.available_bikes + '</b> v&eacute;los disponibles
                + ' et <b>' + point.options.available_bike_stands + ' </b>bornes libres',
                {
                closeButton: false
                }
            );

        }
    });

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

1.4.3 KML avec l'API Maps de Google

Cet exemple montre l'utilisation du service KML avec les bornes vélo'v à travers l'API Google Maps v3. Nécessite une clé pour l'API.



Code source correspondant :

```
<html>
  <head>
    <title>Utilisation des services GrandLyon Smart Data : Google API</title>
    <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
    <style type="text/css">
      html { height: 100% }
      body { height: 100%; margin: 0; padding: 0 }
      #map-canvas { height: 100% }
    </style>
    <script type="text/javascript"
        src="https://maps.googleapis.com/maps/api/js?key=API_KEY&sensor=false">
    </script>

    <script type="text/javascript">
      function initialize() {
        //Init map
        var mapOptions = {
          center: new google.maps.LatLng(45.76, 4.85),
          zoom: 13
        };
        var map = new google.maps.Map(document.getElementById("map-canvas"),
            mapOptions);

        //Ajout WMS sur l'aménagement cyclable
        var urlWMS = "https://download.data.grandlyon.com/wms/grandlyon?"
                + "&REQUEST=GetMap&SERVICE=WMS&VERSION=1.3.0&CRS=EPSG:4171"
                + "&LAYERS=pvo_patrimoine_voirie.pvoamenagementcyclable"
                + "&FORMAT=image/png&TRANSPARENT=TRUE&WIDTH=256&HEIGHT=256";
        //L'API Google Map ne gère pas directement le WMS, il faut passer par un ImageMapType
        var WMS_Layer = new google.maps.ImageMapType({
            getTileUrl: function (coord, zoom) {
                var projection = map.getProjection();
                var zoomfactor = Math.pow(2, zoom);
                var LL_upperleft = projection.fromPointToLatLng(
                    new google.maps.Point(
                        coord.x * 256 / zoomfactor,
                        coord.y * 256 / zoomfactor
                    )
                );
                var LL_lowerRight = projection.fromPointToLatLng(
                    new google.maps.Point(
                        (coord.x + 1) * 256 / zoomfactor,
                        (coord.y + 1) * 256 / zoomfactor
                    )
                );
                var bbox =  "&bbox="
                    + LL_lowerRight.lat() + "," + LL_upperleft.lng() + ","
                    + LL_upperleft.lat() + "," + LL_lowerRight.lng();
                var url = urlWMS + bbox;
                return url;
            },
            tileSize: new google.maps.Size(256, 256),
            isPng: true
        });

        map.overlayMapTypes.push(WMS_Layer);

        //Ajout KML layer
```

```
        var KML_Layer = new google.maps.KmlLayer({
          url: 'https://download.data.grandlyon.com/kml/grandlyon/?'
            +'request=layer&typename=pvo_patrimoine_voirie.pvostationvelov'
        });
        KML_Layer.setMap(map);

      }
      google.maps.event.addDomListener(window, 'load', initialize);
    </script>

  </head>
  <body>
    <div id="map-canvas"/>
  </body>
</html>
```