OSGEO PRIMER
Workshop Open Source GIS & WebCartography

Barend Köbben

Version 1.1
April 20, 2012

Contents

Key points

The ICA Working Group on Open Source Geospatial Technologies runs Open Source GIS and WebCartography sessions at various events around the world. Everyone is invited to our sessions to learn about Open Source tools for GIS and mapping on the Web. We'll start with a short introduction, and then show you hands-on how to actually do it: So bring your laptop, we will provide the theory, the software and guidance! Access to the presentations, demo and workshop materials will be at the workshop website http://kartoweb.itc.nl/kobben/AGILE-OSGEO-Workshop/.

! → In some cases you have to enter Javascript or other code, or a Web address,, indicated in this document. Sometimes these are quit long texts. You may use cut & paste from the exercise PDF, but be aware that from PDF sometimes a multi–line URL includes superfluous return (or newline) symbols that have to be removed. . . !

! → the character →means you **should not** type a `return` or `enter` in this place. The line should be typed **without interruption**, the move to the next line in our example is only because it would not fit otherwise.

1 The OSGEO LiveDVD

In this workshop we will be using the free and open source GIS and webmapping applications from the OSGEO LiveDVD.

> The Open Source Geospatial Foundation, or OSGeo, is a not-for-profit organization whose mission is to support and promote the collaborative development of open geospatial technologies and data. The foundation provides financial, organizational and legal support to the broader open source geospatial community. OSGeo also serves as an outreach and advocacy organization for the open source geospatial community, and provides a common forum and shared infrastructure for improving cross-project collaboration. The foundation's projects are all freely available and useable under an OSI-certified open source license. More information at the website http://www.osgeo.org.

The OSGEO LiveDVD (the current version is OSGeo-Live 5.5) is a self-contained bootable DVD, USB drive or Virtual Machine based on the `Xubuntu` operating system, that allows you to try a wide variety of open source geospatial software without installing anything. It is composed

entirely of free software, allowing it to be freely distributed, duplicated and passed around.

It provides pre-configured applications for a range of geospatial use cases, including storage, publishing, viewing, analysis and manipulation of data. It also contains sample datasets and documentation.

Task 1 : To use the OSGEO LiveDVD, simply:

1. Insert the DVD or the USB stick in a computer or virtual machine.

2. Reboot the computer. In Microsoft Windows, you might have to verify or set the boot device order, to allow startup from a CD (mostly by pressing a key during start–up, typically the F–12 key). On Apple OSX, to boot from CD, you hold the C–key while restarting.

3. For CD booting: Wait for the "boot:" prompt, then press "Enter" to startup and login. For USB booting, you should see a menu where you choose the first option (start normally).

4. The Xubuntu OS should start up, and now you should have a desktop similar to the figure below (depending on the version of the LiveDVD you are using).

•

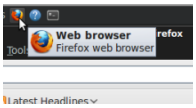
Many applications are also provided with installers for Apple OSX and Microsoft Windows. More information, including alternative disk images and installations, can be found on the website http://live.osgeo.org.

2 Open Data: using OpenStreetMap

First we'll show you a prime source of free maps and data on the web: OpenStreetMap.

> The OpenStreetMap Project, based at OpenStreetMap.org, is the worldwide mapping effort that includes more than 400,000 volunteers around the globe. OpenStreetMap is an initiative to create and provide free geographic data, such as street maps, to anyone.

There are many ways in which you can access the OpenStreetMap: as a simple webmapping service (not unlike Google and Bing Maps, but based on truly free non-proprietary data on a non-commercial website), as a webservice in various gis–viewers and as a database service, providing the actual vector data in raw form. We will try the first two methods, starting with simple web access on the OpenStreetMap site:



Task 2 : Open a webbrowser: Click on the FireFox icon in the menu bar (see left). The browser will start up. Open the URL http://www.openstreetmap.org/.

Browse the web map and try to find your conference location. •

3 Your own Open Data website: using OpenLayers

Using the OpenStreetMap site as described above is fine for casual map browsing, but if you want to disseminate spatial data yourself, you might want to include the OpenStreetMap in your own websites. And of course that is possible. We will use a popular browser–based mapping library for creating such a site: OpenLayers.

> OpenLayers (http://openlayers.org/) is a JavaScript Library based on AJAX-principles (Asynchronous JavaScript And XML). OpenLayers can be used to build general WebMapping clients. OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. It implements industry-standard methods for geographic data access, such as the OpenGIS Consortium's Web Mapping Service (WMS) and Web Feature Service (WFS) protocols, as well as existing webmap providers such as Google, Bing and OpenStreetMap. OpenLayers implements a JavaScript API for building rich web-based geographic applications, similar to the Google Maps,

with one important difference: OpenLayers is Free Software, developed for and by the Open Source software community.

OpenLayers is not a stand–alone program, it is a pure JavaScript library for displaying map data in most modern web browsers, with no server-side dependencies. In this workshop, we will only show the basic building blocks, and how to employ them. Using the Openlayers API is done by putting together *webpages* (using HTML) that include the necessary *JavaScript code* that calls the API, the *methods* from that API to make the necessary *map object* and connect that to an HTML *placeholder* and using the methods and properties of the API to set the content and behaviour of the map. In practice, this means typing (and/or copying) HTML and JavaScript code.

In listing 1 you see the simplest example of using OpenLayers for loading OpenStreetMap. Loading this page in a web browser will show the OpenStreetMap for the Porte Maillot area in Paris, France.

Listing 1: osm.html

```
<html xmlns="http://www.w3.org/1999/xhtml"><head>                    (x)html header
    <title>OpenLayers Basic Single OSM Example</title>
    <script src="http://localhost/openlayers/OpenLayers.js"></script>   include the API
    <script type="text/javascript">
        var myMap, myOSMLayer;                          define map and layer object
        var myCenter = new OpenLayers.LonLat(                        define center
       254031,6254016                                              at ICC location
    );
        function init() {
            myMap = new OpenLayers.Map( 'mapDiv');                  create map object
            myOSMLayer = new OpenLayers.Layer.OSM( "OSM Map");      create OSM layer
            myMap.addLayers([myOSMLayer]);                          add layer to map
    myMap.setCenter(myCenter,16);                                   zoom to center
        }
    </script>
  </head>
  <body onload="init()">
  <div id="mapDiv"                                                  map placeholder
     style="width:500px ; height400px; border:1px solid;"></div>
  </body></html>
```

Task 3 : You can type the code of listing 1 in an empty text file and save it as a `osm.html file`. But it is easier to copy the code from

the existing file we made for you, that is available at the website http://kartoweb.itc.nl/kobben/agile-osgeo-workshop/usb/.

View the result in the web browser. •

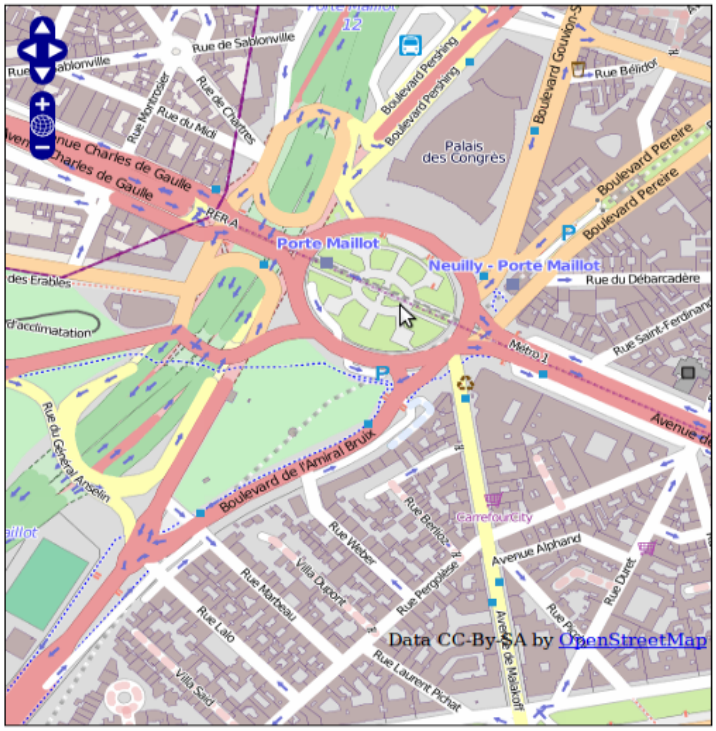


Figure 1: result of loading listing 1.

The result should look like figure 1. The icons you see in the map are the default Graphics User Interface (GUI) of OpenLayers. It offers the following interactivity:

- You can *pan* using the arrow icons, or by dragging the map;
- You can *zoom in* using the + icon, or shift–drag a zoom box in the map;
- You can *zoom out* using the – icon;
- You can *zoom to the full extent* using the globe icon.

You can set up the map to start at any place on the globe, by changing the coordinates that were used in the center command:

```
var myCenter = new OpenLayers.LonLat(254031,6254016);
```

But in order to find which coordinates to use to zoom to, for example, your house, it would be nice to have a knowledge of where (in coordinates) you are in the map. For that we will include a coordinate-readout line and a scale bar:

Task 4 : Add the following line in the script **before** the line with the `myMap.SetCenter` command:

```
myMap.addControl(new OpenLayers.Control.MousePosition());
myMap.addControl(new OpenLayers.Control.ScaleLine());
```

Save the results as `osmPlusCoordinates.html` (or use the pre–made version on the USB stick or the website). Try out the result in the browser.

Now you can zoom and pan to an alternative starting point for the map, e.g., you own hotel or home. Note down the coordinates and change the `myCenter` variable in the script to it, to have the map start at that place... •

The coordinates you see are X– and Y–coordinates in a Spherical Mercator projection on a WGS84 datum. It is the projection used by Google Maps (and is therefore often referred to as the "Google Mercator", with the EPSG code 900913). It is nowadays used by many commercial and open source web–mapping services, and the European Petroleum Standards Group has put a version of it in it's EPSG database of standardized projections (using EPSG code 3857).

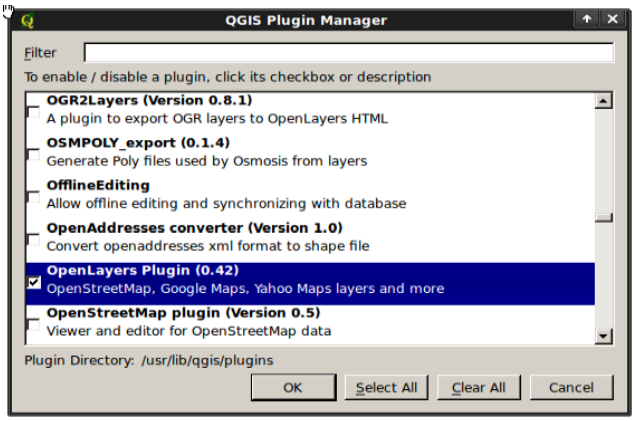
4 Open Source GIS: using QGIS

The web page we made in section 3 is nice, but what if we want to add our own data to that map? In order to do that, we shall first have to create such data. We will do that using the same OpenStreetMap as a reference and using QGIS, a free Open Source GIS.

> QGIS (officially QuantumGIS, `http://qgis.org`) is an Open Source, stand-alone GIS client, programmed in C++ using the multi-platform Qt framework. You can use it to work with vector– and raster–files, databases or any open standard WMS or WFS–compliant server. A strong point of QGIS is its extensibility: you can add plug–ins that are written in either C++ or Python, and you can connect it to GRASS, a powerful GIS analysis tool.

Task 5 :

Start QGIS by going to the menu `Geospatial > Desktop GIS > Quantum→ GIS`. To enable the use of OpenStreetMap, we will enable the plug–in that offers that functionality: Open the menu `Plugins > Manage →Plugins...` From the list, find the one called "OpenLayers plugin" and



enable it by making sure it is selected. Press OK.

Now open the menu `Plugins` again. A new item called `OpenLayers` → `Plugin` should have been added. Choose `Add OpenStreetMap layer` from this submenu, and the OpenStreetMap will be opened, zoomed out on the whole world.

Navigate to the location you saved for webpage you made earlier (your own place or the Porte Maillot). •

QGIS is not limited to OpenStreetMap, it can load maps and data from a huge array of possible sources:

- OGC Open Standard WMS and WFS services;
- spatially-enabled database tables using PostGIS and SpatiaLite, by means of a 'live' connection to such databases;
- vector formats supported by the OGR library, including ESRI shapefiles, MapInfo, KML, GPX and GML;
- raster formats supported by the GDAL library, such as digital elevation models, aerial photography or satellite imagery;
- locations and mapsets from GRASS (an open source GIS);

It also allows you to create data in many of these formats, and that is was we are going to do next...

4.1 Using QGIS to create your own data

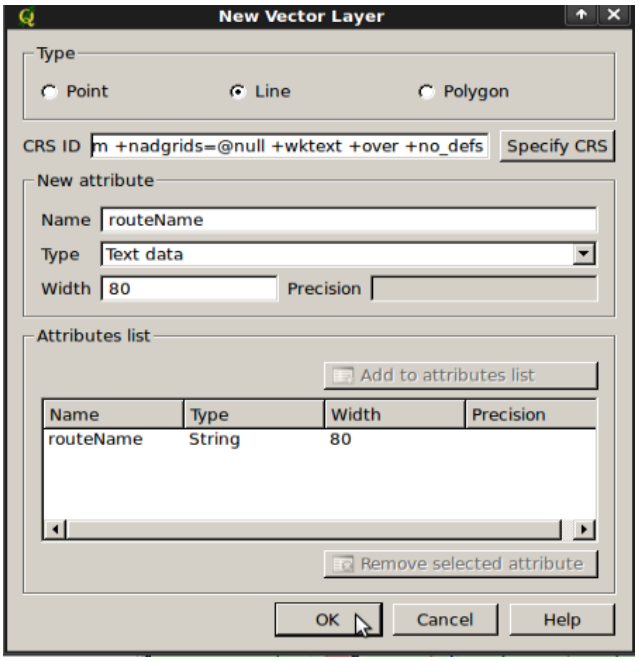
We will create a very simple vector line dataset, that depicts a walking or running route. You could for example try to digitize the road you took this morning from your hotel to the conference location...

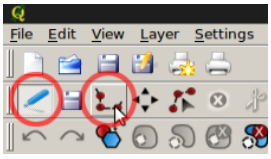
When creating new data, QGIS will adopt the projection of the current project. Therefore we must make sure that the data is saved correctly

projected, otherwise our new layer won't fit the OpenStreetMap base later on:

Task 6 : Choose the menu `Layer > New > New Shapefile Layer...` The New Layer dialog opens (see below). Make the following settings:

1. For `Type` choose `Line`;
2. Click `Specify CRS` and make sure you choose the Google Mercator (EPSG:900913). It can be found under `Projected Coordinate Systems > Mercator`;
3. In the `New attribute` section, create one attribute, named routeName, of type Text. Click the `Add to attribute list` button to actually add it;
4. Click OK to create the new file. Save it on your Hard Disk or the USB stick and name it "myRoutes.shp" [if this is not possible with your configuration, you will be have to later use the prepared file at the website].

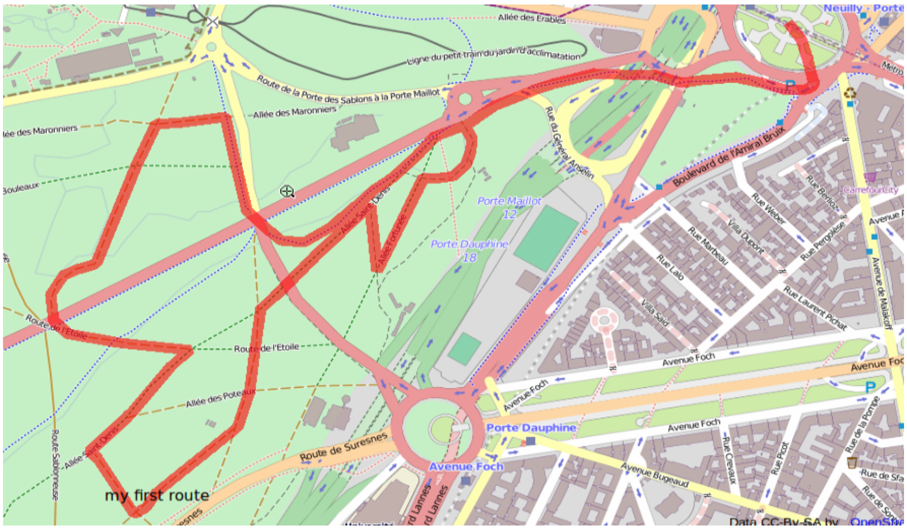




Now you can start adding lines for your route:

Click first the `Toggle Editing` button in the QGIS menubar, then the `Capture Line` button. Create a nice walking route, e.g. one in the Bois de Boulogne near the ICC conference centre. You can add points to the line by clicking in the map, undo them by using the `CTRL-Z` key or `Edit > Undo` menu. If you have finished a route, right-click, fill in the route name and press OK. Use the `Toggle Editing` button again to stop

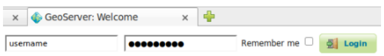
editing. You can change the visualisation of the line by right-clicking the layer name in the layers list, or choosing the `Layer > Properties` menu •



You now have created your own data. But only you can look at it, locally using your GIS viewer. In order to publish this in your web site, you'll have to turn the data into a *webservice*. We will do that creating an Open Standard WMS service, and we will use the GeoServer software to publish it.

5 Open Standards data dissemination: using GeoServer WMS

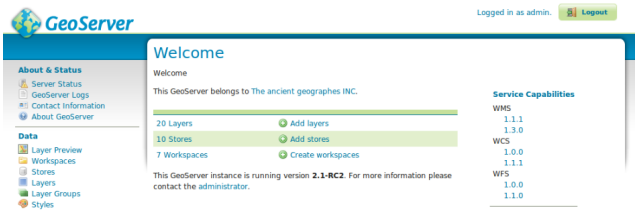
GeoServer is a service: That means that it acts as a background application, listening for requests on the web. You configure it using a series of web pages. It is installed on the OSGEO Live DVD, but the service has to be started up first in order to work:



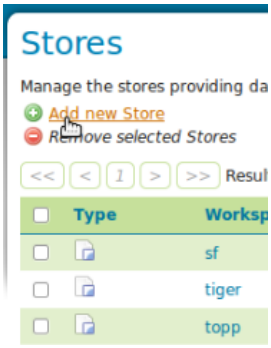
Task 7 : In the Xubuntu menu, choose `Geospatial > Web` → `Services > Start GeoServer`. The server starts up and a browser window is opened to show the connection to the service, at the URL `http://localhost:8082/geoserver/web`. This is the "public" interface, in order to set up the services you will have to login. Fill in the username ("admin") and password ("geoserver") and click the login button. The administrator pages are loaded… •

GeoServer (`http://openserver.org`) is an open source software server written in Java. Designed for interoperability, it publishes data from any major spatial data source using open

standards. Being a community-driven project, GeoServer is developed, tested, and supported by a diverse group of individuals and organizations from around the world. GeoServer is the reference implementation of the Open Geospatial Consortium (OGC) Web Feature Service (WFS) and Web Coverage Service (WCS) standards, as well as a high performance certified compliant Web Map Service (WMS).

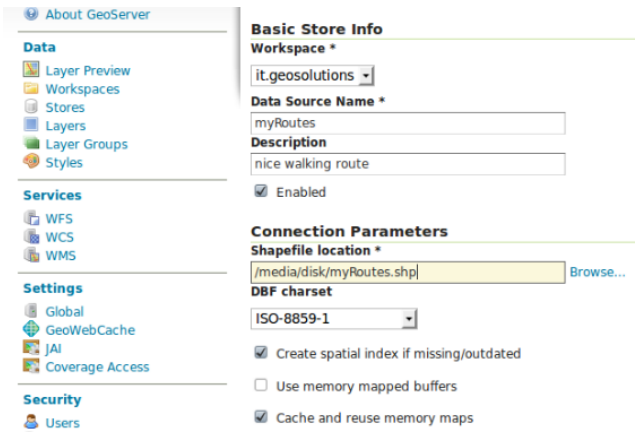


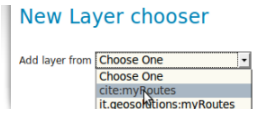
GeoServer uses a fairly elaborate setup: There are several `Workspaces`, that each can hold one or more `DataStores`. These connect the service to various datastores, either simple ones like vector– and raster–files, or more complicated ones like spatial databases or other OGC services. Each datastore can in turn contain one or more `Layers`.



Task 8 : We will use an existing `Workspace`, the default one called "it.geosolutions". In the `Data` menu, on the left side of the screen, click the `Stores` icon or name. Now click the `Add New Store` icon. From the list of datastores, choose the on called "Shapefile – ESRI(tm) Shapefiles (*.shp)".

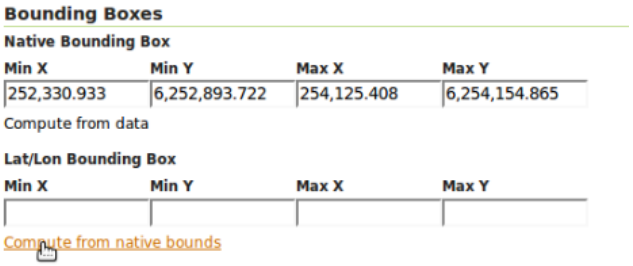
Now fill in the dialog: For the Shapefile location, you can use the `Browse...` button to navigate to the shape–file you created earlier in QGIS [alternatively, you can use the pre–made one at our website. Click the `Save` button.





Now click in the data menu (left side of screen) on the `Layers` item. In the next screen, choose the `Add new Resource` item. From the list choose it.geosolutions:myRoutes, and in the list of layers that is shown next click on the `Publish` link next to the `myRoutes` layers (should be the only one to choose from).

Now you can edit the Layer properties. Most fields should have been filled in correctly by default. Check if the projection is set correctly (to EPSG:900913). Now fill in both the Native Bounding Box and Lat/Lon Bounding Box by clicking the `Compute from data` links under them. Then click the `Save` button.



Now you can test the Layer publishing by choosing `Layer Preview` in the `Data` menu (left side of screen). There are many tests here, the most easy being to click on the `OpenLayers` link. •

Now your data is available for anyone on the internet that connects to one of the webservices that the GeoServer offers. That means you can also add it to the OpenLayers web site you made earlier, combined with the OpenStreetMap base layer. The last step is therefore to edit the web page to include a link to this service...

6 Combining Open Data and your own data in your own website

We will edit the web page we made earlier to now include the new WMS layer as an overlay on top of the OpenStreetMap base layer:

Task 9 : Open the file file `osm.html` in a text editor and save it as `osmPlusWms.html`.

First you'll have to add a variable that will hold the new WMS layer. Change the line that reads

```
var myMap, myOSMLayer;
```

into

```
var myMap, myOSMLayer, myWMSLayer;
```

Now we need to declare additional options for the map. The bounding box of the new routes layer and other variables so that OpenLayers knows the particulars of the projection, in order to fit the new WMS layer on the OpenStreetMap. Add this code right after the line you just edited:

```
var myBounds = new OpenLayers.Bounds(
     252330,6252893, 254125,6254154
     );
var options = {
  maxExtent: myBounds,
  maxResolution: 7,
  projection: "EPSG:900913",
  units: 'm'
  };
```

In order for these options to be used change the line that reads

```
  myMap = new OpenLayers.Map( 'mapDiv');
```

into

```
  myMap = new OpenLayers.Map( 'mapDiv', options);
```

The next step is to declare the new layer. Right after the line that reads

```
  myOSMLayer = new OpenLayers.Layer.OSM( "Simple OSM Map");
```

add the code:

```
myWMSLayer = new OpenLayers.Layer.WMS(
     "my routes",
     "http://localhost:8082/geoserver/wms",
     {layers: "it.geosolutions:myRoutes",
      transparent: "true",
      format: "image/png"}
   );
```

This code declares a layer of type WMS (the OGC Web Map Service), that will be requested from a service running at the web address "localhost:8082/geoserver.wms". It will be requested as a transparent PNG. Now we change the line that reads

```
  myMap.addLayers([myOSMLayer]);
```

into

```
  myMap.addLayers([myOSMLayer, myWMSLayer]);
```

in order add the new layer to the map object.

The final addition is to add the line

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

after the lines adding the coordinate readout and the scale line. This give us a LayerSwitcher we can use to switch the new layer on or off. Test the web page in the FireFox web browser. •

In the end, the webpage should contain the code you find in listing 2 below. You find the completed page on the website.

Listing 2: osmPlusWms.html

```
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<title>OpenLayers OSM + WMS</title>
<script src="http://localhost/openlayers/OpenLayers.js">
</script>
<script type="text/javascript">
  var myMap, myOSMLayer, myWMSLayer;
  var myCenter = new OpenLayers.LonLat(
        254031,6254016
        );
  var myBounds = new OpenLayers.Bounds(
        252330,6252893,
        254125,6254154
        );
  var options = {
      maxExtent: myBounds,
      maxResolution: 7,
      projection: "EPSG:900913",
      units: 'm'
      };
  function init(){
     myMap = new OpenLayers.Map( 'mapDiv',options);
     myOSMLayer = new OpenLayers.Layer.OSM( "Simple OSM Map");
     myWMSLayer = new OpenLayers.Layer.WMS(
      "my routes",
      "http://localhost:8082/geoserver/wms",
      {layers: "it.geosolutions:myRoutes",
       transparent: "true",
       format: "image/png"}
    );
    myMap.addLayers([myOSMLayer, myWMSLayer]);
    myMap.addControl(new OpenLayers.Control.MousePosition());
    myMap.addControl(new OpenLayers.Control.ScaleLine());
    myMap.addControl(new OpenLayers.Control.LayerSwitcher());
    myMap.setCenter(myCenter,16);
  }
</script>
</head>
<body onload="init()">
<div id="mapDiv"
style="width:500px ; height400px; border:1px solid;"></div>
</body></html>
```