# Beosmapper

OpenStreetMap Entrance Collection Application

Administration Guide & Technical Manual

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Authors** Aditya Raju

Anna Gorodetskaya

Kamyar Hasanzadeh

Marco Martins

Maa-123.3550 GIS Application Development – Aalto University

1. Components

Beosmapper is a web application running entirely on the client side, no data is stored in the server and the communication between the user and OSM is done directly. The application is built only with JavaScript using AngularJS JavaScript framework. The map functionality uses OpenLayers JavaScript web map library and OpenStreetMap API. More information about how the components are integrated is available in *2. Architecture*. Next, we describe briefly what each component is and which of their functionalities were use.

1.1 - AngularJS

AngularJS is an open-source Javascript framework, such as *Backbone.js*, that follows the Model-View-Controller (MVC) pattern. In this case the *view* is the HTML document, and the model and the controller are javascript files that are always in sync with the view using two-way data binding.

Address: <http://angularjs.org/>

1.2 - OpenLayers

OpenLayers is an open-source web mapping library. OpenLayers displays map layers in web pages and attaches functionality to the map such as event listeners, markers, geometry, etc. Beosmapper uses the OpenStreetMap layer, and other basic functionality like markers. We also use OpenLayers geolocation.

Address: [http://openlayers.org/](http://dev.openlayers.org/)

1.3 - OpenStreetMap APIs

Beosmapper accesses the OpenStreetMap database using the Overpass API and the OSM API v0.6. OSM database read operations are done using the Overpass API which is a read-only API optimized to query the OSM database. OSM database writes are done using the main OSM API. Both APIs use XML to insert and retrieve data.

OSM API v0.6 address: <https://wiki.openstreetmap.org/wiki/API_v0.6>

OSM Overpass API address: <http://wiki.openstreetmap.org/wiki/Overpass_API>

1.4 - Bootstrap

Bootstrap is a front-end framework. It can be used only as a CSS framework – for grid system, media queries, navigation, form styling, etc., or with Javascript components – for tabs, tooltips, etc. The application uses the base grid system, typography, forms, buttons, responsive tools, and the Alerts component. On top of that, uses the Tab and Collapse JavaScript components.

Address: <http://getbootstrap.com/>

1.5 - Other

* **XML to JSON** (xml2json.js): Converts XML into JSON and vice-versa, used to convert the XML response from OpenStreetMap data into JSON which AngularJS uses. Address: <http://www.thomasfrank.se/xml_to_json.html>.
* **angular-base64** (angular-base64.js): Adds Base 64 encoding/decoding functionality into AngularJS. This is used in the HTTP *Authorization* header required to authenticate the user in OpenStreetMap. Address: <https://github.com/ninjatronic/angular-base64>.
* **jQuery** (1.10): Javascript library required for Javascript components of Bootstrap framework. Address: <http://jquery.com/>.
* **JSON2** (json2.js): *JSON.stringify* polyfill for older browsers that don't support it. Address: <https://github.com/douglascrockford/JSON-js>.

2. Architecture

*Figure *1 *: Beosmapper architecture*

The application is composed of four main elements: *index.html*, *maincontroller.js*, *osmHandler.js* and *olHandler.js*.

2.1 – OpenStreetMap Handler (osmHandler.js)

The OpenStreetMap Handler connects AngularJS with Open Street Map, and it encapsulates every OSM operation in the application. The service has the following responsibilities:

* Retrieve features from OpenStreetMap database using Overpass API.
* Create an OpenStreetMap changeset.
* Add data to OpenStreetMap database.

2.2 – OpenLayers Handler: olHandler.js

The OpenLayers Handler integrates OpenLayers with AngularJS, and it encapsulates every OpenLayers function required by the application. The service has the following responsibilities:

* Initialize OpenStreetMap base layer.
* Set the event listeners.
* Enable/disable event listeners.
* Add new point features into the map.
* Clear all features in the map.
* Get user location – using geolocation.
* Create a buffer box from centered on specific coordinates.

The different service functionalities are called by the controller and it returns information in the form of callbacks or JavaScript promises.

2.3 – View: index.html

Application HTML with the AngularJS DOM extensions that dynamically bind with the controller, e.g. *ng-model*, *ng-click.* No functionality is present directly on this file, functionality only becomes set with the controller.

The OSM tags are only defined in the DOM, the application supports tags and sub-tags. For example, to set the OSM tag *entrance* the following attribute needs to added to the form input representing the tag value:

ng-model='entry.tags.entrance'

To add a sub-tag the tag key is named slightly differently than in OSM because the “:” character needs to be replaced with “\_”. The attribute for the sub-tag *addr:street* looks like this:

ng-model='entry.tags.addr\_street'

2.4 – Controller: maincontroller.js

The controller can be considered the core of the application. It defines the json data model, the data model binds with the view and is used in the application custom AngularJS services (OpenStreetMap Handler and OpenLayers Handler). The controller is responsible for:

* Initialization of the data model. The json data model includes the following fields:
  + Feature type: the feature type to query and add into OpenStreetMap. (Default value: entrance).
  + Tags: list of key value pairs with the tags values that describe the entry. (Default value: {attribution: 'Created with Beosmapper'}).
  + Description: entry description.
  + Location: feature location coordinates in WSG84.
  + Login: *username* and *password* key value pairs representing the user OSM password.
* OpenLayers initialization: OSM map, map event listeners.
* User Interface functionality: tabs, accordion and toggles.

AngularJS has a two-way data binding functionality, so when a value is updated in the view the model is automatically updated, and when the controller updates the model value the respective field in the view is updated – this way the model and the view are always synchronized.

3. Administration

As the application is entirely run on the client side and it doesn't use any internal database almost no administration is required.

3.1- Installation

The only requirement to install Beosmapper is a web server such as Apache or Nginx.

To install Beosmapper in your server you need to clone the project into the web server document folder you want to access Beosmapper from. You can do that with the following git command:

> git clone <https://github.com/coolers9/beosmapper.git>

By default the entrance entries are submitted into the OpenStreetMap test server. You can use the test server to test the functionality of the application, but don't forget the map will only display entries from the OSM production server and not from the testing. Also, you need to create a specific account for OSM test server, normal OSM credentials won't work.

To enable the submission into OSM production database. You need to edit the OSM service file:

{your\_path\_to\_beosmapper}/application/js/services/osmData.js

Then find the following lines in the file:

var osmHost = 'http://api06.dev.openstreetmap.org'; // Test OSM server

//var osmHost = 'http://openstreetmap.org'; // Production OSM server

Comment the first line above, and uncomment the second, it should look like this:

//var osmHost = 'http://api06.dev.openstreetmap.org'; // Test OSM server

var osmHost = 'http://openstreetmap.org'; // Production OSM server

3.3 - Removal

The only task required to remove the application is to remove its files from your server.

4. Course Feedback

The concept of the course is quite interesting and challenging. Especially when you can participate and develop a project that can be really useful for somebody. The practice of developing applications that can be potentially useful in the future for somebody should be continued and encouraged. The usage of the Scrum methodology and Github was appropriate and definitely helps the progress of the application and give some experience into real life tools.

The course attempts to provide a good insight into how programming and GIS can collaborate, however it is excessively biased towards the programming aspect and leaves GIS merely a theme for the course. A better balance between these two should be maintained and students should be provided with more flexibility in terms of the type of project they intend to work on. It would also the interesting to have the option for students to take an existing Github project and contribute to it with specific enhancements, this would make the course easier for more inexperience programmers to learn, as some of those projects have documentation available and the subject is narrowed down.

Overall an interesting course giving the students to develop and application fulfill a real need. The course should be balanced for the real programming experience of the majority of students.