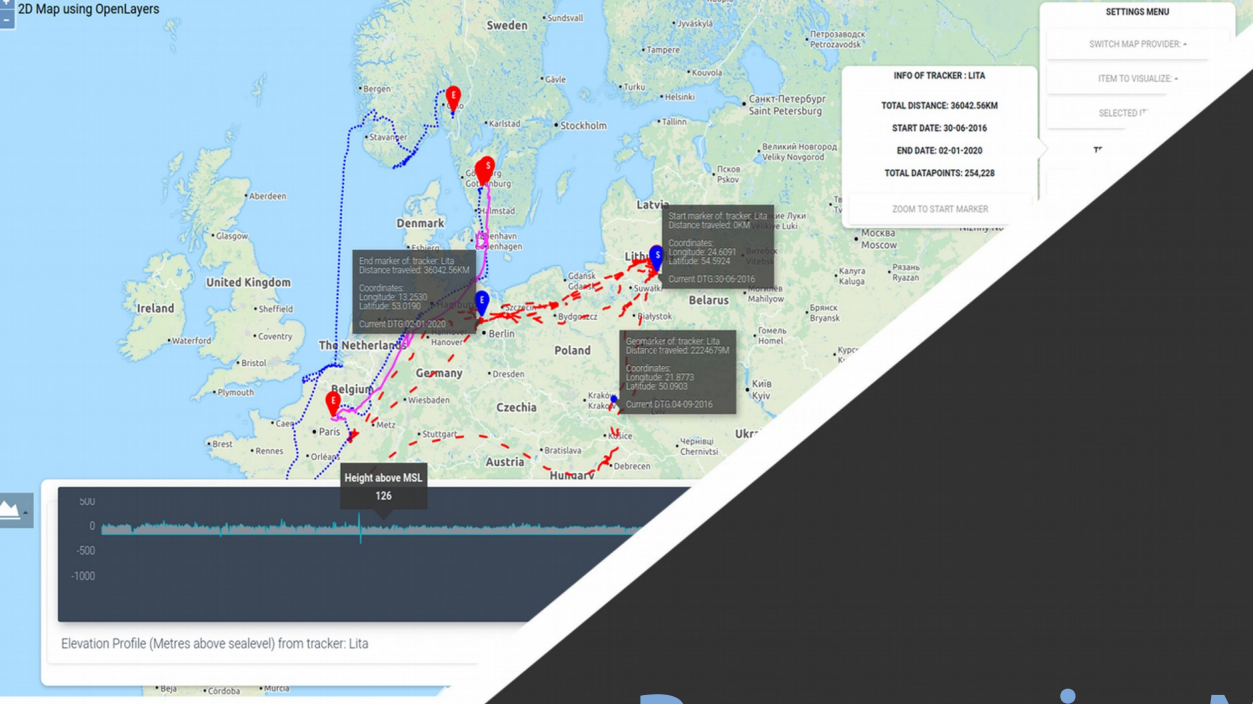
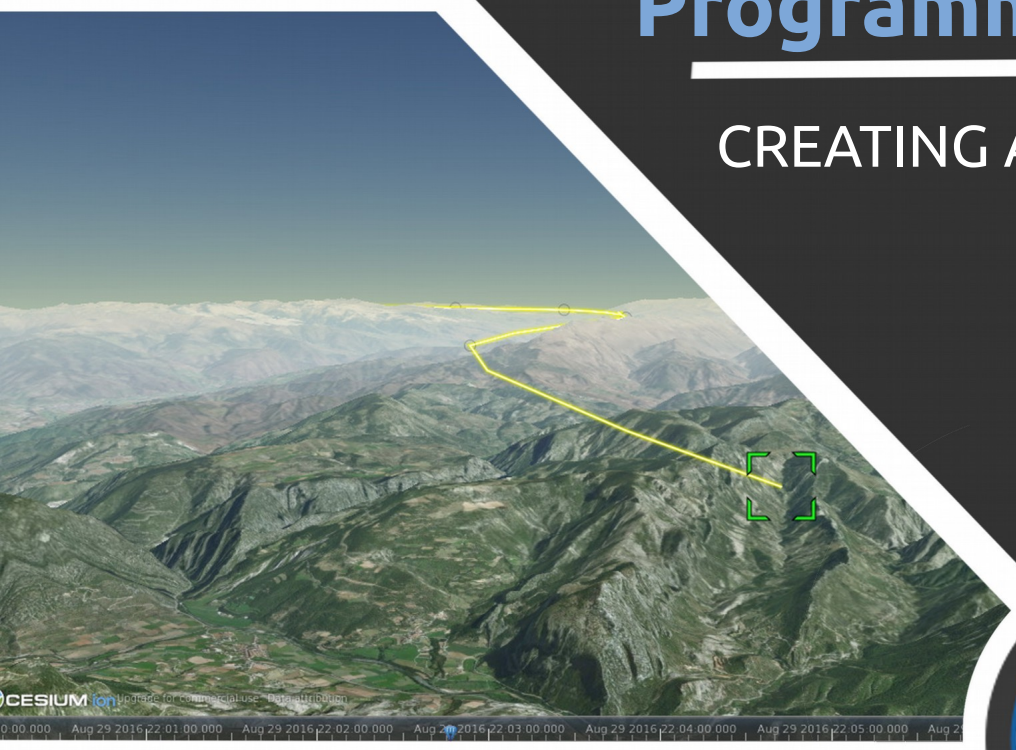
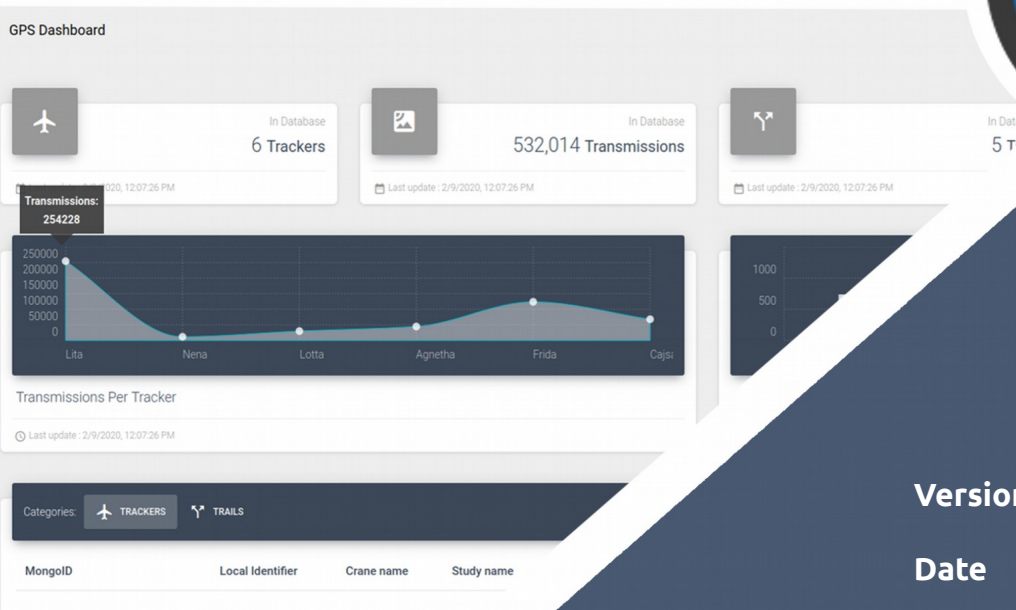


Programming Manual

CREATING AN 3 DIMENSIONAL MAP VIEWER





Version : 1.0

Date : 08-04-2020

Author : The GeoStack Project

Purpose of this document

This programming manual serves as an extension for the following documents:

1) **Cookbook: Creating the GeoStack Course VM:**

   The datastores, tools and libraries used during this programming manual are installed and created in the cookbook: Creating the GeoStack Course VM.

2) **Cookbook: Creating a basic web application:**

   The base application of this Dataset Dashboard has been created during the cookbook: Creating a basic web application.

3) **Cookbook: Data modeling in MongoDB using MongoEngine:**

   The data used during this cookbook, is modeled, indexed and imported in the cookbook: Data modeling in MongoDB using MongoEngine.

4) **Programming manual: Creating the Python-Flask web application:**

   The middleware that will be used during this programming manual is created in the programming manual: Creating the Python Flask web application.

If you have not read these documents yet, please do so before reading this document.

The purpose of this programming manual is to create an 3D map viewer application using the AngularJS JavaScript framework and the JavaScript framework Cesium. This application is an extension of our Angular base application and the 2D Map Viewer.

The reason this application is an extension of the 2D Map Viewer is because the 3D Map Viewer has most of the functionalities which the 2D Map Viewer has.

The Angular apps will perform API calls to our Flask application and our Flask application will then retrieve the requested data via queries, performed on our datastores. The results are then returned to our Angular applications.

This programming manual serves as a guideline for the steps you have to perform to create a 2D Map Viewer using OpenLayers and visualize the data retrieved by the Flask-API.

During this programming manual the code is explained using the inline comments in the source code located in the folder: “POC”. It’s highly recommended to use the source code provided in this folder when creating the web application yourself.

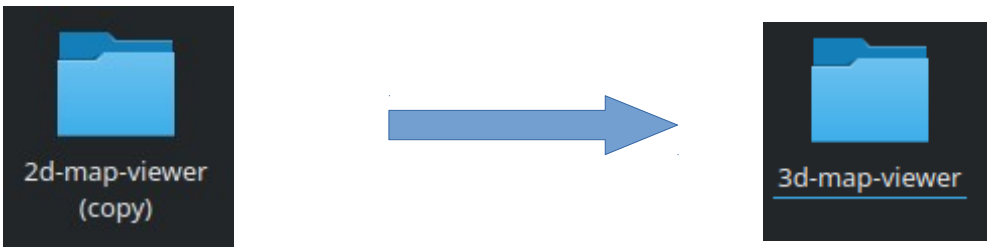
NOTE: Sometimes you will notice that in the code which you have to create some functions do not exist yet. Don’t worry about this since they will be added later on during the programming manual!

Table of Contents

1.Introduction

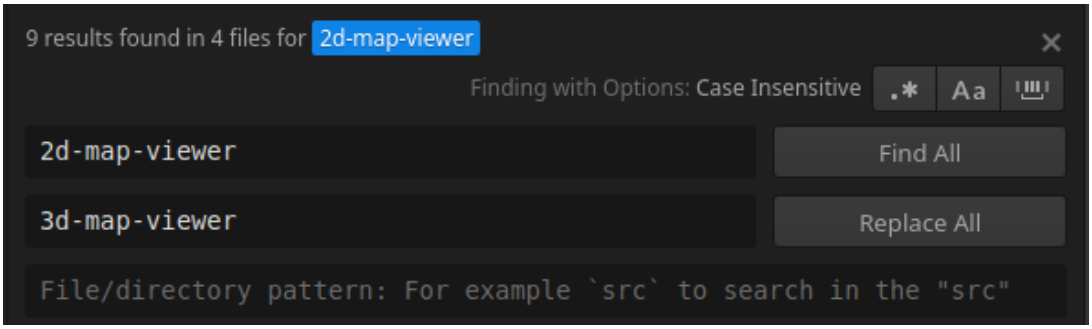
1.1 Getting ready

This application is basically the same as the 2D Map viewer so we can copy this application and start creating the 3D map viewer from there. After we copied the 2d-map-viewer folder, we need to change some names and titles. We start by changing the name of the folder we just copied from 2d-map-viewer to 3d-map-viewer, as shown in the image below.

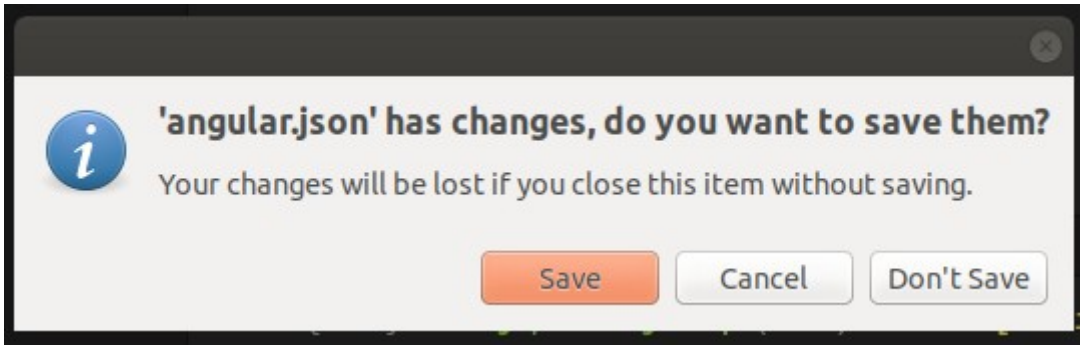


We also need to edit the project name: “2d-map-viewer” to “3d-map-viewer”. If you are using the code-editor Atom, this is done by performing the following steps:

1) In the edit press the keys Ctrl + shift + f in the Atom editor.
2) In the screen that pops up enter: “2d-map-viewer” in the find section and “3d-map-viewer” in the replace section, as shown in the illustration below. Then click on find all.



3) Click on replace all and on the save button in the screen that pops up.

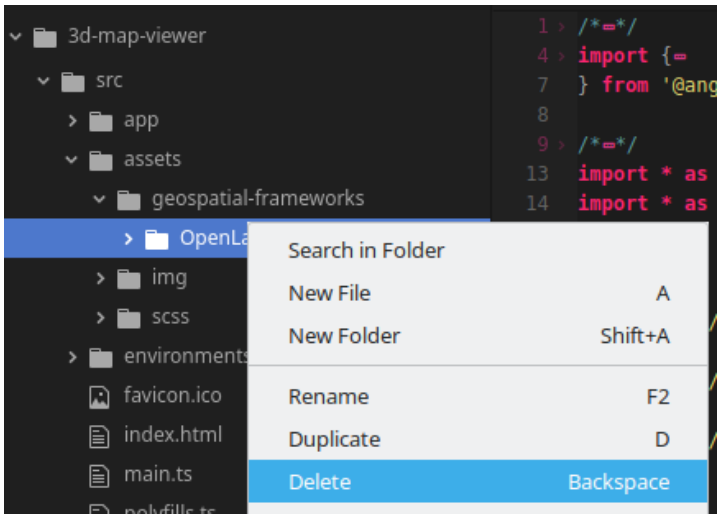


4) In the file: index.html located in the folder 3d-map-viewer/src, replace the title from 2D Map Viewer to 3D Map Viewer.
5) In the file: sidebar.component.html located in the folder src/app/components/sidebar/, change the text: “3D Map Viewer” to “3D Map Viewer”.

1.2 Adding the JavaScript framework Cesium

Since we don't need the geospatial JavaScript framework OpenLayers anymore we can remove the OpenLayers folder which can be found in the folder: "3d-map-viewer/src/assets/geospatial-frameworks/".

Deleting the OpenLayers framework can done by opening the 3d-map-viewer folder in Atom, finding the OpenLayers Folder, right clicking the folder and selecting delete as shown in the illustration below:



Now that we have removed the JavaScript Framework: "OpenLayers" we should add the JavaScript Framework:"Cesium".

Just like with OpenLayers; Adding the geospatial framework to our Angular application can be done in 2 ways which are as follows:

1) **Installing the NPM Package: "cesium":**

   This is the first technique which you can use to install Cesium in your application. During this programming manual we will not be using this technique. If you want to read up on using this technique you should visit the following URL: https://cesium.com/blog/2018/03/12/cesium-and-angular/

2) **Downloading the Cesium source code:**

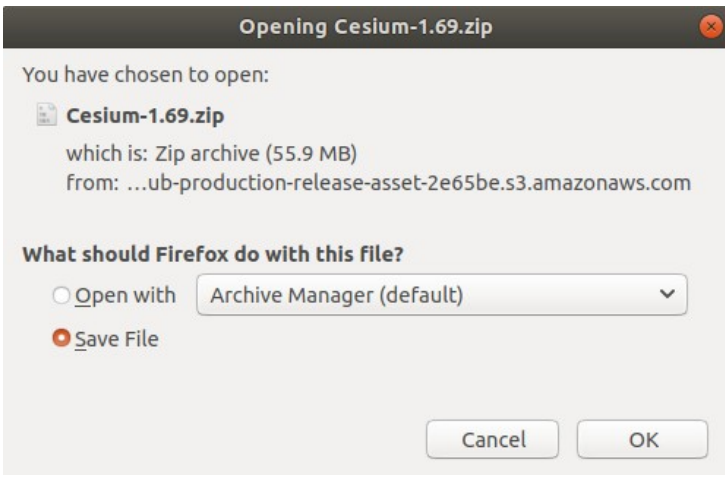
   During this programming manual we are going to use this technique. We do this because, from the version control point of view, this method is the best method since there are no files added to the Node_Modules folder of the application. Using this technique we are going to add the geospatial framework as static files in the assets folder of our application. This enables us to easily switch to a newer or older version of the geospatial framework.

First we want to download the Cesium source code from the Cesium website which is located on the following URL:

https://cesium.com/downloads/

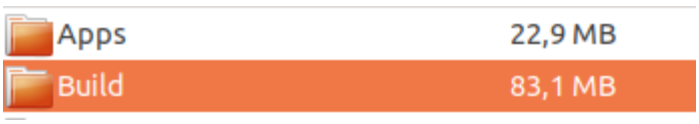
When navigating to the URL mentioned above you should be greeted with a green download button as shown in the illustration below:

Click on the download button and then select Save File and click on Ok as shown in the illustration below:



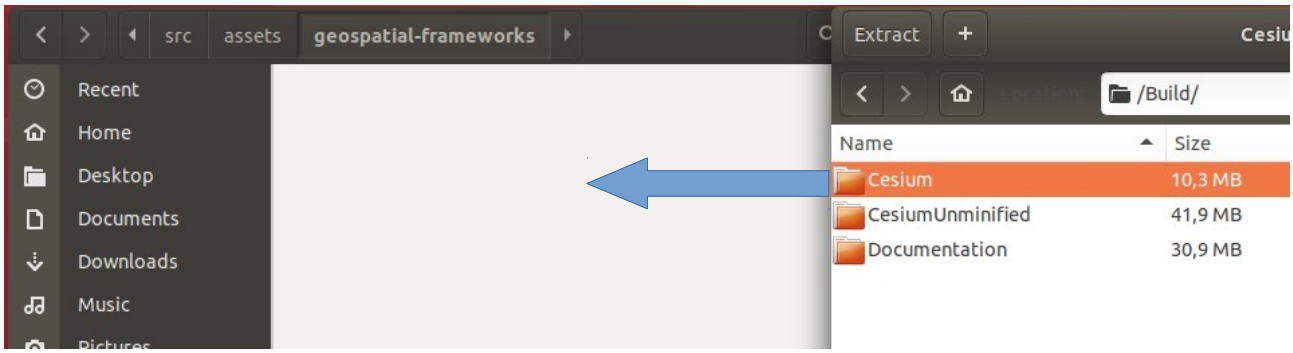
After a few seconds you will end up with a ZIP folder in your downloads folder as shown in the illustration below:

Open the ZIP file. This ZIP contains all the files related to the JavaScript framework Cesium. Since we only need the Cesium source code we should click on the build folder as shown in the illustration below:



Once we are in the build folder we now have to copy the folder called: “Cesium” to the geospatial-frameworks folder in our 3d-map-viewer application (“3d-map-viewer/src/assets/geospatial-frameworks/”)

This can be done by dragging and dropping the Cesium folder as shown in the illustration below:



Now that we have the Cesium source code in our Angular application we need to edit the index.html file of our application so that the application knows where Cesium is located.

So let’s open the index.html file which is located in the folder: “3d-map-viewer/src/”.

We currently have the references to the OpenLayers framework in the <head> tag of the index.html. So let’s change it according to the illustration shown below:

```
<!--Here we add the reference to the OpenLayers style sheet-->
<link rel="stylesheet" href="/assets/geospatial-frameworks/OpenLayers/ol.css"/>

<!--Here we add the reference to the OpenLayers javascript code-->
<script src="/assets/geospatial-frameworks/OpenLayers/ol.js"></script>
```

```
<!--Here we add the reference to the Cesium style sheet-->
<link rel="stylesheet" href="/assets/geospatial-frameworks/Cesium/Widgets/widgets.css"/>

<!--Here we add the reference to the OpenLayers javascript code-->
<script src="/assets/geospatial-frameworks/Cesium/Cesium.js"></script>
```

That’s it! Now we can use the JavaScript framework Cesium throughout our application. In the next section we are going to cleanup the map.component.ts file to remove the code from the 2D Map Viewer application which we don’t need for the 3D Map viewer application.

1.2 Cleaning up the map.component.ts file

Because the 3D Map Viewer is an extension of the 2D Map Viewer we can keep most of the code which we created during the manual: “Creating an 2 Dimensional Map Viewer”.

This section describes what files and code you should remove and keep in order to turn the 2D Map Viewer into the 3D Map Viewer. So let’s open the map.component.ts file located in the folder: “3d-map-viewer/src/app/page/map-page/”.

To make it easier to remove code we can fold the code so that only to first lines of the module imports, functions etc. are shown.

This is done by pressing Ctrl+ Alt + Shift + [on your keyboard. This will result in the following map.component.ts file:

```
1  /*…*/
4  import {…
7  } from '@angular/core';
8
9  /*…*/
13 import * as Chartist from 'chartist';
14 import * as tooltip from 'chartist-plugin-tooltips'
15
16 /*…*/
21 import {…
23 } from 'src/app/services/map.service'
24 import {…
26 } from 'src/app/services/crane.service'
27 import {…
29 } from 'src/app/services/port.service'
30
31 /*…*/
36 declare const ol: any;
37
38 /*…*/
42 export class Item {…};
83
84 /*…*/
93 @Component({…})
98 export class MapComponent implements OnInit {…};
21
```

1.2.1 Cleaning the module imports

We want to start of by removing the module imports which we don’t need anymore. The f

-Chartist modules

- Import portservice

- Provides remove portservie

const ol → const Cesium

MapLayer - > MapTileLayr

SeaLayer

LayerStyles

Color list

WidtList

LineType

StyleDict

ElevationProfile

ElevationProfileOpen

PortLayer

Add Overlay

SetDynamicOverlays

SetStaticOverlays

TogleLayer

ToggleOverlay

SetLayerStyle

AnimateRoute

CleaAnimation

CreateelevationProfile

LoadElevationData

CreatePortLayer