TUTORIAL 2

NodeJS & OpenLayers & PostGIS

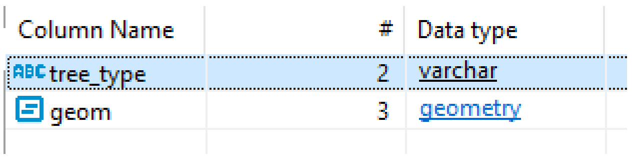
Abdülkadir Çakır

05/05/2020

Implementation of Database

Let’s create a table with these parameters:

```
CREATE TABLE public.trees (
	tree_type varchar NULL,
	geom geometry(POINT, 4326) NULL
);
```

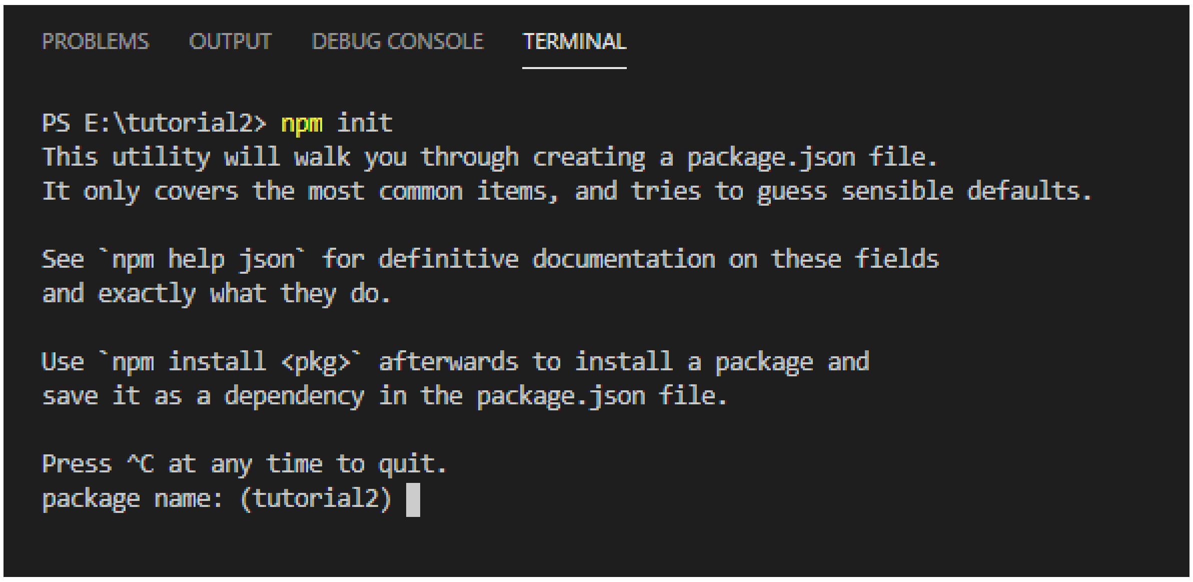


Column Name	#	Data type
tree_type	2	varchar
geom	3	geometry

PS: Don’t forget to run :

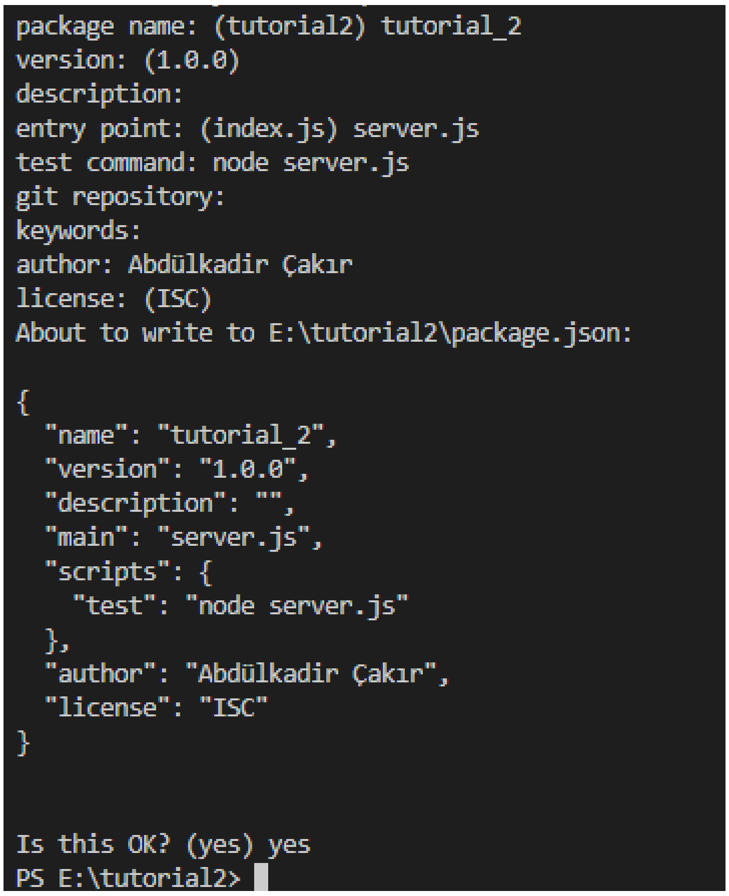
CREATE EXTENSION POSTGIS;

It is time to create Node JS project.



```
PS E:\tutorial2> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (tutorial2)
```

Fill it like this:



```
package name: (tutorial2) tutorial_2
version: (1.0.0)
description:
entry point: (index.js) server.js
test command: node server.js
git repository:
keywords:
author: Abdülkadir Çakır
license: (ISC)
About to write to E:\tutorial2\package.json:

{
  "name": "tutorial_2",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "node server.js"
  },
  "author": "Abdülkadir Çakır",
  "license": "ISC"
}


Is this OK? (yes) yes
PS E:\tutorial2>
```

Install packages using npm install: (example: npm install pg)

pg

express

body-parser

Create a file named server.js (depends on what did you write on npm init entry point)

And go with these steps:

- Importing plugins
- Create DB Connection
- Enabling body-parser for POST method.
- Setting up port
- Setting up working directory

```
const express = require("express");
const pg=require("pg").Pool;
const bodyParser = require('body-parser');
const app = express();
const pool=new pg({host:'localhost',database:'hw_db',user:'postgres',password:'postgres',port:'5432',ssl:false});
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());
const _port = process.env.PORT || 5000;
const _app_folder = __dirname + '/dist' ;
app.use(express.static(__dirname + '/dist' ));
```

After that you can implement you APIs.

- Implement data retrieval API (GET parameter will be enough)
- And data insert API (POST parameter will be enough)

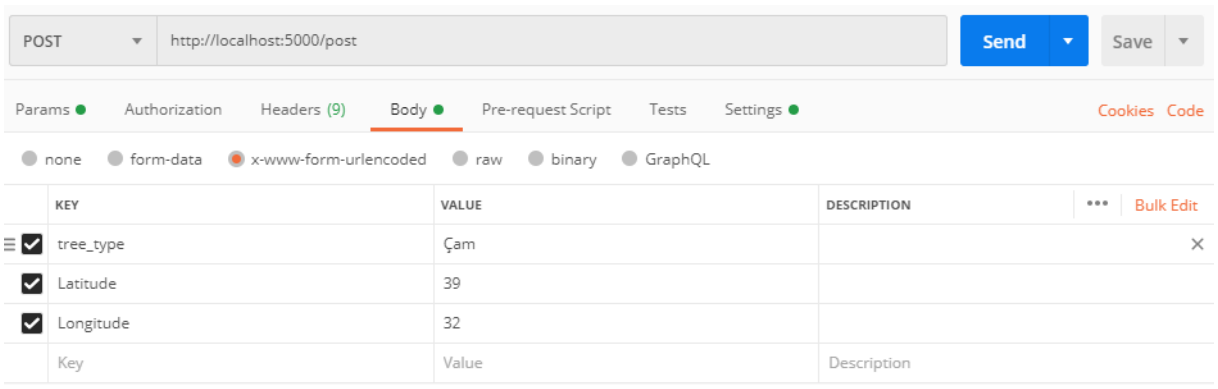
```
var table_name="trees";
app.get("/api/data",function(req,res)
{
    pool.query("SELECT jsonb_build_object('type','FeatureCollection','features
', jsonb_agg(feature)) FROM (SELECT jsonb_build_object('type','Feature','geome
try',ST_AsGeoJSON(geom)::jsonb, 'properties', to_jsonb(row) - 'gid' - 'geom')
AS feature  FROM (SELECT * FROM "+table_name+") row) features;", (err1, res1)
=>
        {
            if(err1) {return console.log(err1);}
            res.json(res1.rows[0]["jsonb_build_object"]);
        });
});

app.post('/post', function(request, response){
    pool.query("INSERT INTO "+table_name+" VALUES('"+request.body.tree_type+"'
,ST_SETSRID(ST_MAKEPOINT("+request.body.Longitude+","+request.body.Latitude+")
,4326));", (err1, res1) =>
        {
            if(err1)
                {   console.log(request.body);
                    return console.log(err1);}
                response.statusCode = 200;
                response.setHeader('Content-Type', 'text/plain');
                response.end('Data Store Success!\n');
        });
});
```

Now Implement express server startup codes.

```
app.all('*', function (req, res) {
    res.status(200).sendFile(`/`, {root: _app_folder});
});

app.listen(_port, function () {
    console.log("Node Express server for " + app.name + " listening on http://
localhost:" + _port);
});
```

Run it using “node server.js”

You can use PostMan to check your APIs are working.

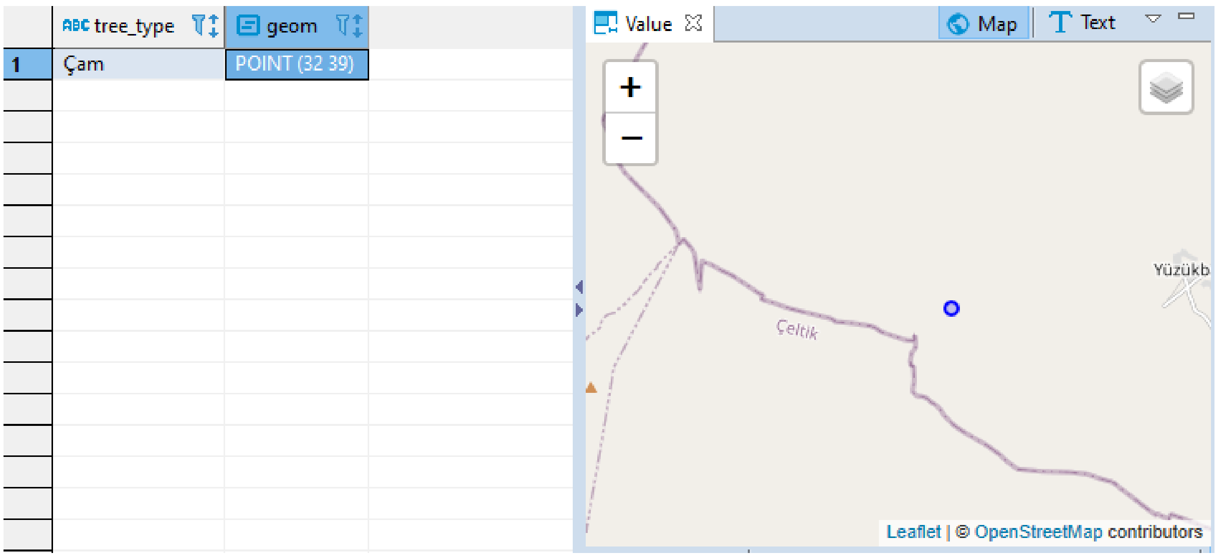


POST http://localhost:5000/post

Body: x-www-form-urlencoded

KEY	VALUE	DESCRIPTION
tree_type	Çam	
Latitude	39	
Longitude	32	
Key	Value	Description

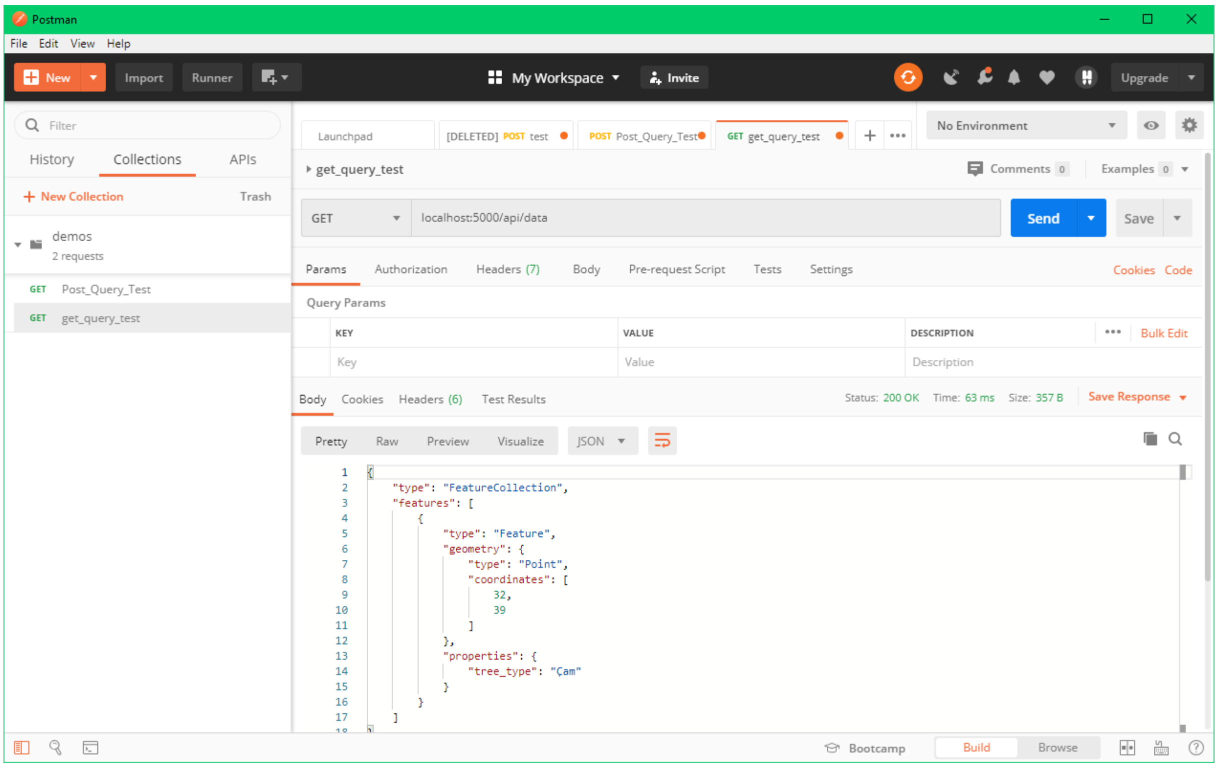
After pressing send button, Record inserted into table.



	tree_type	geom
1	Çam	POINT (32 39)

Let’s check GET API:

(No parameter required)



get_query_test

GET localhost:5000/api/data

Status: 200 OK Time: 63 ms Size: 357 B

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [
                    32,
                    39
                ]
            },
            "properties": {
                "tree_type": "Çam"
            }
        }
    ]
```

Last insert from POST request is available at both in database and /api/data request.

Let’s create front-end!

Create folder named with dist

And create new files in that folder:

- Index.html
- map_handler.js
- styles.css

Index.html:

```html
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v6.2.0/css/ol.css">
        <link rel="stylesheet" href="styles.css">
        <script src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v6.2.0/build/ol.js"></script>
    </head>
    <body>
        <div id="map" class="map"></div>
        <script src="map_handler.js"></script>
        <div id="right_panel" class="right_panel">
            <br>
            <br>
            <br>
            <p>Tree Type:</p>
            <select id="tree_type">
                <option value="Çam">Çam Ağacı</option>
                <option value="Kavak">Kavak Ağacı</option>

            </select>
            <p>Latitude:</p>
            <input type="text" id="Latitude"></input>
            <p>Longitude:</p>
            <input type="text" id="Longitude"></input>
            <br>
            <br>
            <br>
            <button onclick="submit();">Submit</button>
        </div>
    </body>
</html>
```

Map_handler.js:

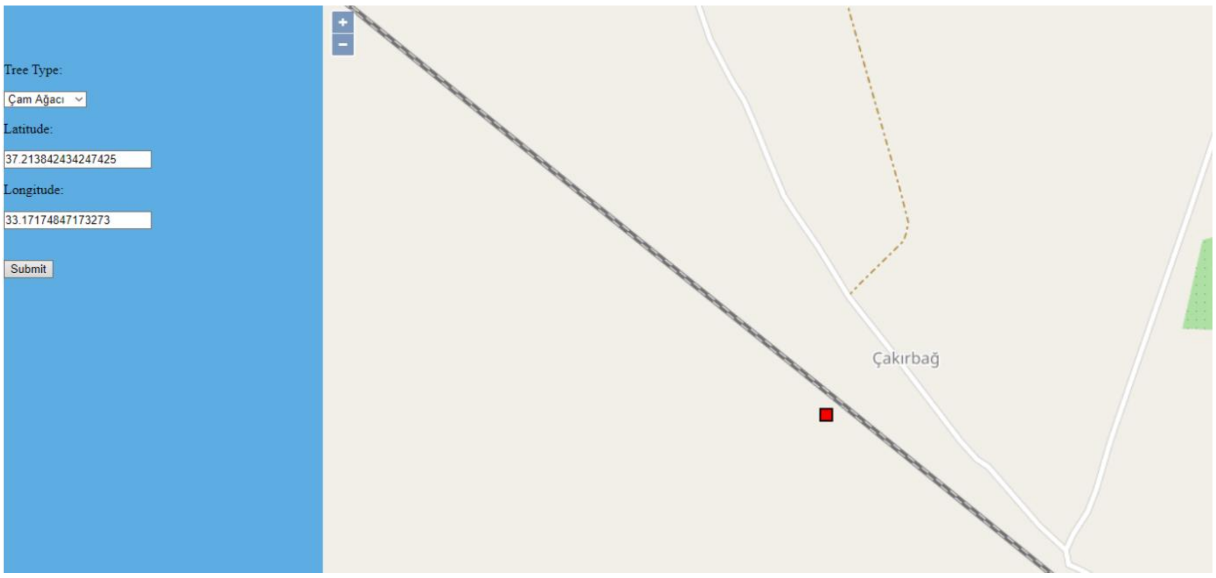
```
var baseMapLayer = new ol.layer.Tile({
    source: new ol.source.OSM()
});
var layer = new ol.layer.Tile({
  source: new ol.source.OSM()
});
var center = ol.proj.fromLonLat([32, 39]);
var view = new ol.View({
  center: center,
  zoom: 10
});
var map = new ol.Map({
    target: 'map',
    view: view,
    layers: [layer]
});

var vectorSource = new ol.source.Vector({
        url:"/api/data",
        format: new ol.format.GeoJSON({ featureProjection: "EPSG:4326" })
});

var stroke = new ol.style.Stroke({color: 'black', width: 2});
var fill = new ol.style.Fill({color: 'red'});

var markerVectorLayer = new ol.layer.Vector({
    source: vectorSource,
    style: new ol.style.Style({
        image: new ol.style.RegularShape({
          fill: fill,
          stroke: stroke,
          points: 4,
          radius: 10,
          angle: Math.PI / 4
        })
      })

});

map.addLayer(markerVectorLayer);
var select = new ol.interaction.Select({multiple:false});
select.on('select', fnHandler);
map.addInteraction(select);
map.on("click",handleMapClick);
function handleMapClick(evt)
{
  var coord=ol.proj.transform(evt.coordinate, 'EPSG:3857', 'EPSG:4326');
  document.getElementById("Latitude").value=coord[1];
  document.getElementById("Longitude").value=coord[0];
}

function fnHandler(e)
{
    var coord = e.mapBrowserEvent.coordinate;
    let features = e.target.getFeatures();
    features.forEach( (feature) => {
        console.log(feature.getProperties().tree_type);

    document.getElementById("tree_type").value=feature.getProperties().tree_ty
pe;
    });
    if (e.selected[0])
    {
    var coords=ol.proj.transform(e.selected[0].getGeometry().getCoordinates(),
 'EPSG:3857', 'EPSG:4326');
    document.getElementById("Latitude").value=coords[1];
    document.getElementById("Longitude").value=coords[0];
    console.log(coords);
    }
}

function submit()
{
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/post", true);
    xhr.setRequestHeader('Content-Type', 'application/json');
    var data=JSON.stringify({

        Latitude: document.getElementById('Latitude').value,
        Longitude: document.getElementById('Longitude').value,
        tree_type: document.getElementById('tree_type').value
    });
    xhr.send(data);
    location.reload();
}
```

Styles.css:

```
html,body
{
width: 100%;
height: 100%;

}
.map
{
    width: 80%;
    height: 100%;
    float: right;
}

.right_panel
{

    background-color: rgb(94, 173, 226);
    width: 20%;
    height: 100%;
}
```

It’s ready!

Save and re-run the server.js



Tree Type:

Çam Ağacı

Latitude:

37.213842434247425

Longitude:

33.17174847173273

Submit