Here are the NEW Map Apex classes, VF Pages, VF components, and static resources, Apex Triggers etc.

Apex Classes:

1. GisClient – It is returning the cases which are not closed in given area (by radius) to show on map.
2. GisClientTest – Test Class for GisClient
3. X311GisConnectorClr – It is queries the data by radius less than double radium miles = radius feet/ 5280 of case records. trying to get the getGeoCodeSpec specs for three objects like

GISCaseRT\_\_c

GISCaseRTLayers\_\_c

GISCaseRTAttributes\_\_c

VF Pages:

1. 311GisMap – It is used to take the address and locating in map itself.
2. 311GisMapAis - It is used to take the address and locating in map itself.
3. GisPopup – It is a pop up window for map on case edit page.
4. GisMap – This is to locate the address on the map.

VF Components:

1. FetchLayerInfoJS - This is a generic template for Visualforce Component. With this template, you may adjust the default elements and values and add new elements and values.
2. GisConnector - This is a generic template for Visualforce Component. With this template, you may adjust the default elements and values and add new elements and values.
3. GisMap - This is a generic template for Visualforce Component. With this template, you may adjust the default elements and values and add new elements and values.
4. GisMapLeaflet – Marker/Pin related

Static Resources:

1. GisAppJs – Used in above vf pages
2. GisConfigJs – Used in above vf pages
3. GisMapJs – Used in above vf pages

Here is the detailed code of above all.

Apex Classes:

1. GisClient

global class GisClient {  
  @RemoteAction  
    global static List<Case> QueryByRadius(String recordType, Double x, Double y, Double radiusFeet) {  
        //system.debug('recordType: ' + recordType);  
        //system.debug('x: ' + x);  
        //system.debug('y: ' + y);  
    //system.debug('radiusFeet: ' + radiusFeet);  
  
        ID recordTypeId = ID.ValueOf(recordType);  
        //system.debug('recordTypeId: ' + recordTypeId);  
  
    // convert to miles  
        Double radiusMiles = radiusFeet / 5280;  
        //system.debug('radiusMiles: ' + radiusMiles);  
  
        // run query  
        List<Case> cases = [  
            SELECT Id, CaseNumber,  Service\_Request\_Type\_\_c,  Subject, Street\_\_c, Centerline\_\_longitude\_\_s, Centerline\_\_latitude\_\_s  
            FROM Case  
            WHERE  
        Centerline\_2272x\_\_c != null AND  
            Centerline\_2272y\_\_c != null AND  
            Centerline\_\_Longitude\_\_s != null AND  
            Centerline\_\_Latitude\_\_s != null AND  
            RecordTypeId = :recordTypeId AND  
            IsClosed = false AND  
            DISTANCE(Centerline\_\_c, GEOLOCATION(:y, :x), 'mi') < :radiusMiles  
        ];  
  
    return cases;  
    }  
}

1. GisClientTest

@isTest  
public class GisClientTest {  
  
@isTest static void doCallTest() {  
  
RecordType rt = [SELECT id,Name   
                           FROM RecordType   
                           WHERE SobjectType='Case' AND Name='Abandoned Bike' ];  
  
Case cse=New Case(Type='Service Request Type',Department\_\_c='Streets Department', RecordTypeId =rt.Id,Service\_Request\_Type\_\_c='Abandoned Bike',   
Tagged\_With\_Yellow\_Alert\_Notice\_\_c ='Yes',subject='Abandoned Bike',origin='Email',Missing\_Damaged\_Parts\_\_c='YES', Unusable\_\_c='No',  
Time\_Locked\_in\_Same\_Location\_\_c='One month or more');  
Insert cse;  
  
GisClient.QueryByRadius (rt.Id,22,18,2);  
  
    }  
  
}

1. X311GisConnectorClr

global class X311GISConnectorClr {   
  global class GeoCodeSpec {  
    global string address;  
    global string geocodeType;  
    global List<string> layers;  
    global Map<string, string> attributes;  
      
    global GeoCodeSpec() {  
      layers=new List<string>();  
      attributes=new Map<string, string>();  
    }  
  }  
  
  global class GeoCodeData {  
    global string x;  
    global string y;  
    global string standardizedAddress;  
    global string geocodeType;  
    global Map<string, string> attributes;  
      
    global GeoCodeData() {  
      attributes=new Map<string, string>();  
    }  
  }  
  
  global class GeoCode {  
    global decimal lat;  
    global decimal lng;  
  }  
    
  // this is actually what an ESRI "Extent" is, sans the type and spatialreference fields  
  global class ViewPort {  
    global double xmin;  
    global double xmax;  
    global double ymin;  
    global double ymax;  
  }  
    
  global class CasePoint {  
    global decimal x;  
    global decimal y;  
    global string label;  
    global string description;  
    global string id;  
  }  
    
  @RemoteAction  
  global static List<CasePoint> Query(string aString, ViewPort vp) {  
    system.debug(aString);  
    system.debug(vp);  
      
    ID anId = ID.ValueOf(aString);  
      
    List<CasePoint> lstCasePoints= new List<CasePoint>();  
      
    for (Case cs : [  
        SELECT  Id, CaseNumber, Subject, Street\_\_c, Centerline\_\_longitude\_\_s, Centerline\_\_latitude\_\_s   
          FROM  Case   
         WHERE  Centerline\_2272x\_\_c != null   
           AND  Centerline\_2272y\_\_c != null  
           AND  Centerline\_2272x\_\_c >= :vp.xmin AND Centerline\_2272x\_\_c <= :vp.xmax   
           AND  Centerline\_2272y\_\_c >= :vp.ymin AND Centerline\_2272y\_\_c <= :vp.ymax  
           AND  Centerline\_\_Longitude\_\_s != null   
           AND  Centerline\_\_Latitude\_\_s != null  
           AND  RecordTypeId = :anId   
           AND  IsClosed=false  
      ]) {  
      CasePoint cp=new CasePoint();  
      cp.x=cs.Centerline\_\_longitude\_\_s;  
      cp.y=cs.Centerline\_\_latitude\_\_s;  
      cp.id=cs.Id;  
      cp.label=cs.CaseNumber;  
      cp.Description = cs.Street\_\_c + ' : ' + cs.Subject;  
        
      lstCasePoints.add(cp);  
    }  
    return lstCasePoints;  
  }  
    
  @RemoteAction  
  global static List<CasePoint> QueryByRadius(String recordType, Double x, Double y, Double radiusFeet) {  
    //system.debug('recordType: ' + recordType);  
    //system.debug('x: ' + x);  
    //system.debug('y: ' + y);  
      
    ID recordTypeId = ID.ValueOf(recordType);  
    //system.debug('recordTypeId: ' + recordTypeId);  
      
    //system.debug('radiusFeet: ' + radiusFeet);  
      
    // radiusFeet had to be a Double for this operation to work  
    Double radiusMiles = radiusFeet / 5280;  
    //system.debug('radiusMiles: ' + radiusMiles);  
      
    List<CasePoint> lstCasePoints = new List<CasePoint>();  
      
    // run query  
    List<Case> cases = [  
      SELECT Id, CaseNumber, Subject, Street\_\_c, Centerline\_\_longitude\_\_s, Centerline\_\_latitude\_\_s   
      FROM Case   
      WHERE Centerline\_2272x\_\_c != null   
          AND Centerline\_2272y\_\_c != null  
          AND Centerline\_\_Longitude\_\_s != null   
          AND Centerline\_\_Latitude\_\_s != null  
          AND RecordTypeId = :recordTypeId   
          AND IsClosed = false  
          AND DISTANCE(Centerline\_\_c, GEOLOCATION(:y, :x), 'mi') < :radiusMiles  
    ];  
      
    // loop over results  
    for (Case aCase : cases) {        
      CasePoint cp = new CasePoint();  
      cp.x = aCase.Centerline\_\_longitude\_\_s;  
      cp.y = aCase.Centerline\_\_latitude\_\_s;  
      cp.id = aCase.Id;  
      cp.label = aCase.CaseNumber;  
      cp.Description = aCase.Street\_\_c + ' : ' + aCase.Subject;  
        
      lstCasePoints.add(cp);  
    }  
      
    return lstCasePoints;  
  }  
  
  @RemoteAction  
  global static GeoCodeSpec getGeoCodeSpec(string address, string aString) {  
    system.debug(address);  
    system.debug(aString);  
      
    ID anId = ID.ValueOf(aString);  
      
    RecordType aType = [ select id, name, developerName from RecordType where id = :anId limit 1 ];  
      
    GeoCodeSpec gcs=new GeoCodeSpec();  
      
    gcs.address=address;  
    for (GISCaseRT\_\_c grt : [SELECT GeoCode\_Type\_\_c FROM GISCaseRT\_\_c WHERE RecordTypeDevName\_\_c = :aType.DeveloperName]) {  
      gcs.layers.add(grt.GeoCode\_Type\_\_c);  
    }  
      
    for (GISCaseRTLayers\_\_c lyr : [SELECT GISLayerName\_\_c FROM GISCaseRTLayers\_\_c WHERE RecordTypeDevName\_\_c = :aType.DeveloperName]) {  
      gcs.layers.add(lyr.GISLayerName\_\_c);  
    }  
      
    for (GISCaseRTAttributes\_\_c atr : [SELECT GISAttributeName\_\_c FROM GISCaseRTAttributes\_\_c WHERE RecordTypeDevName\_\_c = :aType.DeveloperName]) {  
      gcs.attributes.put(atr.GISAttributeName\_\_c, '');  
    }  
      
    return gcs;  
  }

VF Pages:

1. 311GisMap

<!-- 311GISMap is used for getting the user input address and locating it on the map -->

<apex:page showHeader="false" sidebar="false">

<!-- Use JSON2 library for parsing json in IE7 -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/json2/20140204/json2.min.js" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/encoder.js')}" />

<input id="address" type="text" value=""/>

<button id="locate" onclick="window.AddressCandidate();">locate</button>

<button onclick="window.addLayer();">add layers</button>

<br />

<div id="multiple" style="display:none">

There are similar addresses. Please select one from the list

<span id="list">

<select id="selectAddress">

</select>

<button id="locate" onclick="locateFromList()">locate</button>

</span>

</div>

<c:gisMap />

<script>

if(typeof aGisApi !== "undefined"){

window.opener.setGisApi(aGisApi);

}

function addPoint(aStreetAddress) {

if (window.opener && window.opener.saveGeodata) {

aGisApi.mapAddress(aStreetAddress, 'CenterlineOnly', function (data) {

console.log('going to save geodata', data);

window.opener.saveGeodata(data);

});

} else {

aGisApi.mapAddress(aStreetAddress);

}

}

var AddressCandidate = function() {

actionStatusOn('Checking similar addresses');

$$('#multiple').css('display','none');

anAddress = document.getElementById('address').value;

console.log("anAddress: " + anAddress);

GISController.AddressCandidates(anAddress, function(result, event) {

if (event.status == false) {

alert('callout failed');

return;

}

OnSuccess(result, event, function(result) {

if (result.successful == 'false') {

actionStatusOff();

console.log(JSON.stringify(result));

alert(Encoder.htmlDecode(result.message[0]));

return;

}

console.log(result);

//console.log("addresses :",result.jmap.addresses);

if (result.jmap.addresses.length < 2){

actionStatusOff();

addPoint(document.getElementById('address').value);

} else {

var addressArray = [];

for (var j = 0; j < result.jmap.addresses.length; j++) {

addressArray.push(Encoder.htmlDecode(result.jmap.addresses[j].standardizedAddress));

}

console.log("addresses :",addressArray);

var select = document.getElementById("selectAddress");

select.options.length = 0;

for (var i = 0; i < addressArray.length; i++) {

var opt = addressArray[i];

var el = document.createElement("option");

el.textContent = opt;

el.value = opt;

select.appendChild(el);

}

actionStatusOff();

$$('#multiple').css('display','block');

}

});

},{escape:true, buffer:false});

}

function locateFromList(){

var selectedValue= $$('#selectAddress').val();

addPoint(selectedValue);

}

var addLayer = function() {

var url = 'https://www.arcgis.com/sharing/rest/generateToken?username=311application.philly&password=IGCD15MS&client=referer&f=json&referer=www.salesforce.com';

var aClient = new XedeHttpClient();

aClient.post(url,

function(result) {

window.console && console.log('finished : ', url);

// window.console && console.log('result', result);

var resultObject = JSON.parse(result);

aGisApi.addLayer([

// [ "Street Lights", "https://services2.arcgis.com/YRlMlhu838MHTM0n/ArcGIS/rest/services/streetPoles/FeatureServer/0?token=" + resultObject.token ], Commented for support ticket #11112987

// [ "Parks", "https://services2.arcgis.com/YRlMlhu838MHTM0n/ArcGIS/rest/services/PPR\_adminBound/FeatureServer/0?token=" + resultObject.token ], Commented for support ticket #11112987

"https://services2.arcgis.com/YRlMlhu838MHTM0n/ArcGIS/rest/services/streetPoles/FeatureServer/0?token=" + resultObject.token, // Street poles for support ticket #11112987

"https://services2.arcgis.com/YRlMlhu838MHTM0n/arcgis/rest/services/water\_Hydrants/FeatureServer/0?token=" + resultObject.token, // water hydrants

"https://services2.arcgis.com/YRlMlhu838MHTM0n/arcgis/rest/services/water\_Inlets/FeatureServer/0?token=" + resultObject.token, // water inlets

"https://services2.arcgis.com/YRlMlhu838MHTM0n/ArcGIS/rest/services/PPR\_adminBound/FeatureServer/0?token=" + resultObject.token, // parks for support ticket #11112987

//uncommented for support ticket# :08967149

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Business/MapServer/8", // business permits

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Construction/MapServer/12", // zoning permits

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Construction/MapServer/18", // building permits

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Construction/MapServer/0", // Construction Violation

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Construction/MapServer/10", // Mechanical Permits

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Construction/MapServer/14", // Electrical Permits

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Construction/MapServer/16", // Plumbing Permits

//"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Hospitals/MapServer/0", // hospital

//"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/Health\_Centers/MapServer/0", // health center

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ServiceAreas/MapServer/8", // police district

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ServiceAreas/MapServer/12", // rubbish

//"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/MDO/Philly\_Rising\_Boundaries/MapServer/0", // philly rising boundaries

//"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ServiceAreas/MapServer/19", // L&I districts

//"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ZoningMap/MapServer/0", // zoning parcels

//"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ZoningMap/MapServer/1", // zoning overlay

"{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ZoningMap/MapServer/6", // zoning base

[ "Zoning" , "{!$Setup.GIS\_\_c.ESRIServices\_\_c}/PhilaGov/ZoningMap/MapServer" ], // zoning all

"https://services2.arcgis.com/YRlMlhu838MHTM0n/arcgis/rest/services/Smoke\_Alarm\_Requests/FeatureServer/0?token=" + resultObject.token, // smoke alarm requests

]);

},

function(result) {

window.console && console.log('FAILED : ', url);

window.console && console.log('result', result);

alert("Unable to login to Philly GIS Services.");

}

);

}

function isIE () {

var myNav = navigator.userAgent.toLowerCase();

return (myNav.indexOf('msie') != -1) ? parseInt(myNav.split('msie')[1]) : false;

}

var XedeHttpClient = function() {

//Use XDomainRequest in IE 9 and below to prevent CORS issue

var useXDomain = false;

if(typeof isIE() === 'number'){

if(isIE() <=9) {

useXDomain = true;

}

}

this.do1 = function(aMethod, anArray) {

if (anArray.length != 2 && anArray.length != 3)

return;

var aUrl = anArray[0];

var onSuccess = anArray[1];

var onFailure = function() {};

if (arguments.length === 3)

onFailure = arguments[2];

var anHttpRequest;

if(useXDomain){

anHttpRequest = new XDomainRequest();

anHttpRequest.onload = function() {

onSuccess(anHttpRequest.responseText);

}

if(isIE()===9)

anHttpRequest.onprogress = function() {};

}else{

anHttpRequest = new XMLHttpRequest();

anHttpRequest.onreadystatechange = function() {

//window.console && console.log('anHttpRequest', anHttpRequest);

if (anHttpRequest.readyState === 4) {

if (anHttpRequest.status >= 200 && anHttpRequest.status <= 299){

onSuccess(anHttpRequest.responseText);

}else{

onFailure(anHttpRequest.responseText);

}

}

}

}

if(useXDomain){

anHttpRequest.open(aMethod, aUrl);

}else{

anHttpRequest.open(aMethod, aUrl,true);

}

if (this.timeout)

anHttpRequest.timeout = this.timeout;

anHttpRequest.send( null );

}

this.post = function(aUrl) {

this.do1("POST", arguments);

}

this.get = function(aUrl) {

this.do1("GET", arguments);

}

this.timeout;

}

$$(function() {

if (window.opener && window.opener.getGeoCodeSpec) {

window.opener.getGeoCodeSpec(function (geoCodeSpec) {

if (geoCodeSpec) {

$$("input#address").val(Encoder.htmlDecode(geoCodeSpec.address));

var layerList = [];

geoCodeSpec.layers.forEach(function(eachLayer, index, list) {

layerList.push('{!$Setup.GIS\_\_c.ESRIServices\_\_c}/' + eachLayer);

});

aGisApi.addLayer(layerList);

}

});

}

});

// easy function for handling errors from Javascript Remote requests

// this guy deals with the errors, and calls aFunction on success.

function OnSuccess(result, event, aFunction) {

if (event.status) {

aFunction(result);

return;

}

if (event.type === 'exception' && event.message.indexOf("Logged in?") != -1) {

var host = window.location.host;

var protocol = window.location.protocol;

window.top.location.href = protocol +"//" + host ;

}

else

alert("Exception: " + Encoder.htmlDecode(event.message));

}

</script>

</apex:page>

2. 311GisMapAis

<!-- 311GISMap is used for getting the user input address and locating it on the map -->

<apex:page showHeader="false" sidebar="false" controller="X311GISConnectorClr">

<!-- Use lodash for array utilities -->

<script src="//cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.4/lodash.min.js"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/async/2.5.0/async.min.js"></script>

<script src="//code.jquery.com/jquery-2.2.4.min.js"></script>

<!-- Use Proj4.js for transforming geographic coordinates -->

<script src="//cdnjs.cloudflare.com/ajax/libs/proj4js/2.4.3/proj4.js"></script>

<!-- TODO take out placeholder value -->

<!--<input id="address" type="text" value="1234 market st"/>

<button id="locate" onclick="handleLocateButtonClick();">locate</button>

<button onclick="window.addLayer();">add layers</button>-->

<div id="multiple" style="display:none">

There are similar addresses. Please select one from the list

<span id="list">

<select id="selectAddress">

</select>

<button id="locate" onclick="handleLocateButtonClickMultiple()">locate</button>

</span>

</div>

<script>

window.$$ = $.noConflict();

/\* AIS \*/

// listen for 'enter' key on address input

$$("input#address").on('keyup', function (e) {

if (e.keyCode == 13) {

handleLocateButtonClick();

}

});

// configure proj4 to use local state plane projection

proj4.defs('EPSG:2272', '+proj=lcc +lat\_1=40.96666666666667 +lat\_2=39.93333333333333 +lat\_0=39.33333333333334 +lon\_0=-77.75 +x\_0=600000 +y\_0=0 +ellps=GRS80 +datum=NAD83 +to\_meter=0.3048006096012192 +no\_defs ');

// helper fn to convert a string to camel case

// https://stackoverflow.com/a/4068586/676001

function toCamelCase(s) {

return s.replace(

/\w+/g,

function (w) { return w[0].toUpperCase() + w.slice(1).toLowerCase(); }

);

}

// handle clicks to the locate button

function handleLocateButtonClick(e) {

actionStatusOn('Validating address...');

$$('#multiple').css('display','none');

var inputAddress = document.getElementById('address').value;

fetchAis(inputAddress);

}

// handle clicks to the locate button in the multiple result selector

function handleLocateButtonClickMultiple(e) {

var selected = $$('#selectAddress').val();

console.log('did select from multiple:', selected);

fetchAis(selected);

}

// this function gets user input and queries the Address Information System. this is

// intended to replace the Address Candidate function.

function fetchAis(inputAddress) {

console.log("fetch ais:", inputAddress);

var url = 'https://api.phila.gov/ais/v1/addresses/' + encodeURIComponent(inputAddress);

$$.ajax({

url: url,

data: {

include\_units: true,

opa\_only: true,

gatekeeperKey: 'cb076d7301c315ba0f039b8f7744ac6d',

},

success: didFetchAis,

error: didNotFetchAis,

});

}

// on successful ais fetch

function didFetchAis(data) {

console.log('did fetch ais', data);

actionStatusOff();

// check number of results

var features = data.features,

resultCount = Array.isArray(features) ? features.length : 0;

if (resultCount === 1) {

var feature = features[0];

didGetValidAddress(feature);

// if we got multiple results

} else if (resultCount > 1) {

// show dropdown

populateMultipleResultDropdown(features);

// if we got no results

} else {

console.log('ais successful but no features');

alert('Could not locate that address. Please try another search.');

}

}

// on unsuccessful ais fetch

function didNotFetchAis(jqXHR, textStatus, errorThrown) {

console.log('error fetching ais:', errorThrown, jqXHR);

actionStatusOff();

alert('Error while validating address: ' + errorThrown);

}

function populateMultipleResultDropdown(features) {

var $select = $$('#selectAddress');

$select.empty();

\_.each(features, function (feature) {

console.log('feature', feature);

var address = feature.properties.street\_address;

var $opt = $$('<option>')

.text(address)

.val(address);

$select.append($opt);

});

$$('#multiple').show();

}

function didGetValidAddress(feature) {

console.log('did get valid address', feature);

var coords = feature.geometry.coordinates,

x = coords[0],

y = coords[1],

address = feature.properties.street\_address;

// add marker and zoom

MAP.addGeocodeMarker(x, y, address);

MAP.zoomTo(x, y);

// get "related" cases (recent, nearby cases with the same record type)

var recordTypeId = window.opener && window.opener.RECORD\_TYPE\_ID;

// DEBUG: if no parent window, use record type id for Dangerous Sidewalk

if (!recordTypeId) recordTypeId = '012G00000014Gt6IAE';

fetchRelated(recordTypeId, x, y);

// fetch layer info

var layerInfo = fetchLayerInfo\_(feature, null, function () {

console.log('did fetch layer info');

});

}

// configure query layers

var QUERY\_LAYERS = {

//address: {

// type: 'esri',

// url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Addresses/MapServer/0',

// options: {

// where: function (feature) {

// return "ADDRESS = '" + feature.properties.street\_address + "'";

// }

// }

//},

// REVIEW do we still need this now that there's a separate query for demo permits?

//buildingPermit: {

// type: 'where',

// endpoint: 'PhilaGov/Construction/MapServer/18',

// options: {

// where: function (feature) {

// return 'ADDRKEY = ' + feature.properties.li\_address\_key;

// }

// }

//},

//businessLicense: {

// type: 'where',

// endpoint: 'PhilaGov/Business/MapServer/8',

// options: {

// where: function (feature) {

// return 'ADDRKEY = ' + feature.properties.li\_address\_key;

// }

// }

//},

//district: {

// type: 'intersects',

// endpoint: 'PhilaGov/ServiceAreas/MapServer/19',

// options: {

// //TODO

// }

//},

demoPermit: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Construction/MapServer/18',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " AND APPLICATION\_TYPE = 'BP\_DEMO'";

}

}

},

dumpsterPrivate: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Business/MapServer/8',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " AND TYPE = '3230'";

},

},

},

familyDaycare: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Business/MapServer/8',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " AND TYPE = '3397'";

},

},

},

foodLicense: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Business/MapServer/6',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

healthCenter: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Health\_Centers/MapServer/0',

options: {

where: function (feature) {

// note: this layer doesn't have standardized addresses, so there's no

// guarantee this query will work. adding some logic to improve matching.

return "UPPER(FULL\_ADDRESS) LIKE '" + feature.properties.street\_address + "%'";

}

}

},

hospital: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Hospitals/MapServer/0',

options: {

where: function (feature) {

return "UPPER(ADDRESS) LIKE '" + feature.properties.street\_address + "%'";

}

}

},

rentalLicense: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/RentalLicense/MapServer/2',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

rentalViolation: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/RentalLicense/MapServer/0',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

vacancy: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Vacancy/MapServer/2',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

violationProperty: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Violations/MapServer/0',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " and VIOLATION\_STATUS <> 'NOT COMPLIED'";

}

}

},

// using ais to get zoning

//zoningBaseLayers: {

// type: 'intersects',

// endpoint: 'PhilaGov/Addresses/MapServer/0',

// options: {

// // TODO

// }

//},

zoningPermit: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Construction/MapServer/12',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

property: {

type: 'carto',

url: '//phl.carto.com/api/v2/sql',

options: {

q: function (feature) {

var accountNum = feature.properties.opa\_account\_num;

return "SELECT \* FROM opa\_properties\_public WHERE parcel\_number = '" + accountNum + "'";

},

},

},

};

function fetchLayerInfo\_(feature, caseType) {

console.log('fetch layer info', feature, caseType);

actionStatusOn('Querying layers...');

//var GIS\_BASE\_URL = '{!$Setup.GIS\_\_c.ESRIServices\_\_c}' + '/',

var queryLayerFns = {};

// build up an array of query fns for each layer

\_.forOwn(QUERY\_LAYERS, function (queryLayerDef, queryLayerName) {

//console.log('query layer', queryLayerName, queryLayerDef);

var queryLayerFn = function (callback) {

//var endpoint = queryLayerDef.endpoint,

var url = queryLayerDef.url,

type = queryLayerDef.type,

options = queryLayerDef.options;

switch(type) {

case 'esri':

// form params

var params = {

outFields: '\*',

returnGeometry: 'true',

f: 'json'

},

whereFn = options.where,

where = whereFn(feature);

params.where = where;

$$.ajax({

url: url + '/query',

data: params,

success: function (dataStr) {

// parse json string

var data = $$.parseJSON(dataStr);

// check for error

if (data.error) {

console.log('did get error for where query', queryLayerName, data);

callback(true);

return;

}

//console.log('did get data for where query', queryLayerName, data, url);

// unpack attributes object

var features = data.features,

feature = Array.isArray(features) && features.length > 0 ? features[0] : null,

result;

if (feature) {

result = feature.attributes;

console.log('set', queryLayerName, result);

}

callback(null, result);

},

error: function () {

console.log('did get esri error');

callback(queryLayerName + ' had error');

}

});

break;

case 'carto':

var qFn = options.q,

q = qFn(feature),

params = {

q: q,

};

$$.ajax({

url: url,

data: params,

success: function (data) {

// unpack attributes

var rows = data.rows,

row = Array.isArray(rows) && rows.length > 0 ? rows[0] : null;

callback(null, row);

},

error: function () {

console.warn('did get carto error');

callback(queryLayerName + ' had error');

},

});

break;

default:

console.warn('unhandled query type: ' + type);

callback("unhandled query type '" + type + "'", null);

break;

}

};

queryLayerFns[queryLayerName] = queryLayerFn;

});

// execute query fns in parallel

async.parallel(

queryLayerFns,

// then do this

function (err, results) {

actionStatusOff();

// check for an error

if (err) {

console.log('async parallel error:', err);

return;

}

console.log('async parallel finished', results);

saveGeodata(feature, results);

}

);

}

function saveGeodata(feature, layerInfo) {

console.log('save geodata', feature, layerInfo);

// pass data back to parent window

var geodata = createGeodataObject(feature, layerInfo);

console.log('geodata', geodata);

// DEBUG: if window.opener isn't defined, we're probably just testing this

// from the X311GISMap page directly. don't try to call saveGeodata on the parent.

if (!window.opener) {

console.warn('window.opener not defined. geodata will not be saved.');

return;

}

window.opener.saveGeodata(geodata);

}

// takes an ais feature and formats it as a "geodata" object which the parent window can understand.

function createGeodataObject(feature, layerInfo) {

console.log('create geodata object', feature, layerInfo);

var props = feature.properties,

coords = feature.geometry.coordinates,

coordsStatePlane = proj4('EPSG:2272', coords),

// parent page checks for the token `Intersection`, so camel case this

matchType = toCamelCase(feature.ais\_feature\_type),

addressKey = (props.li\_address\_key || '').split('|')[0];

// we're now getting the L&I district from AIS, so update layer info

layerInfo.district = props.li\_district;

var geodata = {

gisData: {

standardizedAddress: props.street\_address,

matchType: matchType,

// address keys are pipe-delimited, so take the first one

addressKey: addressKey,

// REVIEW is it ok to use the opa address here?

liAddress: props.opa\_address,

zoning: props.zoning,

},

// this doesn't seem to be used

description: null,

// this doesn't seem to be used

featureData: null,

// TODO populate this based on if there's a related case

parentId: null,

// this seems to be related to the parent case

label: null,

x: coords[0],

y: coords[1],

x2272: coordsStatePlane[0],

y2272: coordsStatePlane[1],

layerInfo: layerInfo,

// this seems to be where service area values go

attributes: {

SA\_STREETS\_ZipCode: props.zip\_code,

SA\_PHILLYRISING: props.philly\_rising\_area,

SA\_POLICE\_District: props.police\_district,

SA\_STREETS\_Rubbish: props.rubbish\_recycle\_day,

SA\_PLANNING\_2016Councilmanic: props.council\_district\_2016,

SA\_STREETS\_Sanitation\_Districts: props.sanitation\_district,

}

};

return geodata;

}

function fetchRelated(recordTypeId, x, y, callback) {

console.log('fetch related');

X311GISConnectorClr.QueryByRadius(recordTypeId, x, y, 500, didFetchRelated);

}

function didFetchRelated(results, event) {

console.log('did fetch related', results, event);

// put markers on map

MAP.addRelatedMarkers(results);

// TODO populate list on right-hand side

}

</script>

<c:gisMapLeaflet />

3.GisPop

<apex:page showHeader="false" sidebar="false" controller="GisClient">

<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" />

<link rel="stylesheet" href="//unpkg.com/leaflet@1.1.0/dist/leaflet.css" />

<style>

/\*disable button highlighting\*/

button {

outline: none;

}

#search-bar {

background: #2176d2;

height: 50px;

padding-left: 10px;

padding-right: 10px;

padding-top: 5px;

}

#search-input-container {

float: left;

}

#search-input {

font-size: 16px;

height: 30px;

padding-left: 5px;

padding-right: 5px;

width: 300px;

}

.search-button {

font-size: 16px;

height: 35px;

padding-left: 20px;

padding-right: 20px;

}

#search-status {

float: right;

padding-top: 10px;

display: none;

}

#search-action {

color: #ffffff;

font-size: 16px;

font-style: italic;

font-weight: bold;

}

#search-spinner {

color: #ffffff;

margin-right: 3px;

margin-top: -2px;

}

#map {

clear: both;

position: absolute;

top: 50px;

bottom: 0;

left: 0;

right: 0;

}

/\*MODAL\*/

.modal {

display: none;

position: fixed;

z-index: 2000;

left: 0;

top: 0;

width: 100%;

height: 100%;

overflow: auto;

background-color: rgb(0,0,0);

background-color: rgba(0,0,0,0.4);

}

.modal-content {

background-color: #fefefe;

margin: 15% auto;

padding: 20px;

border: 1px solid #888;

width: 500px;

}

.modal-content h1 {

font-size: 16px;

}

.modal-content select {

height: 35px;

font-size: 16px;

padding-left: 5px;

padding-right: 5px;

width: 380px;

}

.modal-content button {

margin-left: 4px;

}

.modal-close {

color: #aaa;

float: right;

font-size: 28px;

font-weight: bold;

}

.modal-close:hover,

.modal-close:focus {

color: black;

text-decoration: none;

cursor: pointer;

}

</style>

<div id="container">

<div id="search-bar">

<div id="search-input-container">

<input id="search-input" placeholder="Enter an address to search for"/>

<button id="search-button" class="search-button">Locate</button>

</div>

<div id="search-status">

<span id="search-spinner" class="fa fa-spinner fa-lg fa-pulse"></span>

<span id="search-action"></span>

</div>

</div>

<div id="map"></div>

<div id="multiple-results-modal" class="modal">

<div class="modal-content">

<span class="modal-close">&times;</span>

<h1>Your search had multiple results.</h1>

<p>Please choose one of the following addresses:</p>

<select></select>

<button class="search-button">Locate</button>

</div>

</div>

</div>

<!-- JavaScript: Vendor -->

<script src="//code.jquery.com/jquery-2.2.4.min.js"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.4/lodash.min.js"></script>

<script src="//unpkg.com/leaflet@1.1.0/dist/leaflet.js"></script>

<script src="//unpkg.com/esri-leaflet@2.0.8"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/async/2.5.0/async.min.js"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/proj4js/2.4.3/proj4.js"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/OverlappingMarkerSpiderfier-Leaflet/0.2.6/oms.min.js"></script>

<!-- Sentry for error logging -->

<script src="//cdn.ravenjs.com/3.16.1/raven.min.js" crossorigin="anonymous"></script>

<script>

Raven.config('https://e1b674da28104b5aa6e3be5222684fc8@sentry.io/190393').install()

</script>

<!-- JavaScript: Local -->

<script src="{!URLFOR($Resource.GisConfigJs)}"></script>

<script src="{!URLFOR($Resource.GisAppJs)}"></script>

<script src="{!URLFOR($Resource.GisMapJs)}"></script>

</apex:page>

</apex:page>

1. GisMap

<apex:page showHeader="false" sidebar="false">

<!--

This Page is intended for demonstration purposes. Do Not Deploy to Production!

-->

<input id="address" type="text" value="10 s broad st"/>

<button id="locate" onclick="addPoint()">locate</button>

<c:gisMap />

<script>

function addPoint() {

aGisApi.mapAddress(document.getElementById('address').value);

}

</script>

</apex:page>

VF Components:

1. FetchLayerInfoJs

<apex:component >

<script type="text/javascript">

function fetchLayerInfo(aGisPoint, layerCallBack) {

window.console && console.log('aGisPoint:', aGisPoint);

// thanks to https://github.com/caolan/async#parallel

actionStatusOn('Querying layers');

var layerTimestamp = (new Date()).getTime();

var xCrd;

var yCrd;

async.parallelLimit({

// address info - 4200 monument rd

address: function(aCallback) {

esriQueryAddress( 'PhilaGov/Addresses/MapServer/0', aGisPoint.gisData.taxAddress, aCallback);

},

// building permit : 1628 MANTON ST

buildingPermit: function(aCallback) {

esriQueryAddressKey('PhilaGov/Construction/MapServer/18', aGisPoint.gisData.addressKey, aCallback);

},

// business license

// family daycare : 1515 BARRINGER ST

// dumpster private : 1547 spring garden

// dumpster inactive : 600 JAMESTOWN

businessLicense: function(aCallback) {

esriQueryAddressKey('PhilaGov/Business/MapServer/8', aGisPoint.gisData.addressKey, aCallback);

},

// food license : 2154 BRIDGE ST

foodLicense: function(aCallback) {

esriQueryAddressKey('PhilaGov/Business/MapServer/6', aGisPoint.gisData.addressKey, aCallback);

},

// health centers - 1412 Fairmount

healthCenter: function(aCallback) {

esriQueryAddress( 'PhilaGov/Health\_Centers/MapServer/0', aGisPoint.gisData.standardizedAddress, aCallback);

},

// hospitals - 4200 monument rd

hospital: function(aCallback) {

esriQueryAddress( 'PhilaGov/Hospitals/MapServer/0', aGisPoint.gisData.standardizedAddress, aCallback);

},

// L and I district

district: function(aCallback) {

esriQueryPoint('PhilaGov/ServiceAreas/MapServer/19', aGisPoint.x, aGisPoint.y, aCallback);

},

//rental licenses 12412 Academy Rd

rentalLicense: function(aCallback) {

esriQueryAddressKey( 'PhilaGov/RentalLicense/MapServer/2', aGisPoint.gisData.addressKey, aCallback);

},

//rental violations 4936 ROSEHILL

rentalViolation: function(aCallback) {

esriQueryAddressKey( 'PhilaGov/RentalLicense/MapServer/0', aGisPoint.gisData.addressKey, aCallback);

},

//vacancy 1515 barringer

vacancy: function(aCallback) {

esriQueryAddressKey( 'PhilaGov/Vacancy/MapServer/2', aGisPoint.gisData.addressKey, aCallback);

},

//violations 5626 WARRINGTON

violationProperty: function(aCallback) {

esriQueryAddressKeyAnd( 'PhilaGov/Violations/MapServer/0', aGisPoint.gisData.addressKey, "and VIOLATION\_STATUS != 'NOT COMPLIED'", aCallback);

},

//zoning permit : 1650 MARKET

zoningPermit: function(aCallback) {

esriQueryAddressKey( 'PhilaGov/Construction/MapServer/12', aGisPoint.gisData.addressKey, aCallback);

},

zoningBaseLayers: function(aCallback) {

//Added for ticket #10569385 BY SUNDAR

$$.get("https://api.phila.gov/ulrs/v3/addresses/"+ encodeURIComponent(aGisPoint.gisData.standardizedAddress) +"?srid=4326&format=json").done(function(data){

console.log("Data Loaded: " + JSON.stringify(data) );

xCrd = data.addresses[0].xCoord;

yCrd = data.addresses[0].yCoord;

esriQueryPointZoning( 'PhilaGov/ZoningMap/MapServer/6', xCrd , yCrd, aCallback);

});

}

},

3, // parallelLimit to 3-outstanding requests at a time

function(err, results) {

window.console && console.log(err);

window.console && console.log('elapsed millisecods : ', (new Date()).getTime() - layerTimestamp);

window.console && console.log('fetchLayerInfo RESULTS :',results);

var layerInfo = {};

for (each in results) {

if (each == 'businessLicense')

continue;

if (results[each] && results[each].features && results[each].features.length) {

layerInfo[each] = results[each].features[0].attributes;

}

}

if (results.businessLicense && results.businessLicense.features) {

for (i = 0; i < results.businessLicense.features.length; i++) {

var attributes = results.businessLicense.features[i].attributes;

switch (attributes.TYPE) {

case '3397':

layerInfo.familyDaycare = attributes;

break;

case '3230':

layerInfo.dumpsterPrivate = attributes;

break;

default:

break;

}

}

}

if (results.buildingPermit && results.buildingPermit.features) {

for (i = 0; i < results.buildingPermit.features.length; i++) {

var attributes = results.buildingPermit.features[i].attributes;

switch (attributes.APPLICATION\_TYPE) {

case 'BP\_DEMO':

layerInfo.demoPermit = attributes;

break;

default:

break;

}

}

}

if (layerCallBack)

layerCallBack(layerInfo);

});

}

function esriQueryAddress(layer, aStandardizedAddress, aCallback) {

// get address field to test against

var testField;

// don't use upper function for querying address layer (pwd parcels)

if (layer === 'PhilaGov/Addresses/MapServer/0') {

testField = 'ADDRESS';

} else {

testField = 'UPPER(ADDRESS)';

}

// form where clause

var where = 'where=' +

testField +

" = '" +

aStandardizedAddress +

"'";

esriRestQuery(

layer,

encodeURIComponent(where),

aCallback

);

}

function esriQueryAddressKey(layer, anAddressKey, aCallback) {

esriQueryAddressKeyAnd(layer, anAddressKey, 'and 1=1', aCallback)

}

function esriQueryAddressKeyAnd(layer, anAddressKey, andClause, aCallback) {

var aQuery = new esri.tasks.Query();

aQuery.where = "ADDRKEY=" + anAddressKey + ' ' + andClause;

//esriLayerQuery(layer, aQuery, aCallback);

esriRestQuery(layer, 'where=ADDRKEY%3D' + anAddressKey, aCallback);

}

function esriQueryPoint(layer, anX, aY, aCallback) {

esriRestQuery(

layer,

'geometry={0}%2C{1}&geometryType=esriGeometryPoint&inSR=4326&spatialRel=esriSpatialRelIntersects'.format(anX, aY),

aCallback

);

}

//Added for ticket #10569385

//ADDED BY BRINDHA

function esriQueryPointZoning(layer, anX, aY, aCallback) {

esriRestQuery(

layer,

'geometry={0}%2C{1}&geometryType=esriGeometryPoint&inSR=4326&spatialRel=esriSpatialRelIntersects&relationParam=&outFields=\*&returnGeometry=true'.format(anX,aY),

aCallback

);

}

//ADDED BY BRINDHA

function esriLayerQuery(layer, query, aCallback) {

// https://developers.arcgis.com/javascript/jssamples/fl\_paging.html

var url = '{!$Setup.GIS\_\_c.ESRIServices\_\_c}/' + layer;

window.console && console.log('esriLayerQuery url', url);

var aLayer = new esri.layers.FeatureLayer(url, { outFields:["\*"] });

aLayer.queryFeatures(query,

function(featureSet) {

window.console && console.log('featureSet : ', featureSet);

window.console && console.log('split : ', (new Date()).getTime() - splitStamp);

aCallback(null, featureSet);

},

function(error) {

window.console && console.log('FAILED : ', url);

window.console && console.log('split : ', (new Date()).getTime() - splitStamp);

aCallback(null, error);

}

);

}

function esriRestQuery(layer, where, aCallback) {

var splitStamp = (new Date()).getTime();

var url = '{!$Setup.GIS\_\_c.ESRIServices\_\_c}/{0}/query?{1}&outFields=\*&returnGeometry=true&f=json'.format(layer, where);

window.console && console.log('esriRestQuery url', url);

var aClient = new XedeHttpClient();

aClient.get(url,

function(result) {

window.console && console.log('finished : ', url);

window.console && console.log('split : ', (new Date()).getTime() - splitStamp);

aCallback(null, JSON.parse(result));

},

function(result) {

window.console && console.log('FAILED : ', url);

window.console && console.log('result', result);

window.console && console.log('split : ', (new Date()).getTime() - splitStamp);

aCallback(null, JSON.parse(result));

}

);

}

</script>

</apex:component>

1. GisConnector

<apex:component controller="X311GISConnectorClr">

<apex:attribute name="recordType" type="string" description="the value to use for a record type" required="true" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets, 'js/jquery-1.10.2.min.js')}"/>

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/encoder.js')}" />

<script src="//cdnjs.cloudflare.com/ajax/libs/es5-shim/1.2.4/es5-shim.min.js" />

<script>

var j$ = jQuery.noConflict();

var aGisApi;

var aGisPoint = null;

var theGeoData = null;

var license = null;

window.gisConnector = this;

window.console && console.log('window = ', window, 'gisConnector', this);

window.RECORD\_TYPE\_ID = '{!recordtype}';

function viewportChangeHandler(aValue) {

getCaseList(aValue, function(result) {

var dataArray = [];

result.forEach(function(each, index, array){

var aCasePoint = {

x: each.x,

y: each.y,

label: each.label,

description: each.description,

uniqueId: each.id

};

dataArray.push(aCasePoint);

});

aGisApi.drawDataArray(dataArray);

});

}

// this function is used to add layers to the map. GisAPI's addLayer()

// this function is called by the iframe's gismap.component. If the component

// isn't inside an iframe, the code I have there "window.parent.setGisApi()"

// probably isn't going to work.

function setGisApi(anObject) {

aGisApi = anObject;

aGisApi.onViewportChange(viewportChangeHandler);

}

j$(function(){

var win,

$addressInput = j$('input[id$=street]');

// allows gis popup to get the search input from this page

window.getSearchInput = function () {

return $addressInput.val();

}

j$("<button class='btn' id='gis'>...</button>").insertAfter($addressInput);

j$("button#gis").click(function(){

win=window.open('{!URLFOR($Page.GisPopup)}','gisMap','toolbar=0,scrollbars=1,location=0,statusbar=0,menubar=0,resizable=1,width=1100,height=600,top=200,left=200');

return false;

});

j$(window).unload(function() {

win ? win.close() : '';

});

});

function getGeoCodeSpec(callback) {

X311GISConnectorClr.getGeoCodeSpec(

j$("input[id$=street]").val(),

'{!recordtype}',

function(result, event) {

if (event.status == "false") {

alert(result.message[0]);

} else {

callback(result);

}

}, {escape:true, buffer:false}

);

}

function getCaseList(viewPort, callback) {

X311GISConnectorClr.Query(

'{!recordtype}', { xmin: viewPort.xmin, xmax: viewPort.xmax, ymin: viewPort.ymin, ymax: viewPort.ymax },

function(result, event) {

if (event.status == "false") {

alert(result.message[0]);

} else {

window.console && console.log(result);

callback(result);

}

}, {escape:true, buffer:false});

}

function saveGeodata(geoData) {

window.console && console.log('saveGeoData()');

window.console && console.log(geoData);

theGeoData = geoData;

updatePageFromGeoData();

}

function updatePageFromGeoData() {

/\*

map a text field using

j$("input[id$=textFieldId]").val(.. the value ..);

and a combo box field using

j$("[id$=comboBoxId]").val(.. the value ..);

\*/

j$("input[id$=street]").val(theGeoData.gisData.standardizedAddress);

j$("input[id$=clMatchType]").val(theGeoData.gisData.matchType);

j$("input[id$=hansenAddressKey]").val(theGeoData.gisData.addressKey);

j$("input[id$=hansenAddress]").val(theGeoData.gisData.liAddress);

j$("input[id$=clLongitude]").val(theGeoData.x);

j$("input[id$=clLatitude]").val(theGeoData.y);

// not sure why the street field was getting overridden by this(?)

//j$("input[id$=street]").val(Encoder.htmlDecode(theGeoData.description));

j$("input[id$=cl2272x]").val(theGeoData.x2272);

j$("input[id$=cl2272y]").val(theGeoData.y2272);

j$("input[id$=featureData]").val(theGeoData.featureData);

//Added for ticket #10767426 starts

j$("[id $='rent']").val('');

//Added for ticket #10767426 ends

if (theGeoData.parentId && theGeoData.parentId.trim()) {

j$("input[id$=parentId]").val(theGeoData.label);

j$("input[id$=parentId\_lkid]").val(theGeoData.label); // Changed to label from parentId for Related cases change 10577449 - UNISYS

createDup( j$("input[id$=parentId\_lkid]").val(theGeoData.label));

}

if (j$("input[id$=clMatchType]").val() == 'Intersection')

j$("[id$=InterMidblock]").val('Intersection');

else

j$("[id$=InterMidblock]").val('Mid-block');

for (var fld in theGeoData.attributes) {

if(j$("input[id$=" + fld + "]").val() != theGeoData.attributes[fld]) {

j$("input[id$=" + fld + "]").val(theGeoData.attributes[fld]);

j$("input[id$=" + fld + "]").change();

}

}

if (theGeoData.layerInfo) {

var attribute;

if (attribute = theGeoData.layerInfo.district) {

window.console && console.log('district', attribute);

j$("input[id$=liDistrict]").val(attribute);

}

if (attribute = theGeoData.layerInfo.familyDaycare) {

window.console && console.log('familyDaycare', attribute);

j$("input[id$=daycareName]").val(attribute.CONTACT\_COMPANY\_NAME);

if (attribute.STATUS == 'ACTIVE')

j$("[id$=daycareLicenseActive]").val('Yes');

else

j$("[id$=daycareLicenseActive]").val('No');

}

if (attribute = theGeoData.layerInfo.foodLicense) {

window.console && console.log('foodLicense', attribute);

if (attribute.STATUS == 'ACTIVE') {

j$("[id$=foodLicenseActive]").val('Yes');

if (license == null)

license = 'No';

else

license = 'Yes';

}

else

j$("[id$=foodLicenseActive]").val('No');

}

if (attribute = theGeoData.layerInfo.rentalLicense) {

window.console && console.log('rentalLicense', attribute);

j$("input[id$=rentalLicenseDescription]").val(attribute.DESCRIPTION);

if (attribute.STATUS == 'ACTIVE') {

window.console && console.log('STATUS=' + attribute.STATUS);

j$("[id$=rentalLicenseActive]").val('Yes');

}

}

if (attribute = theGeoData.layerInfo.rentalViolation) {

window.console && console.log('rentalViolation', attribute);

j$("input[id$=rentalViolationDescription]").val(attribute.DESCRIPTION);

j$("input[id$=rentalViolationStatus]").val(attribute.VIOLATION\_STATUS);

}

if (attribute = theGeoData.layerInfo.vacancy) {

window.console && console.log('vacancy', attribute);

j$("input[id$=vacancyDescription]").val(attribute.DESCRIPTION);

if (attribute.STATUS == 'ACTIVE')

j$("[id$=vacancyStatus]").val('Yes');

else

j$("[id$=vacancyStatus]").val('No');

}

if (attribute = theGeoData.layerInfo.hospital) {

window.console && console.log('hospital', attribute);

j$("input[id$=healthName]").val(attribute.NAME);

j$("input[id$=hospitalName]").val(attribute.NAME);

j$("input[id$=hospitalPhone]").val(attribute.PHONE);

}

if (attribute = theGeoData.layerInfo.healthCenter) {

window.console && console.log('healthCenter', attribute);

j$("input[id$=healthName]").val(attribute.NAME);

j$("input[id$=healthCenterlName]").val(attribute.NAME);

j$("input[id$=healthCenterPhone]").val(attribute.PHONE);

}

if (attribute = theGeoData.layerInfo.dumpsterPrivate) {

window.console && console.log('dumpsterPrivate', attribute);

j$("input[id$=dumpsterDescription]").val(attribute.DESCRIPTION);

j$("input[id$=dumpsterStatus]").val(attribute.STATUS);

if(attribute.STATUS == 'ACTIVE')

if (license == null)

license = 'No';

else

license = 'Yes';

}

if (attribute = theGeoData.layerInfo.property) {

window.console && console.log('property', attribute);

var owner = attribute.owner\_1;

j$("input[id$=addressOwner]").val(owner);

j$("input[id$=addressBuildingDescription]").val(attribute.building\_code\_description);

if (owner && owner.length > 0) {

j$("[id$=resident]").val('Yes');

} else {

j$("[id$=resident]").val('No');

}

}

// Function added for checking residential or commercial for ticket #10569385

// Id changed from rorc to rent for ticket #10569385

// TODO use theGeoData.gisData.zoning for this instead

if (attribute = theGeoData.gisData.zoning) {

window.console && console.log('zoning', attribute);

// check for residential zoning code

if (attribute.includes('RM') ||

attribute.includes('RS') ||

attribute.includes('RT')

) {

j$("[id$=rent]").val('Residential');

} else {

j$("[id$=rent]").val('Commercial');

}

$("[id$=rent]").trigger("change")

}

if (attribute = theGeoData.layerInfo.zoningPermit) {

window.console && console.log('zoningPermit', attribute);

j$("input[id$=zoningPermitDescription]").val(attribute.DESCRIPTION\_OF\_WORK);

if (attribute.APNO != null) {

window.console && console.log('STATUS=' + attribute.STATUS);

j$("[id$=zoningPermitActive]").val('Yes');

}

}

if (j$("input[id$=zoningPermitActive]").val() == 'Yes')

j$("[id$=zoningPermitActive1]").val('Yes');

else

j$("[id$=zoningPermitActive1]").val('No');

if(license == 'Yes')

j$("[id$=dump1]").val('Yes');

// REVIEW what is this and where is it defined?

//address();

}

}

</script>

</apex:component>

1. GisMap

<apex:component controller="XedeRemote" extensions="GISController" >

<apex:attribute name="layers" type="string" required="false" description="feature layers" />

<link rel="stylesheet" href="//js.arcgis.com/3.8/js/esri/css/esri.css" />

<!-- Changed version number to 3.14 in the below script for the fix of support ticket #09085677 -->

<script src="//js.arcgis.com/3.14/" type="text/javascript" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/jquery-1.10.2.min.js')}" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/jquery-ui.min.js')}" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/handlebars.js')}" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/handlebars.helpers.js')}" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/encoder.js')}" />

<apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/prototype.js')}" />

<apex:stylesheet value="{!URLFOR($Resource.XedeDefaultAssets,'css/reset.css')}" />

<apex:stylesheet value="{!URLFOR($Resource.GISAssets,'css/checknet.css')}" />

<apex:stylesheet value="{!URLFOR($Resource.GISAssets,'visualforceBootstrap/css/bootstrap.css')}" />

<!-- load async via CDN instead of locally -->

<apex:includeScript value="{!URLFOR($Resource.GISAssets,'js/async.js')}" />

<!--<script src="//cdnjs.cloudflare.com/ajax/libs/async/2.5.0/async.min.js"></script>-->

<!-- Commented out XedeHttpClient.js and added inside X311GISMap -->

<!-- <apex:includeScript value="{!URLFOR($Resource.XedeDefaultAssets,'js/XedeHttpClient.js')}" /> -->

<style>

html, body,#map {

height: 100%;

width: 100%;

margin: 0;

padding: 0;

}

#force .alert {

padding: 0;

}

.feedback {

float:right;

}

#map {

width:78%;

border-right:1px solid #c9c9c9;

float: left;

}

.sidePanel{

width:21%;

float:left;

}

#container {

width:100%;

height: 98%;

}

#featureLayers {

border-bottom:1px solid #c9c9c9;

height: 400px;

}

#relatedItems,#featureLayers {

padding-top:20px;

}

#relatedItems {

height:500px;

overflow:auto;

}

.action-status-wrapper {

margin-left: 38%;

margin-right: auto;

position: absolute;

top: 50%;

width: 110px;

}

.infotable {

width: 96%;

margin: 6px;

}

.infotableproperty {

min-width: 120px;

max-width: 160px;

padding-right: 12px;

padding-top: 6px;

text-transform: capitalize;

}

</style>

<!--[if lte IE 7]>

<style>

#map\_zoom\_slider{

width : 62px !important;

}

</style>

<![endif]-->

<div id="container" class="claro">

<div id="map">

<c:ActionStatus />

</div>

<div id="force" class="sidePanel">

<div id="featureLayers">

</div>

<div id="relatedItems"></div>

</div>

</div>

<script id="layerTemplate" type="text/x-handlebars-template">

{{#each this}}

<label class="checkbox">

<input type="checkbox" value="{{@key}}"/>

{{@key}}

</label>

{{/each}}

</script>

<!-- Added alert message on onclick functionality for support ticket #10836325 at line number 133 -->

<script id="relatedTemplate" type="text/x-handlebars-template">

<ul class="unstyled">

{{#each this}}

<li class="alert alert-info">

<a target="\_blank" href="{!$Site.BaseUrl}/{{uniqueId}}">{{label}}</a> -- {{htmlDecode description}}

<br />

<br />

<div style="text-align:center">

<button class="btn btn-mini btn-primary" onclick="aGisApi.centerAndZoom({{x}},{{y}})">Map</button>

{{#is label "!==" "New Case"}}

<button class="btn btn-mini btn-primary" onclick="alert('{{label}} case has been related to the current case'); setParentId('{{uniqueId}}','{{label}}');">Relate</button>

{{/is}}

</div>

</li>

{{/each}}

</ul>

</script>

<c:FetchLayerInfoJS />

<c:GisMapJs />

<script type="text/javascript">

var $$=$.noConflict(); //Commented out the initialization and instead done in 311GISMap since that part of code is initialized first

var aGisApi = new GisApi();

var aGisPoint = null;

var map, points, currentId, prevGraphic;

// Added esri/layers/ArcGISDynamicMapServiceLayer and ArcGISDynamicMapServiceLayer in the require function below for the support ticket #09085677

require([

"esri/map",

"esri/geometry/Point",

"esri/geometry/Extent",

"esri/SpatialReference",

"esri/layers/ArcGISTiledMapServiceLayer",

"esri/layers/ArcGISDynamicMapServiceLayer",

"esri/graphic",

"esri/InfoTemplate",

"esri/symbols/SimpleMarkerSymbol",

"esri/symbols/Font",

"esri/symbols/TextSymbol",

"dojo/\_base/Color",

"dojo/number",

"dojo/dom",

"esri/tasks/GeometryService",

"esri/layers/FeatureLayer",

"dijit/ProgressBar",

"dojox/encoding/base64",

"dojo/domReady!" // wont return until the dom is ready - custom is to make it the last and not assign it

], function (

Map,

Point,

Extent,

SpatialReference,

ArcGISTiledMapServiceLayer,

ArcGISDynamicMapServiceLayer,

Graphic,

InfoTemplate,

SimpleMarkerSymbol,

Font,

TextSymbol,

Color,

number,

dom,

GeometryService,

FeatureLayer,

//Query,

ProgressBar,

Base64

) {

map = new esri.Map("map", {sliderOrientation : "horizontal" });

var baseLayer= new ArcGISTiledMapServiceLayer("{!$Setup.GIS\_\_c.BaseMaps\_\_c}", {

"opacity" : 0.5

});

map.addLayer(baseLayer);

map.on("zoom-end", function(evt){

setExtent(map.extent);

window.console && console.log('map extent:',map.extent);

});

map.on("pan-end", function(evt){

setExtent(map.extent);

});

map.on("update-start",function(evt){

actionStatusOn();

});

map.on("update-end",function(evt){

actionStatusOff();

});

//-- change infowindow size

map.infoWindow.resize(500,125);

var layerMap ={};

function addLayerURL(aStringOrStringArray) {

actionStatusOn('Fetching GIS Layers');

var aList = [].concat(aStringOrStringArray);

var newAList = [];

for(var i=0;i<aList.length; i++){

if(typeof aList[i] === 'string')

newAList.push(aList[i]);

}

var layerSource = $$("#layerTemplate").html();

var layerTemplate = Handlebars.compile(layerSource);

var aClient = new XedeHttpClient();

async.eachSeries(newAList, function(each, aCallback) {

var aUrl = each;

if(aUrl==undefined){

return;

}

if (each.length == 2) // the named urls are arrays with two parts -- brittle, but it works for now

aUrl = each[1];

if (aUrl.indexOf('?') >= 0)

aUrl += '&f=json';

else

aUrl += '?f=json';

var response = aClient.get(aUrl, function(response) {

var layerObject = JSON.parse(response);

if (layerObject.tileInfo) {

layerMap[each.length == 2 ? each[0] : layerObject.serviceDescription] = new ArcGISTiledMapServiceLayer(aUrl, {

"opacity" : 0.5

});

}

else if (layerObject.fields) {

layerMap[each.length == 2 ? each[0] : layerObject.name] = new FeatureLayer(aUrl, {

mode: FeatureLayer.MODE\_ONDEMAND,

"opacity" : 0.5,

outFields: '\*',

infoTemplate: new InfoTemplate(layerObject.name, formatFeatureHTML)

});

}

aCallback(null);

});

}, function(err) {

actionStatusOff();

window.console && console.log('layerMap: ', layerMap); // fb-02820

document.getElementById('featureLayers').innerHTML = layerTemplate(layerMap);

$$('input[type=checkbox]').on('click', function() {

var $$this = $$(this);

var selectedLayer = $$this.val()

if ($$this.is(':checked')) {

map.addLayer(layerMap[selectedLayer]);

} else {

map.removeLayer(layerMap[selectedLayer]);

}

});

});

}

// building the info template content is more problematic than we would

// normally encounter, because we have to include an onclick button to send

// an object, now base64 encoded, to the parent

function formatFeatureHTML(aGraphic) {

var tableHTML ='<table class="infotable">';

for (var eachField in aGraphic.attributes)

tableHTML += '<tr><td class="infotableproperty">{0}:</td><td>{1}</td></tr>'.format(eachField.replace(/\_/g, ' '), aGraphic.attributes[eachField]);

tableHTML += "</table> ";

// I got tired of trying to escape quotes and double quotes etc

// so I decided to just base64 encode the little pecker.

var aString = JSON.stringify(aGraphic.attributes);

var byteArray = [];

for (var i = 0; i < aString.length; i++)

byteArray.push(aString.charCodeAt(i));

var base64data = Base64.encode(byteArray);

return "<button onclick=\"setFeatureData('" + base64data + "');\">Set Feature Data</button>" + tableHTML;

}

function mapAddress() {

window.console && console.log('mapAddress()');

setTimeout(function(){

map.graphics.remove(prevGraphic);

},700);

actionStatusOn();

if (arguments.length < 2 || arguments.length > 3)

return;

var anAddress = arguments[0];

var pageCallback = arguments.length == 2 ? arguments[1] : arguments[2];

var aGeocodeType = arguments.length == 2 ? 'CenterlineOnly' : arguments[1];

aClient = new XedeHttpClient();

var safeAddress = encodeURIComponent(anAddress.replace('&', 'and'));

var responseList = [];

async.series([

// first we will get the standardized address

function(aCallback) {

actionStatusOn('Geocoding address');

GISController.Address(safeAddress, '4326',

function(result, event) {

if (event.status == false) {

aCallback(event.message);

} else {

window.console && console.log('standardized address', result);

if (result.successful == 'false') {

aCallback('Address can not be located');

}

else {

responseList.push(result.jmap);

aCallback(null, 'address');

}

}

},{escape:true, buffer:false}

);

},

// now get the service areas information

function(aCallback) {

//Filter is defined for older unsupported browsers like IE7,8

if (!Array.prototype.filter) {

Array.prototype.filter = function (fun) {

"use strict";

if (this === void 0 || this === null)

throw new TypeError();

var t = Object(this);

var len = t.length >>> 0;

if (typeof fun !== "function")

throw new TypeError();

var res = [];

var thisp = arguments[1];

for (var i = 0; i < len; i++) {

if (i in t) {

var val = t[i]; // in case fun mutates this

if (fun.call(thisp, val, i, t))

res.push(val);

}

}

return res;

};

}

if (responseList.length !== 1) {

aCallback(null, 'address ERROR');

return;

}

if (responseList[0].addresses === null) {

aCallback(null, 'missing addresses');

return;

}

var aList = responseList[0].addresses[0].links.filter(function(each) { return each.rel == 'parcels'});

actionStatusOn('Getting service areas');

GISController.HttpGet(

'{0}?format=json'.format(aList[0].href.replace('/parcels', '/service-areas')),

function(result, event) {

if (event.status == false) {

window.console && console.log(event);

alert(event.message);

aCallback(event);

} else {

window.console && console.log(result);

responseList.push(result.jmap);

aCallback(null, 'service-areas');

}

},{escape:true, buffer:false}

);

}

],

function(err, results){

window.console && console.log(err);

window.console && console.log(results);

window.console && console.log(responseList);

if (results[0] == "address") {

aGisPoint = new GisPoint(

responseList[0].addresses[0].xCoord,

responseList[0].addresses[0].yCoord,

label = 'New Case',

description = responseList[0].addresses[0].standardizedAddress,

null,

null,

pageCallback

);

aGisPoint.setGisData(responseList[0]);

// don't callback until after we project to 2272 -- we'll get

// new 2272 coordinates inside the GisPoint

fetchLayerInfo(aGisPoint, function(result) {

LocateXY(aGisPoint, function() {

aGisPoint.layerInfo = result;

aGisPoint.callback();

actionStatusOff();

});

});

}

else {

alert(err);

actionStatusOff();

}

if (results[1] == "service-areas") {

var aMap = {};

responseList[1].serviceAreaValues.forEach(function(each, index, list) {

// change related to GIS support ticket #11002316 START

if(index<=43)

{

aMap[each.serviceAreaId] = each.value;

}

//// change related to GIS support ticket #11002316 END

});

aGisPoint.setAttributes(aMap);

}

aGisPoint.callback();

});

}

function centerAPoint(aGisPointX,aGisPointY){

var anEsriPoint = new esri.geometry.Point(aGisPointX, aGisPointY, new SpatialReference({ wkid: 4326 }));

projectTo(map.spatialReference.wkid, anEsriPoint, function(result) {

anEsriPoint = result[0];

map.centerAndZoom(anEsriPoint, 8);

aCallback();

});

}

function LocateXY(aGisPointOrArray, aCallback) {

aGisPointOrArray.forEach(function (eachGisPoint, index, array) {

var anEsriPoint = new esri.geometry.Point(eachGisPoint.x, eachGisPoint.y, new SpatialReference({ wkid: 4326 }));

actionStatusOn('Projecting');

projectTo(map.spatialReference.wkid, anEsriPoint, function(pointList) {

actionStatusOff();

anEsriPoint = pointList[0];

eachGisPoint.x2272 = anEsriPoint.x;

eachGisPoint.y2272 = anEsriPoint.y;

var symbol = new SimpleMarkerSymbol();

var infoTemplate = new InfoTemplate(

eachGisPoint.description,

"Case Number: ${caseNumber}<br />Description: ${description}<br /><br />"

);

if (aGisPointOrArray instanceof GisPoint) {

symbol.setStyle(SimpleMarkerSymbol.STYLE\_DIAMOND);

symbol.setSize("32");

symbol.setColor(new Color("yellow"));

} else {

symbol.setStyle(SimpleMarkerSymbol.STYLE\_SQUARE);

symbol.setColor(new Color("blue"));

}

var attributes = {

caseNumber: eachGisPoint.label,

description: eachGisPoint.description,

id: eachGisPoint.uniqueId

};

var graphic = new Graphic(anEsriPoint, symbol, attributes, infoTemplate);

if (aGisPointOrArray instanceof GisPoint) {

graphic.id="highlight";

prevGraphic = graphic;

}

map.graphics.add(graphic);

// only zoom if we are mapping a single point

if (aGisPointOrArray instanceof GisPoint) {

map.centerAndZoom(anEsriPoint, 8);

}

aCallback();

});

});

var relatedSource = $$("#relatedTemplate").html();

var relatedTemplate = Handlebars.compile(relatedSource);

var listArray = [];

aGisPointOrArray.forEach(function(each, index, list) {

if (each instanceof GisPoint)

listArray.push(each);

});

if (listArray.length > 0)

document.getElementById('relatedItems').innerHTML = relatedTemplate(listArray);

window.console && console.log(listArray);

}

function projectTo(srId, aPoint, callback) {

gsvc = new GeometryService("{!$Setup.GIS\_\_c.ESRIServices\_\_c}/Geometry/GeometryServer");

gsvc.project([aPoint], new SpatialReference(srId), callback);

}

aGisApi.locateXY = LocateXY;

aGisApi.mapAddress = mapAddress;

aGisApi.EsriPoint = esri.geometry.Point;

aGisApi.addLayer = addLayerURL;

aGisApi.centerAndZoom = centerAPoint;

//window.parent.setGisApi(aGisApi);

});

function setParentId(id,label){

aGisPoint.setParentId(id);

aGisPoint.setParentLabel(label);

aGisPoint.callback();

}

function setFeatureData(value) {

aGisPoint.setFeatureData(value);

aGisPoint.callback();

}

function setExtent(value) {

aGisApi.setExtent(value);

}

</script>

</apex:component>

1. GisMapLeafLet

<apex:component >

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.1.0/dist/leaflet.css" />

<style>

html, body, #map {

height: 100%;

width: 100%;

margin: 0;

padding: 0;

}

#force .alert {

padding: 0;

}

.feedback {

float: right;

}

#map-container: {

/\*height: 100%;

width: 100%;\*/

position: absolute:

top: 0;

bottom: 0;

right: 0;

left: 0;

}

/\*#map {

width:78%;

border-right:1px solid #c9c9c9;

float: left;

height: 600px;

}\*/

.sidePanel{

width:21%;

float:left;

}

/\*#container {

width:100%;

height: 98%;

}\*/

#featureLayers {

border-bottom:1px solid #c9c9c9;

height: 400px;

}

#relatedItems, #featureLayers {

padding-top:20px;

}

#relatedItems {

height:500px;

overflow:auto;

}

.action-status-wrapper {

margin-left: 38%;

margin-right: auto;

position: absolute;

top: 50%;

width: 110px;

}

.infotable {

width: 96%;

margin: 6px;

}

.infotableproperty {

min-width: 120px;

max-width: 160px;

padding-right: 12px;

padding-top: 6px;

text-transform: capitalize;

}

</style>

<div id="container">

<div id="map-container">

<div id="map">

<c:ActionStatus />

</div>

</div>

<!--<div id="force" class="sidePanel">

<div id="featureLayers">

</div>

<div id="relatedItems"></div>

</div>

-->

</div>

<!-- JavaScript -->

<script src="//unpkg.com/leaflet@1.1.0/dist/leaflet.js"></script>

<script src="//unpkg.com/esri-leaflet@2.0.8"></script>

<script>

GIS.map = (function () {

var map,

geocodeMarker,

relatedMarkers;

return {

init: function () {

var MAP\_OPTIONS = {

// center over city hall

center: [39.952388, -75.1658127],

minZoom: 11,

zoom: 15,

};

// create map

map = L.map('map', MAP\_OPTIONS);

// move zoom control to bottom-right, a la google

map.zoomControl.setPosition('bottomright');

map.attributionControl.addAttribution('City of Philadelphia GIS Services Group');

// DEV

map.on('zoomend', function (e) {

console.log('zoomend', e.target.getZoom());

});

// add gsg city basemap with labels

// REVIEW add labels separately?

L.esri.tiledMapLayer({

url: '//tiles.arcgis.com/tiles/fLeGjb7u4uXqeF9q/arcgis/rest/services/CityBasemap\_Slash/MapServer'

}).addTo(map);

},

zoomTo: function (x, y, level) {

// default zoom level to 15

level = level || 15;

map.setView([y, x], level);

},

colorMarker: function (lat, lng, color) {

var AVAILABLE\_COLORS = [

'blue',

'red',

'green',

'orange',

'yellow',

'violet',

'grey',

'black',

];

// lower case, just in case

color = color.toLowerCase();

if (!AVAILABLE\_COLORS.includes(color)) {

throw 'Leaflet marker not available for color: ' + color;

}

var iconUrl = '//cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-' + color + '.png',

icon = new L.Icon({

iconUrl: iconUrl,

shadowUrl: '//cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',

iconSize: [25, 41],

iconAnchor: [12, 41],

popupAnchor: [1, -34],

shadowSize: [41, 41],

}),

marker = new L.marker([lat, lng], {icon: icon});

console.log('marker', marker);

return marker;

},

removeMarkers: function () {

if (geocodeMarker) {

map.removeLayer(geocodeMarker);

}

\_.forEach(relatedMarkers, function (relatedMarker) {

map.removeLayer(relatedMarker);

});

geocodeMarker = null;

relatedMarkers = [];

},

addGeocodeMarker: function (x, y, address) {

console.log('add geocode marker', x, y);

MAP.removeMarkers();

//geocodeMarker = L.marker([y, x]);

geocodeMarker = this.colorMarker(y, x, 'yellow');

geocodeMarker.bindPopup('<strong>' + address + '</strong>');

geocodeMarker.addTo(map);

geocodeMarker.openPopup();

},

addRelatedMarkers: function (relateds) {

console.log('add related markers', relateds);

\_.forEach(relateds, function (related) {

//console.log('related', related);

var x = related.x,

y = related.y,

marker = this.colorMarker(y, x, 'blue');

marker.addTo(map);

relatedMarkers.push(marker);

}.bind(this));

},

};

})();

// call initMap on dom ready

$$(MAP.init);

</script>

</apex:component>

Static Resource:

1. GisAppJs

var GIS = (function () {

var $searchInput,

$searchAction,

$multipleResultsModal,

$multipleResultsSelect,

$multipleResultsButton,

ANIMATION\_DELAY = 250; // milliseconds

return {

config: GIS\_CONFIG,

// initial state. this is really only here to spec out what the state looks

// like.

// REVIEW should initial values be null, since they get reset to null on

// each ais search?

state: {

// this is the ais feature that matched the user's search

aisFeature: {},

// this gets used when there are multiple matching ais features

multipleResults: [],

// this stores related data, such as service areas

layerInfo: {},

},

init: function () {

// configure proj4 to use local state plane projection

proj4.defs('EPSG:2272', '+proj=lcc +lat\_1=40.96666666666667 +lat\_2=39.93333333333333 +lat\_0=39.33333333333334 +lon\_0=-77.75 +x\_0=600000 +y\_0=0 +ellps=GRS80 +datum=NAD83 +to\_meter=0.3048006096012192 +no\_defs ');

$searchInput = $('#search-input');

$searchAction = $('#search-action');

// check for search input from parent page

if (window.opener) {

var searchInput = window.opener.getSearchInput();

$searchInput.val(searchInput);

// if there's search input, "click" the locate button

if (searchInput.length > 0) {

GIS.handleSearchButtonClick();

}

}

// listen for 'enter' key on address input

$searchInput.on('keyup', function (e) {

if (e.keyCode == 13) {

GIS.handleSearchButtonClick();

}

});

// if there's something in the search input

$('#search-button').click(GIS.handleSearchButtonClick);

/\*

MULTIPLE RESULTS MODAL

\*/

$multipleResultsModal = $('#multiple-results-modal');

$multipleResultsSelect = $multipleResultsModal.find('select');

$multipleResultsButton = $multipleResultsModal.find('button');

// callbacks

$('.modal-close').click(function (e) {

$multipleResultsModal.fadeOut(ANIMATION\_DELAY);

});

$('.modal').click(function (e) {

$multipleResultsModal.fadeOut(ANIMATION\_DELAY);

});

$('.modal-content').click(function (e) {

e.stopPropagation();

});

// $multipleResultsSelect.change(function (e) {

$multipleResultsButton.click(function (e) {

// get index of selected option

var i = $multipleResultsSelect.find('option:selected').index();

// pass selected address to ais callback

// (this is not ideal but ais doesn't currently have a way to validate

// a single address)

var aisFeature = GIS.state.multipleResults[i],

featureData = {

features: [aisFeature],

};

GIS.didFetchAis(featureData);

// empty dropdown

$multipleResultsSelect.empty()

// hide modal

$multipleResultsModal.fadeOut(ANIMATION\_DELAY);

// clear out state

GIS.state.multipleResults = null;

});

},

setSearchAction: function (action) {

// append ellipsis

action && (action += '...');

// set text

$searchAction.text(action);

// show search status if there's an action

$('#search-status').toggle(!!action);

},

toCamelCase: function (s) {

return s.replace(

/\w+/g,

function (w) {

return w[0].toUpperCase() + w.slice(1).toLowerCase();

}

);

},

// handle clicks to the locate button

handleSearchButtonClick: function (e) {

GIS.setSearchAction('Validating address');

var inputAddress = $searchInput.val();

GIS.fetchAis(inputAddress);

},

// this function gets user input and queries the Address Information System. this is

// intended to replace the Address Candidate function.

fetchAis: function (inputAddress) {

// console.log('fetch ais', inputAddress);

var url = GIS.config.ais.baseUrl + encodeURIComponent(inputAddress);

// clear out state

GIS.state.aisFeature = null;

GIS.state.layerInfo = null;

$.ajax({

url: url,

data: GIS.config.ais.params,

success: GIS.didFetchAis,

error: GIS.didNotFetchAis,

});

},

// on successful ais fetch

didFetchAis: function (data) {

// console.log('did fetch ais', data);

GIS.setSearchAction(null);

// check number of results

var features = data.features,

resultCount = Array.isArray(features) ? features.length : 0;

if (resultCount === 1) {

var feature = features[0];

// WORKAROUND ais returns pipe-delimited address keys, so split them

// and take the first one

var addressKeysJoined = feature.properties.li\_address\_key || '',

addressKeys = addressKeysJoined.split('|'),

addressKey = addressKeys.length > 0 ? addressKeys[0] : null;

feature.properties.li\_address\_key = addressKey;

GIS.didGetValidAddress(feature);

// if we got multiple results

} else if (resultCount > 1) {

GIS.state.multipleResults = features;

// show dropdown

GIS.showMultipleResults(features);

// if we got no results

} else {

console.log('ais successful but no features');

alert('Could not locate that address. Please try another search.');

}

},

// on unsuccessful ais fetch

didNotFetchAis: function (jqXHR, textStatus, errorThrown) {

console.log('error fetching ais:', errorThrown, jqXHR);

GIS.setSearchAction(null);

var msg;

if (errorThrown === 'NOT FOUND') {

msg = 'Could not locate that address. Please try another search.'

} else {

msg = 'Error while validating address: ' + errorThrown;

}

alert(msg);

},

showMultipleResults: function (features) {

$multipleResultsSelect.empty();

\_.forEach(features, function (feature) {

var address = GIS.addressForFeature(feature),

$option = $('<option>').text(address);

$multipleResultsSelect.append($option);

});

$multipleResultsModal.fadeIn(ANIMATION\_DELAY);

},

// returns the fully-standardized street address for an ais feature

addressForFeature(feature) {

var props = feature.properties,

featureType = feature.ais\_feature\_type,

address;

// form address based on the feature type

switch(featureType) {

case 'address':

address = props.street\_address;

break;

case 'intersection':

var comps = [

props.street\_1.street\_full,

props.street\_2.street\_full

],

address = comps.join(' & ');

break;

default:

throw new Error('Unhandled feature type: ' + featureType);

break;

}

return address;

},

didGetValidAddress: function (feature) {

// console.log('did get valid address', feature);

var coords = feature.geometry.coordinates,

x = coords[0],

y = coords[1],

address = GIS.addressForFeature(feature);

// add marker and zoom

GIS.map.addGeocodeMarker(x, y, address);

GIS.map.zoomTo(x, y);

// save to state

GIS.state.aisFeature = feature;

// if it's an address, get layer info

if (feature.ais\_feature\_type === 'address') {

var layerInfo = GIS.fetchLayerInfo(feature, null);

// otherwise call the remaining two functions

} else {

// pass data back to parent

GIS.saveGeodata();

// get related/nearby cases

GIS.fetchRelated();

}

},

fetchLayerInfo: function (feature, caseType) {

// console.log('fetch layer info', feature, caseType);

GIS.setSearchAction('Querying layers');

//var GIS\_BASE\_URL = '{!$Setup.GIS\_\_c.ESRIServices\_\_c}' + '/',

var queryLayers = GIS.config.queryLayers,

queryLayerFns = {};

// build up an array of query fns for each layer

\_.forOwn(queryLayers, function (queryLayerDef, queryLayerName) {

//console.log('query layer', queryLayerName, queryLayerDef);

var queryLayerFn = function (callback) {

//var endpoint = queryLayerDef.endpoint,

var url = queryLayerDef.url,

type = queryLayerDef.type,

options = queryLayerDef.options;

switch(type) {

case 'esri':

// form params

var params = {

outFields: '\*',

returnGeometry: 'true',

f: 'json'

},

whereFn = options.where,

where = whereFn(feature);

params.where = where;

$.ajax({

url: url + '/query',

data: params,

success: function (dataStr) {

// parse json string

var data = $.parseJSON(dataStr);

// check for error

if (data.error) {

console.log('did get error for where query', queryLayerName, data);

callback(true);

return;

}

//console.log('did get data for where query', queryLayerName, data, url);

// unpack attributes object

var features = data.features,

feature = Array.isArray(features) && features.length > 0 ? features[0] : null,

result;

if (feature) {

result = feature.attributes;

console.log('set', queryLayerName, result);

}

callback(null, result);

},

error: function () {

console.log('did get esri error');

callback(queryLayerName + ' had error');

}

});

break;

case 'carto':

var qFn = options.q,

q = qFn(feature),

params = {

q: q,

};

$.ajax({

url: url,

data: params,

success: function (data) {

// unpack attributes

var rows = data.rows,

row = Array.isArray(rows) && rows.length > 0 ? rows[0] : null;

callback(null, row);

},

error: function () {

console.warn('did get carto error');

callback(queryLayerName + ' had error');

},

});

break;

default:

console.warn('unhandled query type: ' + type);

callback("unhandled query type '" + type + "'", null);

break;

}

};

queryLayerFns[queryLayerName] = queryLayerFn;

});

// execute query fns in parallel

async.parallel(

queryLayerFns,

// then do this

function (err, results) {

// check for an error

if (err) {

alert('An error occurred while querying layers. Please try again.');

console.log('async parallel error:', err);

return;

}

// console.log('async parallel finished', results);

// save to state

GIS.state.layerInfo = results;

// pass data back to parent

GIS.saveGeodata();

// get related/nearby cases

GIS.fetchRelated();

}

);

},

// this takes an optional argument `parentCase` which is used when relating

// cases. parentCase should be an object with two keys: id and caseNumber.

// id is the guid of the case, and caseNumber is the shorter numeric id.

saveGeodata: function (parentCase) {

console.log('save geodata', feature, layerInfo);

var state = GIS.state,

feature = state.aisFeature,

layerInfo = state.layerInfo;

// parentId = state.parentId;

// pass data back to parent window

var geodata = GIS.createGeodataObject(feature, layerInfo, parentCase);

console.log('geodata', geodata);

// DEBUG: if window.opener isn't defined, we're probably just testing this

// from the X311GISMap page directly. don't try to call saveGeodata on the parent.

if (!window.opener) {

console.warn('window.opener not defined. geodata will not be saved.');

return;

}

window.opener.saveGeodata(geodata);

},

// takes an ais feature and formats it as a "geodata" object which the parent window can understand.

createGeodataObject: function (feature, layerInfo, parentCase) {

console.log('create geodata object', feature, layerInfo);

var props = feature.properties,

coords = feature.geometry.coordinates,

coordsStatePlane = proj4('EPSG:2272', coords),

// parent page checks for the token `Intersection`, so camel case this

matchType = GIS.toCamelCase(feature.ais\_feature\_type),

addressKey = (props.li\_address\_key || '').split('|')[0],

parentCase = parentCase || {},

parentId = parentCase.id,

label = parentCase.num;

// we're now getting the L&I district from AIS

if (layerInfo) {

layerInfo.district = props.li\_district;

}

var geodata = {

gisData: {

standardizedAddress: GIS.addressForFeature(feature),

matchType: matchType,

// address keys are pipe-delimited, so take the first one

addressKey: addressKey,

// REVIEW is it ok to use the opa address here?

liAddress: props.opa\_address,

zoning: props.zoning,

},

// this doesn't seem to be used

description: null,

// this doesn't seem to be used

featureData: null,

// this gets passed in when "relating" one case to another

parentId: parentId,

// this is the number that populates the parent case field when

// relating

label: label,

x: coords[0],

y: coords[1],

x2272: coordsStatePlane[0],

y2272: coordsStatePlane[1],

layerInfo: layerInfo,

// this seems to be where service area values go

attributes: {

// this is not a service area in ais, so fake it like this.

SA\_STREETS\_ZipCode: props.zip\_code,

}

};

// map service areas

var serviceAreas = GIS.config.serviceAreas;

\_.forOwn(serviceAreas, function (aisKey, ulrsKey) {

var val = props[aisKey];

geodata.attributes[ulrsKey] = val;

});

return geodata;

},

fetchRelated: function () {

// console.log('fetch related');

GIS.setSearchAction('Locating nearby cases');

var feature = GIS.state.aisFeature,

coords = feature.geometry.coordinates,

x = coords[0],

y = coords[1],

RADIUS = GIS.config.nearbyRadius;

// get "related" cases (recent, nearby cases with the same record type)

var recordTypeId = window.opener && window.opener.RECORD\_TYPE\_ID;

// DEBUG: if no parent window, use record type id for Dangerous Sidewalk

if (!recordTypeId) recordTypeId = '012G00000014Gt6IAE';

// get cases with the same record type within a radius

GisClient.QueryByRadius(recordTypeId, x, y, RADIUS, GIS.didFetchRelated);

},

didFetchRelated: function (results, event) {

// console.log('did fetch related', results, event);

GIS.setSearchAction(null);

// put markers on map

GIS.map.addRelatedMarkers(results);

},

};

})();

$(GIS.init);

1. GisConfigJs

var GIS\_CONFIG = {

// this is the radius within which to search for nearby (aka "related") cases

// (in feet)

nearbyRadius: 500,

// Address Information System config. this is the api that handles geocoding.

ais: {

// TODO put this in a global variable

baseUrl: '//api.phila.gov/ais/v1/search/',

params: {

include\_units: true,

on\_street: true,

opa\_only: true,

// TODO put this in a global variable

gatekeeperKey: 'cb076d7301c315ba0f039b8f7744ac6d',

},

},

// these are the layers that related data get pulled from

queryLayers: {

demoPermit: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Construction/MapServer/18',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " AND APPLICATION\_TYPE = 'BP\_DEMO'";

}

}

},

dumpsterPrivate: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Business/MapServer/8',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " AND TYPE = '3230'";

},

},

},

familyDaycare: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Business/MapServer/8',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " AND TYPE = '3397'";

},

},

},

foodLicense: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Business/MapServer/6',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

healthCenter: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Health\_Centers/MapServer/0',

options: {

where: function (feature) {

// note: this layer doesn't have standardized addresses, so there's no

// guarantee this query will work. adding some logic to improve matching.

return "UPPER(FULL\_ADDRESS) LIKE '" + feature.properties.street\_address + "%'";

}

}

},

hospital: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Hospitals/MapServer/0',

options: {

where: function (feature) {

return "UPPER(ADDRESS) LIKE '" + feature.properties.street\_address + "%'";

}

}

},

property: {

type: 'carto',

url: '//phl.carto.com/api/v2/sql',

options: {

q: function (feature) {

var accountNum = feature.properties.opa\_account\_num;

return "SELECT \* FROM opa\_properties\_public WHERE parcel\_number = '" + accountNum + "'";

},

},

},

rentalLicense: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/RentalLicense/MapServer/2',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

rentalViolation: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/RentalLicense/MapServer/0',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

vacancy: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Vacancy/MapServer/2',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

violationProperty: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Violations/MapServer/0',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key + " and VIOLATION\_STATUS <> 'NOT COMPLIED'";

}

}

},

zoningPermit: {

type: 'esri',

url: '//gis.phila.gov/arcgis/rest/services/PhilaGov/Construction/MapServer/12',

options: {

where: function (feature) {

return 'ADDRKEY = ' + feature.properties.li\_address\_key;

}

}

},

},

// this is a mapping of ULRS-style service area names to AIS ones

serviceAreas: {

SA\_PLANNING\_2016Councilmanic: 'council\_district\_2016',

SA\_PLANNING\_Ward: 'political\_ward',

SA\_PLANNING\_Ward\_Divisions: 'political\_division',

// this is not a service area in ais

// SA\_STREETS\_ZipCode: '',

SA\_POLICE\_District: 'police\_district',

SA\_POLICE\_Division: 'police\_division',

SA\_STREETS\_Recycling: 'rubbish\_recycle\_day',

SA\_STREETS\_Rubbish: 'rubbish\_recycle\_day',

SA\_STREETS\_Rubbish\_Recyc: 'rubbish\_recycle\_day',

SA\_STREETS\_Leaf: 'leaf\_collection\_area',

SA\_STREETS\_HISTORIC: 'historic\_street',

SA\_SCHOOLS\_Elementary\_School\_Catchment: 'elementary\_school',

SA\_SCHOOLS\_Middle\_School\_Catchment: 'middle\_school',

SA\_SCHOOLS\_High\_School\_Catchment: 'high\_school',

SA\_STREETS\_Highway\_District: 'highway\_district',

SA\_STREETS\_Highway\_Section: 'highway\_section',

SA\_STREETS\_Highway\_Subsection: 'highway\_subsection',

SA\_PLANNING\_Planning\_Districts: 'planning\_district',

SA\_PWD\_CenterCityDistrict: 'pwd\_center\_city\_district',

SA\_CENTER\_CITY\_DISTRICT: 'center\_city\_district',

SA\_STREETS\_Recycling\_Diversion\_Rate: 'recycling\_diversion\_rate',

SA\_STREETS\_Sanitation\_Area: 'sanitation\_area',

SA\_STREETS\_Sanitation\_Districts: 'sanitation\_district',

SA\_STREETS\_Street\_Lights\_Routes: 'street\_light\_route',

SA\_Streets\_Traffic\_District: 'traffic\_district',

SA\_Streets\_Traffic\_PM\_District: 'traffic\_pm\_district',

SA\_PHILLYRISING: 'philly\_rising\_area',

PWD\_MAINT\_DIST: 'pwd\_maint\_district',

PWD\_PRES\_DIST: 'pwd\_pressure\_district',

PWD\_WTPSA: 'pwd\_treatment\_plant',

SA\_LNI\_DISTRICT: 'li\_district',

SA\_POLICE\_PSA: 'police\_service\_area',

SA\_WATER\_Water\_Plate\_Index: 'pwd\_water\_plate',

LadderLocal: 'ladder\_local',

EngineLocal: 'engine\_local',

// don't think these exist in ais

// SA\_PLANNING\_2000CensusTract,

// SA\_PLANNING\_2000CensusBlockGroup

// SA\_PLANNING\_2000CensusBlock

// SA\_PLANNING\_2000CensusBlock

// SA\_PLANNING\_2010CensusBlockGroup

// SA\_PLANNING\_2010CensusBlockGroup

// SA\_PLANNING\_2010CensusBlockGroup

// SA\_RECREATION\_Recreation\_District

}

};

1. GisMapJs

GIS.map = (function () {

var map,

spiderfier,

geocodeMarker,

relatedMarkers,

POPUP\_OPTIONS = {

minWidth: 130,

};

return {

init: function () {

var MAP\_OPTIONS = {

// center over city hall

center: [39.952388, -75.1658127],

minZoom: 11,

zoom: 15,

};

// create map

map = L.map('map', MAP\_OPTIONS);

// move zoom control to bottom-right, a la google

map.zoomControl.setPosition('bottomright');

map.attributionControl.addAttribution('City of Philadelphia | CityGeo');

// DEBUG

// map.on('zoomend', function (e) {

// console.log('zoomend', e.target.getZoom());

// });

// add gsg city basemap with labels

// REVIEW add labels separately?

L.esri.tiledMapLayer({

url: '//tiles.arcgis.com/tiles/fLeGjb7u4uXqeF9q/arcgis/rest/services/CityBasemap\_Slash/MapServer',

maxZoom: 22,

}).addTo(map);

// create spiderfier (this is an object that expands, aka "spiderfies"

// overlapping map markers when clicked on)

spiderfier = new OverlappingMarkerSpiderfier(map);

spiderfier.addListener('click', function (marker) {

var popup = marker.popup;

popup.setLatLng(marker.getLatLng());

map.openPopup(popup);

});

// listen for relate clicks

$(document).on('click', '.relate-link', GIS.map.handleRelateClick);

},

zoomTo: function (x, y, level) {

level = level || 18;

map.setView([y, x], level);

},

colorMarker: function (lat, lng, color) {

var AVAILABLE\_COLORS = [

'blue',

'red',

'green',

'orange',

'yellow',

'violet',

'grey',

'black',

];

// lower case, just in case

color = color.toLowerCase();

if (!AVAILABLE\_COLORS.includes(color)) {

throw 'Leaflet marker not available for color: ' + color;

}

var iconUrl = '//cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-' + color + '.png',

icon = new L.Icon({

iconUrl: iconUrl,

shadowUrl: '//cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',

iconSize: [25, 41],

iconAnchor: [12, 41],

popupAnchor: [1, -34],

shadowSize: [41, 41],

}),

marker = new L.marker([lat, lng], {icon: icon});

// console.log('marker', marker);

return marker;

},

removeMarkers: function () {

if (geocodeMarker) {

map.removeLayer(geocodeMarker);

}

\_.forEach(relatedMarkers, function (relatedMarker) {

map.removeLayer(relatedMarker);

});

geocodeMarker = null;

relatedMarkers = [];

},

addGeocodeMarker: function (x, y, address) {

// console.log('add geocode marker', x, y);

GIS.map.removeMarkers();

geocodeMarker = this.colorMarker(y, x, 'yellow');

var popupContent = '\

<div>\

<strong>' + address + '</strong>' + '\

</div>\

<div>\

<a href="http://cityatlas.phila.gov/#/' + address + '/" target="\_blank">\

Internal Map For SF Users <span class="fa fa-external-link"></span>\

</a>\

</div>\

<div>\

<a href="https://atlas.phila.gov/#/' + address + '/" target="\_blank">\

External Map For Partners Users <span class="fa fa-external-link"></span>\

</a>\

</div>\

',

popup = new L.Popup(POPUP\_OPTIONS).setContent(popupContent);

geocodeMarker.popup = popup;

geocodeMarker.addTo(map);

spiderfier.addMarker(geocodeMarker);

},

addRelatedMarkers: function (relateds) {

// console.log('add related markers', relateds);

\_.forEach(relateds, function (related) {

var serviceType = related.Service\_Request\_Type\_\_c;

var lat = related.Centerline\_\_Latitude\_\_s,

lng = related.Centerline\_\_Longitude\_\_s,

marker = GIS.map.colorMarker(lat, lng, 'blue'),

// description = related.description,

// popup = description + '<br><a href="#" class="related-link">Relate</a>';

caseId = related.Id,

caseNumber = related.CaseNumber,

caseType = related.Subject,

address = related.Street\_\_c,

popupContent = '\

<div>' +

'<strong>' + address + ' : ' + caseType + ' - ' + serviceType + '</strong>\

</div>\

<div>\

Case #<a target="\_blank" href="//philly311.my.salesforce.com/' + caseId + '" target="\_blank">' + caseNumber + '</a>' +

'</div>\

<div>\

<a class="relate-link"\

href="#"\

data-case-id="' + caseId + '" \

data-case-number="' + caseNumber + '"\

>\

Relate\

</a>\

</div>\

';

marker.popup = new L.Popup(POPUP\_OPTIONS)

.setContent(popupContent);

marker.addTo(map);

spiderfier.addMarker(marker);

relatedMarkers.push(marker);

});

},

handleRelateClick: function (e) {

// console.log('handle relate click', e);

// get ids

var $link = $(this),

caseId = $link.attr('data-case-id'),

caseNumber = $link.attr('data-case-number'),

caseObj = {

id: caseId,

num: caseNumber

};

// to "relate" one case to another, we set the parent id to the related's

// case id.

GIS.saveGeodata(caseObj);

alert('This case has been related to case #' + caseNumber + '.');

},

};

})();

// call initMap on dom ready

$(GIS.map.init);