Build Your Own Maps with Big Data Discovery using a Custom Visualization Component

BIWA Summit 2016

Chris Hughes
BI and Spatial Specialist

Oracle Public Sector
Jan 19, 2016
Agenda

1. Overview
2. Recipe Ingredients
3. JS Code Review
4. Demonstration
5. Conclusion
1. Overview

Architecture
BDD 1.1 Improvements
Issue Statement
CVC Overview
1. Overview

4th Generation **Data Architecture** for Public Sector

**Business Data**
- Data Streams
- Social and Log Data
- Enterprise Data
- Other Data Sources

**Data Streaming**
- Events
- Decisions

**Data Management**
- Reservoir
- Data Factory
- Warehouse

**Discovery Lab**
- Search
- Data Quality
- Modeling

**Analytics**
- Business Monitoring
- Business Intelligence

**Dashboards**
- Telematics
- Industry Services
- Internet of Things
- Sentiment

**Data Services**
- Reporting-oriented
- Often enterprise wide in scope, cross LoB
  - “you know the questions to ask”

**Discovery**
- Data Exploration
- Highly visual and/or interactive
  - “you don’t know the questions to ask”
1. Overview

Big Data Discovery 1.1 Applications:

• **Beautiful data**
  – Attractive interactive data visualizations democratize data analysis
  – Easy-to-use custom visualization component unlocks endless extensions

• **JDBC data**
  – Mash up trusted enterprise data with unstructured big data

• **Live data**
  – Full refreshes, sample expansions, and incremental updates
  – All transformations, enrichments, joins, and visualizations update with the data

• **Safe data**
  – Dataset-level security makes it practical to roll out BDD to larger analyst communities
1. Overview

Issue Statement

1. Many Public Sector Agencies manage assets across a set of locations and geography eg:
   - Environmental health
   - Citizens in need of service (health, safety, public utility, etc)
   - Transportation Infrastructure

2. Public Sector (and Resource Extraction Companies) agencies need to visualize and analyze their data in multiple spatial contexts

3. Oracle Big Data Discovery tools only support a basic set of geographic visualizations

4. The broader open source community can create visualizations faster than Oracle development can

Now users and partners can create any visualization they want, quickly and easily:

Introducing the CVC in Big Data Discovery
1. Overview

What is a CVC? *Custom Visualization Component*

- **Goal**: allow fast authoring and publishing of any visualization
  - Innovative projects require innovative visualizations
- **Minimize time to use**
  - Tweak the JS from `bl.ocks.org` and other sites, or write your own
  - Only JS and SQL-like language
  - No archives, Java, or complicated deployment
- **Publish as a first-class component**
  - Looks and feels to end users like any other component
1. Overview

What is a CVC? 3 Key Components

1. EQL statement(s) to acquire the data to visualize

2. A collection of "token" definitions that define variables used in the EQL statements and dictate how those values are acquired

3. A JavaScript class that uses the Custom Visualization Component (CVC) JavaScript API to execute the queries and transform the query response into a graphical representation of the data.
1. Overview

What is in the CVC Framework?

• BDD-CVC JavaScript Library
• Client Side Portlet scaffolding
• Management Interface in Control Panel (need admin rights on BDD server)
• Simple JS code editor
• Resident d3 and JQuery JS libraries
  – Assume Sencha/ExtJS is available too
• Sample JS Code and EQL
1. Overview

Data Notes: Reused AVL data from 2015 BIWA Preso

- Data repository in Oracle Spatial + Graph
- Nearest neighbor and point in poly operations on each position
2. Recipe Ingredients

Documentation
Favorite JS Library
2. Ingredients

Key Tools in Developing your own CVC

1. Awesomely written CVC tutorial by BDD Product Management:

2. Downloadable CVC Example from Tech Network

```javascript
/**
 * fit map to
 * layer
 */
myMap.fitBounds(myLayer.getBounds());
```

Ability to Understand JS development and DOM

Thousands of Different visualizations to float your boat and integrate in BDD Studio
2. Ingredients

10 Steps to Develop a Mapping CVC

1. Download the CVC Tutorial
2. Download the Sample Donut CVC
3. Implement the Donut CVC in your BDD environment
4. Test/Manipulate the EQL in the Donut CVC Sample
5. Identify the JavaScript library you want to use (need a CDN or web reachable URL for the library)
6. Create some test HTML pages to test map/visualization events and functions
7. Review API docs (CVC, d3, Leaflet)
8. Following the CVC Tutorial: build methods and properties for your custom class
9. Iterate: Publish and Test
10. Refine CVC after use within Studio
2. Ingredients

Step 1: Copy others Work and claim it as your own

Process Overview
Creating a Custom Visualization Component in Studio’s Control Panel

1. In the Studio header, click the Configuration Options icon and select Control Panel.

2. From the Control Panel select Custom Visualizations.

• Read the CVC Tutorial
• Good information on:
  • DOM manipulation - Rendering the Visualization (p 26)
  • Color Scales (p31)
  • Portlet messaging (p35)
  • Error Handling (p34)
  • General Workflow (p20)
2. Ingredients

Step 2: Copy others Work and claim it as your own

- Download Sample “Donut” Code
- Install in to your BDD Environment
- Test with your data
2. Ingredients

Step 5: Identify Your JS Library (Leaflet)

Leaflet Quick Start Guide

This step-by-step guide will quickly get you started on Leaflet basics, including setting up a Leaflet map, working with markers, polylines and popups, and dealing with events.

Leaflet Notes

- [http://leafletjs.com/](http://leafletjs.com/)
- Quick tile/slippy map with minimal js code
- Multiple tile providers for maps (like mapbox)
- Multiple tutorials and example to be found

Other Libs to Consider:
- Oracle Map View v2
- D3 has map plugins as well
2. Ingredients

Step 8: Add methods and properties

- Note that you are extending the BaseRenderer Class
- You must have an `init` method
- Think about what common properties and methods you want
- Don’t forget Error Handling 😊
3. JS Code Review

Init

RenderMarkers 1 and 2
3. JS Code Review

Init Block

Leaflet

Review and edit the JavaScript code that powers your visualization. Modifications affect all instances of this visualization in all projects.

```javascript
  map: {},
  markerGroup: {},
  polyLayer: {},

  init: function() {
    //defaults for query params
    var parkDistThreshold = "100", numRecs = "2000";

    /**
     * Get the queryConfig for the avlpark query
     */
    RETURN data AS SELECT
    "longitude" AS lng, "latitude" AS lat,
    %metric_1% AS metric,
    %groupby_1% AS groupby
    FROM "%dataview%"
    WHERE "park_dist" <= %park_limit%
    GROUP BY groupby, "longitude", "latitude"
    ORDER BY metric %sort%
    PAGE(0,%num_rec%);
    */
    var queryConfig = this.getQueryConfig("avlpark");
```
3. JS Code Review

Rendering Block

Leaflet

Review and edit the JavaScript code that powers your visualization. Modifications affect all instances of this visualization in all projects.

```javascript
/**
 * Callback function to render layer in leaflet map
 */
renderMarkers: function(queryResults) {
    var self = this;

    /**
     * test centroid for the map
     */
    var initLL = new L.LatLng(33, -114);

    // Remove any existing markers
    this.markerGroup.clearLayers();

    // var canvas = this.getCanvas({
    //     tagName: 'div',
    //     margins: {top: 0, bottom: 0, left:10, right:10}
    // });

    var isRendered = canvas.isRendered();
    var root = d3.select(canvas.getRoot());

    if (!isRendered) {
        // attribution for tile layers
        var mbAttr = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors, ' +
```
Oracle.BDD.Portlets.Visualization.Renderers.Leaflet =
Oracle.BDD.Portlets.Visualization.Renderers.BaseRenderer.extend({
    map: {},
    markerGroup: {},
    polyLayer: {},
    init: function() {
        //defaults for query params
        var parkDistThreshold = "100", numRecs = "2000";

        /**
         * Get the queryConfig for the avlpark query
         *
         
         RETURN data AS SELECT
         "longitude" AS lng, "latitude" AS lat,
         %metric_1% AS metric,
         %groupby_1% AS groupby
         FROM ":%dataview%"
         WHERE "park dist" <= %park_limit%
         GROUP BY groupby, "longitude", "latitude"
         ORDER BY metric %sort%
         PAGE(0,%num_recs%);
         */
        var queryConfig = this.getQueryConfig("avlpark");
3. JS Code Review

Full Code 2 of 6 (init func)

```
//this is the collection of markers as a Leaflet overlay
this.markerGroup = L.featureGroup();

/**
 * Set default value for num_recs data token
 */
if (!queryConfig.getToken("num_recs").getValue()) {
    queryConfig.getToken("num_recs").setValue(numRecs);
}

/**
 * Set default value for park_dist data token
 */
if (!queryConfig.getToken("park_limit").getValue()) {
    queryConfig.getToken("park_limit").setValue(parkDistThreshold);
}

/**
 * Execute the initial query, indicating renderMarkers() as the callback
 */
this.executeQuery(queryConfig, true, this.renderMarkers);
```

/**
 * Callback function to render layer in leaflet map
 */
renderMarkers: function(queryResults) {
    var self = this;

    /**
     * test centroid for the map
     */
    var initLL = new L.LatLng(33, -114);
    // Remove any existing markers
    this.markerGroup.clearLayers();

    // canvas = this.getCanvas({
    // tagName: 'div',
    // margins: {top: 0, bottom: 0, left:10, right:10}
    // });

    var isRendered = canvas.isRendered();
    var root = d3.select(canvas.getRoot());
if (!isRendered) {
  // attribution for tile layers
  var mbAttr = 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetMap</a> contributors, ' +
    '<a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +
    'Imagery ? <a href="http://mapbox.com">Mapbox</a>\n';

  // key tile URL pattern - v3 is not current as of Jan 2016
  var mbUrl = 'https://{s}.tiles.mapbox.com/v3/{id}/{z}/{x}/{y}.png';

  // tile layer definitions from mapbox
  var grayscale = L.tileLayer(mbUrl, {id: 'examples.map-20v661k', attribution: mbAttr}),
      light = L.tileLayer(mbUrl, {id: 'examples.ik7djhcc', attribution: mbAttr}),
      streets = L.tileLayer(mbUrl, {id: 'examples.map-zr0njcqy', attribution: mbAttr});

  // array to hold the tile layer options
  var baseLayers = {
    "Grayscale": grayscale,
    "Streets": streets,
    "Light": light
  };

  // add a div element to the root CVC canvas
  var mapDiv = root.append('div')
    .attr('class', 'cvc-leaflet-map')
    .attr('id', 'map');
//initialize the Leaflet map
this.map = L.map('map', {center: initLL, zoom: 5, layers: [light, this.markerGroup]});
var overlays = { "AVL Points": this.markerGroup};
L.control.layers(baseLayers, overlays, {position:'bottomleft'}).addTo(this.map);

/**
 * Get references to the data records and the queryConfig that defined the query.
 */
var records = queryResults.getRecords();
var queryConfig = queryResults.getQueryConfig();

/**
 * Get default color manager
 */
var colorMgr = this.getColorManager();

/**
 * Iterate Results and add to layerGroup
 * [{groupby: "France", metric: 5.14}]
 */
3. JS Code Review

Full Code 6 of 6 (renderMarkers func)

```javascript
for (var i = 0; i < records.length; i++) {
    var a = records[i];
    var title = a.groupby;
    var metric = a.metric;
    var marker = L.circleMarker(L.latLng(a.lat, a.lng),
                                 {className:'circle-point', color:colorMgr.getColor(a.groupby),
                                  radius: 5 + (metric + 1)/2})
        .bindPopup('Group:' + title + '<br/>Metric:' + metric);
    this.markerGroup.addLayer(marker);
}

//fit map to layer
this.map.fitBounds(this.markerGroup.getBounds());
```
4. Demonstration
5. Conclusion

What I’d like to do
Recommendations for You
5. Conclusion

Next Steps and Tests: What I didn’t do

• Use MapViewer
  – Is it installed with BDD by default?

• Are there other JS libs we can install with BDD Studio?

• Test GeoRSS or GeoJSON from EQL to add custom layers
  – Any libs to do this in Groovy/Transform?

• Use refine CVC Methods
  – Click on a park and automatically add refinement in Studio

• Use analytic Point Clustering plugin
5. Conclusion

Some Notes and Hints for You

What to Watch Out For:

– Sometimes the server will blow away your JS code
– Manipulating the object in studio will fire a lot of repainting/resizing events
– The snapshot function doesn’t seem to work with CVC portlets (it’s a bug IIRC)
– IMO real polygon or linear map features will need to be stored in Hadoop and rendered through EQL as GeoJSON or GeoRSS

Recommendations:

– Re-read the CVC Tutorial
– Read d3 blog and tutorials
– Read the EQL syntax guide
– Save your js code locally
– Save your settings locally not unlike the CVC sample
– Chrome and Ctrl+Shift+J is your best friend who tells you what is really going on