Client Side Clustering Documentation

Developed by: Ricky Brundritt Date: February 14th, 2011

**Table of Contents**

[Description 2](#_Toc301450056)

[ClusteredEntityCollection Class 2](#_Toc301450057)

[Constructor 2](#_Toc301450058)

[Cluster Options 2](#_Toc301450059)

[Public Methods 3](#_Toc301450060)

[Public Enumerators 3](#_Toc301450061)

[Example 3](#_Toc301450062)

[ClusterPlacementTypes Enumerator 4](#_Toc301450063)

# Description

This document is meant to accompany the V7ClientSideClustering.js JavaScript file. This JavaScript file is meant to create a client side clustering algorithm that has high performance, is reusable, and easy to extend. This method takes in a map reference and a set of cluster options.

The **SetData** method takes in an array of objects. Every object is expected to have a **Latitude** and **Longitude** property. This algorithm adds a property to your data called a **GridKey**. This is useful in relating the raw data back to its associated shape and vice versa. The Pushpin objects are assigned a **GridKey** property as well as an **isClustered** property which property which are used to quickly determine if a shape if clustered and what data it represents. This is also useful when handling mouse over events.

# ClusteredEntityCollection Class

The **ClusteredEntityCollection** class is an optimized clustering algorithm that has several additional built in functions to make handling clustered data easier. The expected general use case is to take data that is coming in from an AJAX call as JSON and be able to take the array of data objects and add it to the map to be clustered in one step. In addition to clustering this class, this class also has tool’s that make it easier to relate pins back to their raw data and raw data back to the pin that is currently being used to represent it on the map. This takes the logic of how a shape should be built outside of the algorithm which makes it easier to customize. In addition there are three built in methods of placing clustered pins so as to customize the experience. There is also an option to offset the clustered layer. This is useful when multiple layers are on a map. By offsetting a layer you reduce the likelihood of clustered pushpins being overlapped such that one pushpin is unreachable.

## Constructor

|  |
| --- |
| var myLayer = new ClusteredEntityCollection(map, options); |

|  |  |
| --- | --- |
| Parameter | Description |
| map | A Microsoft.Maps.Map object that the cluster algorithm is going to be used with. |
| options | A cluster option object that the data will be displayed on. |

## Cluster Options

When creating an instance of the **ClusteredEntityCollection** it is possible to specify options. These options allow you to customize the clustering algorithm.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| clusterPlacementType | int | The type of cluster placement method to use when representing a clustered location. |
| gridSize | int | The size of a grid cell in pixels. |
| clusteringEnabled | bool | A flag that indicates if clustering is enabled or not. All data that is in the map view is represented as a single pushpin. |
| layerOffset | Point | Offsets the layer by a specified amount of pixels. Only works when using the “center” cluster placement method. |
| singlePinCallback | function | A function that takes in a data object and returns a pushpin. |
| clusteredPinCallback | function | A function that takes in an array of data and a latlong coordinate then returns a clustered pushpin. |
| callback | function | A callback function that is called when the clustering algorithm has completed. This |

## Public Methods

|  |  |
| --- | --- |
| Name | Description |
| BringLayerToFront | Sets all layer's z-index to 0 and increases the z-index of the current layer to 1. |
| GetData | Gets the data array and returns it to the user. |
| GetDisplayedData | Returns all data that is the current map view. |
| GetDataByGridKey | Returns an array of data that is in a particular grid cell. A grid key is used to determine this. Optionally a boolean can be passed to indicate if the data is clustered or not. This would be known if retrieving the data based on a reference from its associated pin. |
| GetPinByGridKey | Returns a Pushpin that is in a grid cell. A grid key is used to determine this. |
| GetLayer | Returns a reference to the EntityCollection being used. |
| GetOptions | Returns the clustering options. |
| SetOptions | Sets the clustering options. |
| SetData | Sets the data that is to be clustered and displayed on the map. |

## Public Enumerators

|  |  |
| --- | --- |
| Name | Description |
| ClusterPlacementTypes | Represents the type of method to be used to place a clustered pushpin. |

## Example

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html>  <head>  <title>Client Side Clustering</title>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>  <!-- Add a reference to the Bing Maps Control -->  <script charset="UTF-8" type="text/javascript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>    <!-- This is a reference to a class for generating test data. Not needed in production apps. -->  <script type="text/javascript" src="scripts/TestDataGenerator.js"></script>  <script type="text/javascript">  var map, myLayer, infobox;  function GetMap() {  map = new Microsoft.Maps.Map(document.getElementById("myMap"),  { credentials: "Your\_Bing\_Maps\_Key" });  //Register and load the Client Side Clustering Module  Microsoft.Maps.registerModule("ClientSideClusteringModule", "scripts/V7ClientSideClustering.js");  Microsoft.Maps.loadModule("ClientSideClusteringModule", { callback: function () {  myLayer = new ClusteredEntityCollection(map, {  singlePinCallback: createPin,  clusteredPinCallback: createClusteredPin  });  });  }  function createPin(data) {  var pin = new Microsoft.Maps.Pushpin(data.\_LatLong);  return pin;  }  function createClusteredPin(cluster, latlong) {  var pin = new Microsoft.Maps.Pushpin(latlong, { text: '+' });  return pin;  }  //Makes a request for data  function RequestData() {  var size = parseInt(document.getElementById('dataSize').value);  TestDataGenerator.GenerateData(size, RequestDataCallback);  }  function RequestDataCallback(response) {  if (response != null) {  myLayer.SetData(response);  }  }  </script>  </head>  <body onload="GetMap();">  <div id='myMap' style="position:relative; width:100%; height:500px;"></div>  <br />  Data Size: <input type="text" id="dataSize" />  <input type="button" value="Get Mock Data" onclick="RequestData();" />  </body>  </html> |

# ClusterPlacementTypes Enumerator

This enumerator is used to specify the type of calculation to be used when calculating the location of a cluster of locations.

|  |  |
| --- | --- |
| Name | Description |
| MeanAverage | Mean Average placement calculates the average position of a group of coordinates. This method creates a more realistic representation of the data, however it may require more processing power depending on the size of data being clustered. This method also increases the chances of pushpins overlapping |
| GridCenter | This method is one of the simplest methods and works fast although it may not accurately represent the data. |
| FirstLocation | This method is the simplest way to represent a cluster. It places the cluster at the first location in the cluster. This method may not accurately represent the data but requires little processing power. |