[Loading GeoRSS-Feeds with Dynamic Modules in the Bing Maps AJAX Control v7](http://jkebeck.wordpress.com/2011/05/29/loading-georss-feeds-with-dynamic-modules-in-the-bing-maps-ajax-control-v7/" \o "Permalink to Loading GeoRSS-Feeds with Dynamic Modules in the Bing Maps AJAX Control v7)

Author: Johannes Kebeck

Source: <http://jkebeck.wordpress.com/>

# Introduction

In May several Updates have been released to the Bing Maps AJAX Control v7 which Keith Kinnan has summarized [here](http://blogs.msdn.com/b/keithkin/archive/2011/05/16/update-to-bing-maps-ajax-control-v7.aspx). Possibly the most significant one is the ability to dynamically load additional modules.

When the Bing Maps AJAX Control had been re-written for v7 some of the design goals had been to improve performance and to support mobile devices. In order to achieve these goals the Bing Maps AJAX control supports now HTML5 and it was also put on a diet in order to slim down to a size that can load quickly even on mobile devices. As a consequence the core-control itself supports only the minimum requirements that most mapping sites will have. However, the control was designed in such a way that additional modules can be optionally and dynamically added later on when needed. The general principal is explained in the [SDK](http://msdn.microsoft.com/en-us/library/hh125837.aspx) and in the [interactive SDK](http://www.bingmapsportal.com/ISDK/AjaxV7) you will find an example that implements client-side clustering. This module for client-side clustering adds about 14kB to the weight of the website and while it is a useful feature for some not everybody might need it and therefore it seems to be a good idea to stick it into a module and let the developer decide if and when he needs it.

Another feature that may be helpful for some is the ability to import GeoRSS-feeds. In this blog-post we will have a look at the steps to create such a custom module and load it on demand. To limit the amount of code for this blog-post we parse only GeoRSS-feeds following the [Simple serialization](http://georss.org/simple). However a similar approach could be used to parse [GeoRSS-feeds derived from GML](http://georss.org/gml) as well as GPX- or KML-files.

# GeoRSS-Feed

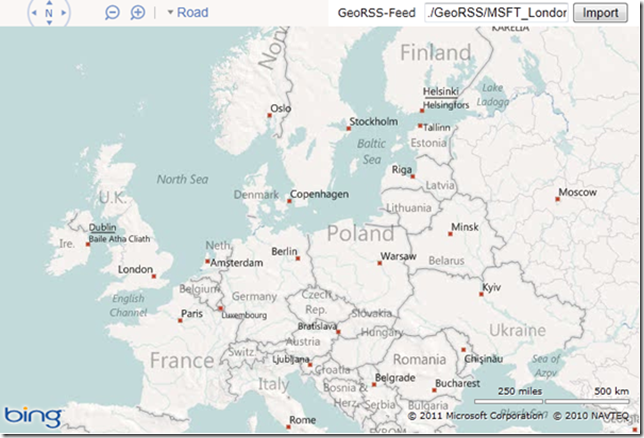
A GeoRSS-feed following the [Simple serialization](http://georss.org/simple) contains tags and to describe points (<georss:point>), lines () and polygons () as a sequence of latitudes and longitudes. The example below is a GeoRSS-feed that contains a polygon representing the Microsoft Office in London, a point representing the nearest underground station and a line representing the walk from the tube station to the Microsoft Office.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <rss xmlns:georss="http://www.georss.org/georss" version="2.0">  <channel>  <title>Microsoft London</title>  <link>http://www.bing.com/maps</link>  <description />  <language>en-gb</language>  <item>  <title>Microsoft</title>  <link>http://www.bing.com/maps/?cid=42E1F70205EC8A96!14501</link>  <description>Cardinal Place, 100 Victoria Street, London, SW1E 5JL</description>  <guid isPermaLink="false">e7aea3b0d2e1c7b8</guid>  <pubDate>Fri May 27 22:37:13 UTC 0100 2011</pubDate>  <georss:polygon>51.49673999638715 -0.14145107778161137 51.496997150067386 -0.1394555142745313 51.49772184808794 -0.1397022775039014 51.497568226428456 -0.1408288052901563 51.49676003438838 -0.1414403489455518 51.49673999638715 -0.14145107778161137</georss:polygon>  </item>  <item>  <title>Victoria</title>  <link>http://www.bing.com/maps/?cid=42E1F70205EC8A96!14501</link>  <description>Underground Station</description>  <guid isPermaLink="false">b03dc79bd7bdb81e</guid>  <pubDate>Fri May 27 22:37:39 UTC 0100 2011</pubDate>  <georss:point>51.49644610469038 -0.14391334565724278</georss:point>  </item>  <item>  <title>Walk from Victoria to Cardinal Place</title>  <link>http://www.bing.com/maps/?cid=42E1F70205EC8A96!14501</link>  <description>180m</description>  <guid isPermaLink="false">bf8ee4e437813477</guid>  <pubDate>Fri May 27 22:38:24 UTC 0100 2011</pubDate>  <georss:line>51.496643145923684 -0.14391334565724278 51.496506219055234 -0.14225574048603917 51.49657969206017 -0.1415208152159586 51.4967199583771 -0.14146180661763097</georss:line>  </item>  </channel>  </rss> |

We will use the location tags to draw points, lines and polygons and the text within the title and description tags as content for the InfoBox that pops up when we click on the object.

# The Website

On the website we have a simple map and add a text-box to enter the path to the GeoRSS-feed as well as a button to load the module and import the GeoRSS-feed.

[](http://jkebeck.files.wordpress.com/2011/05/image.png)

The code so far is shown below.

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html>  <head>  <title></title>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <script type="text/javascript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>  <script type="text/javascript">  var map = null;  var MM = Microsoft.Maps;  function GetMap() {  var options = { credentials: "YOUR BING MAPS KEY",  enableClickableLogo: false,  enableSearchLogo: false,  mapTypeId: Microsoft.Maps.MapTypeId.road,  center: new MM.Location(51.4611794944098, -0.9259434789419174),  zoom: 17  };  map = new MM.Map(document.getElementById('mapDiv'), options);  // Hide the info box when the map is moved.  MM.Events.addHandler(map, 'viewchange', hideInfobox);  }  </script>  </head>  <body onload="GetMap();">  <div id='mapDiv' style="position:relative; width:800px; height:600px;"></div><br />  <a>GeoRSS-Feed</a><input id="txtGeoRSS" type="text" value="MSFT\_London.xml" />  <input id="Button2" type="button" value="Import" onclick="LoadModule()" />  </body>  </html> |

We add now to the website a function to [register and dynamically load an additional module](http://msdn.microsoft.com/en-us/library/hh125837.aspx). To register the module we give it a unique name and point to the location of the JavaScript that contains the module. When we load the module we can optionally specify a callback-function that is being executed when the loading is completed.

|  |
| --- |
| function LoadModule() {  // Register and load a new module  MM.registerModule("SimpleGeoRSSModule", "scripts/SimpleGeoRSSModule.js");  MM.loadModule("SimpleGeoRSSModule", { callback: ModuleLoaded });  } |

# The Module

The module is a basically a separate JavaScript-file that we can register on demand in our website and that can make use of the namespace Microsoft.Maps. In this module we start by defining the style of lines and polygons.

|  |
| --- |
| function GeoRSSModule(map) {  var myFillColor = new Microsoft.Maps.Color(100,255,165,0);  var myStrokeColor = new Microsoft.Maps.Color(200,255,165,0);  var myStrokeThickness = 5;  var myPolygonOptions={fillColor: myFillColor,  strokeColor: myStrokeColor,  strokeThickness: myStrokeThickness};  var myPolylineOptions={strokeColor: myStrokeColor,  strokeThickness: myStrokeThickness}; |

Next we extend the Pushpin, Polyline and Polygon classes in the namespace Microsoft.Maps with properties that can hold the title and description for these objects. For Polylines and Polygons we also add properties that can hold the position where we want the InfoBox to appear.

|  |
| --- |
| Microsoft.Maps.Pushpin.prototype.title = null;  Microsoft.Maps.Pushpin.prototype.description = null;  Microsoft.Maps.Polyline.prototype.title = null;  Microsoft.Maps.Polyline.prototype.description = null;  Microsoft.Maps.Polyline.prototype.anchorLat = null;  Microsoft.Maps.Polyline.prototype.anchorLon = null;  Microsoft.Maps.Polygon.prototype.title = null;  Microsoft.Maps.Polygon.prototype.description = null;  Microsoft.Maps.Polygon.prototype.anchorLat = null;  Microsoft.Maps.Polygon.prototype.anchorLon = null; |

A module can have one or more functions and for our module the main logic is implemented in the function ImportGeoRSS. Before we load the GeoRSS-feed we first remove all other entities from the map.

|  |
| --- |
| this.ImportGeoRSS = function (MyFeed) {  map.entities.clear(); |

Next we load the GeoRSS-feed.

|  |
| --- |
| var xmlhttp = new XMLHttpRequest();  xmlhttp.open("GET", MyFeed, false);  xmlhttp.send();  var xmlDoc = xmlhttp.responseXML; |

In the following part we parse the XML-feed into objects of type Microsoft.Maps.Pushpin, Polyline and Polygon – including the additional properties that we prototyped before.

|  |
| --- |
| var itemCount = xmlDoc.getElementsByTagName("item").length;  var allLocs = new Array()  for (i = 0; i <= itemCount - 1; i++) {  var childNodeCount = xmlDoc.getElementsByTagName("item")[i].childNodes.length;  var tagName = null;  var geomType = null;  var geom = null;  var myTitle = null;  var myDesc = null;  var anchorLat = null;  var anchorLon = null;  for (j = 0; j <= childNodeCount - 1; j++) {  tagName = xmlDoc.getElementsByTagName("item")[i].childNodes[j].nodeName;  if (tagName in { 'georss:point': '', 'georss:line': '', 'georss:polygon': '' }) {  geomType = tagName;  geom = xmlDoc.getElementsByTagName("item")[i].childNodes[j].childNodes[0].nodeValue;  }  else if (tagName == "title") {  try {  myTitle = xmlDoc.getElementsByTagName("item")[i].childNodes[j].childNodes[0].nodeValue;  }  catch (err) {  }  }  else if (tagName == "description") {  try {  myDesc = xmlDoc.getElementsByTagName("item")[i].childNodes[j].childNodes[0].nodeValue;  }  catch (err) {  }  }  }  var coords = new Array();  coords = geom.split(" ");  var thisLocs = new Array()  var anchorCoord = null;  if ((coords.length/2) % 2) {  anchorCoord = coords.length / 2-1;  }  else {  anchorCoord = coords.length / 2;  }  for (k = 0; k <= coords.length - 1; k = k + 2) {  var thisLoc = new Microsoft.Maps.Location(coords[k], coords[k + 1]);  thisLocs.push(thisLoc);  allLocs.push(thisLoc);  if (k == anchorCoord) {  anchorLat = coords[k];  anchorLon = coords[k + 1];  }  }  var shape = null;  switch (geomType) {  case "georss:point":  shape = new Microsoft.Maps.Pushpin(thisLocs[0]);  break;  case "georss:line":  shape = new Microsoft.Maps.Polyline(thisLocs, myPolylineOptions);  shape.anchorLat = anchorLat;  shape.anchorLon = anchorLon;  break;  case "georss:polygon":  shape = new Microsoft.Maps.Polygon(thisLocs, myPolygonOptions);  shape.anchorLat = anchorLat;  shape.anchorLon = anchorLon;  break;  }  shape.title = myTitle;  shape.description = myDesc; |

We also attach an event to the object that will show an InfoBox with further information before we add the object to the map.

|  |
| --- |
| pushpinClick = Microsoft.Maps.Events.addHandler(shape, 'click', showInfoBox);  map.entities.push(shape);  } |

When all objects are added to the map we set the map-view to a zoom-level and centre-point that shows all objects.

|  |
| --- |
| map.setView({ bounds: Microsoft.Maps.LocationRect.fromLocations(allLocs) });  }  } |

Finally we signal back to the map that the module is now loaded and trigger the execution of the callback-function.

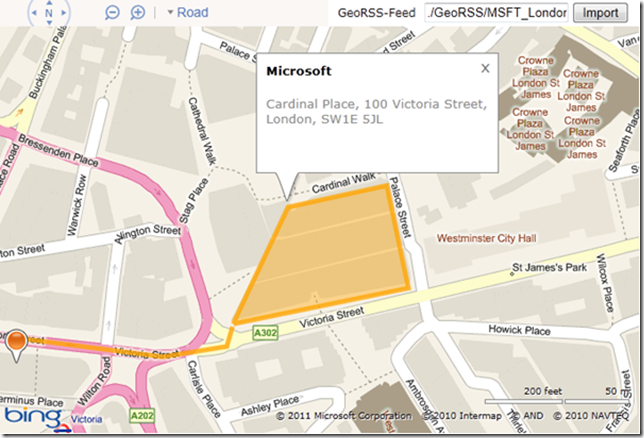
|  |
| --- |
| Microsoft.Maps.moduleLoaded('SimpleGeoRSSModule'); |

# Back to the Website

We had already prepared the website with a function to load the module but we still have to add the callback-function. In this callback-function we execute the ImportGeoRSS-function on a feed as specified in the text-box.

|  |
| --- |
| function ModuleLoaded() {  // Use the function provided by the newly loaded module  var myModule = new GeoRSSModule(map);  myModule.ImportGeoRSS(document.getElementById("txtGeoRSS").value);  collectionInfoBox = new MM.EntityCollection;  map.entities.push(collectionInfoBox);  } |

Finally we add some code to handle the InfoBoxes and we’re done.

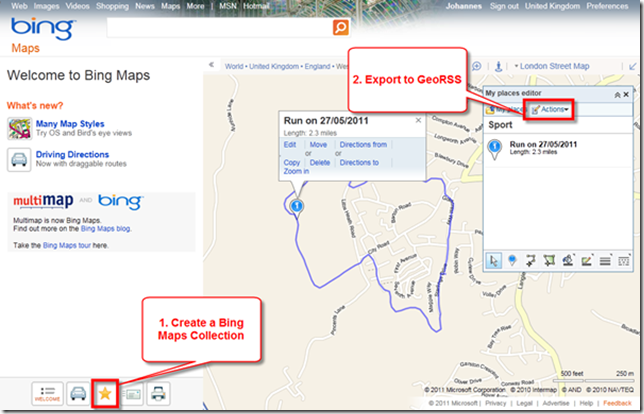
[](http://jkebeck.files.wordpress.com/2011/05/image1.png)

The complete source of our website is listed below. To see the code in action follow [this link](http://talkingdonkey.cloudapp.net/BM-AJ-V7/Default.aspx) and select “GeoRSS” under the accordion-pane “Miscellaneous”.

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html>  <head>  <title></title>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <script type="text/javascript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"></script>  <script type="text/javascript">  var map = null;  var MM = Microsoft.Maps;  var infobox = null;  var collectionInfoBox = null;  function GetMap() {  var options = { credentials: "YOUR BING MAPS KEY",  enableClickableLogo: false,  enableSearchLogo: false,  mapTypeId: Microsoft.Maps.MapTypeId.road,  center: new MM.Location(51.4611794944098, -0.9259434789419174),  zoom: 17  };  map = new MM.Map(document.getElementById('mapDiv'), options);  // Hide the info box when the map is moved.  MM.Events.addHandler(map, 'viewchange', hideInfobox);  }  function LoadModule() {  // Register and load a new module  MM.registerModule("SimpleGeoRSSModule", "scripts/SimpleGeoRSSModule.js");  MM.loadModule("SimpleGeoRSSModule", { callback: ModuleLoaded });  }  function ModuleLoaded() {  // Use the function provided by the newly loaded module  var myModule = new GeoRSSModule(map);  myModule.ImportGeoRSS(document.getElementById("txtGeoRSS").value);  collectionInfoBox = new MM.EntityCollection;  map.entities.push(collectionInfoBox);  }  //Display InfoBox  function showInfoBox(e) {  if (e.targetType == "pushpin") {  collectionInfoBox.clear();  infobox = new MM.Infobox(e.target.getLocation(), { title: e.target.title, description: e.target.description, offset: new MM.Point(0, 30), visible: true });  collectionInfoBox.push(infobox);  }  else if (e.targetType == "polygon" || e.targetType == "polyline") {  collectionInfoBox.clear();  infobox = new MM.Infobox(new MM.Location(e.target.anchorLat, e.target.anchorLon), { title: e.target.title, description: e.target.description, offset: new MM.Point(0, 0), visible: true });  collectionInfoBox.push(infobox);  }  }  function hideInfobox(e) {  try {  infobox.setOptions({ visible: false });  }  catch (err) {  }  }  </script>  </head>  <body onload="GetMap();">  <div id='mapDiv' style="position:relative; width:800px; height:600px;"></div><br />  <a>GeoRSS-Feed</a><input id="txtGeoRSS" type="text" value="MSFT\_London.xml" />  <input id="Button2" type="button" value="Import" onclick="LoadModule()" />  </body>  </html> |

# Tip: Importing Bing Maps Collections in your own Website

On the [Bing Maps consumer site](http://www.bing.com/maps/) you can create your own collections. If you wanted to use such a collection in your own websites you can simply use the “My Places Editor” to export to GeoRSS and then use the module above to import it.

[](http://jkebeck.files.wordpress.com/2011/05/image2.png)