Ovi Maps API - How to embed an Ovi Maps on your webpage and show a map marker for a stored GPS location

Under Construction: (20110101) This article is under construction and it may have outstanding issues. If you have any comments please use the comments tab.

Introduction

This article will explain how to create a web page with a basic Ovi Maps embedded and further, how to put a map marker on the geocoordinates (latitude and longitude) saved previously to a file in Wiki article no (_link referring to the Wiki article_). The coordinates were sent from a mobile device running a web runtime widget. The widget is included and downloadable from within the same Wiki article.

Prerequisites

- A mobile device with Web-Runtime support
- A web server with a valid location data file available and readable
- Java Platform Standard ed. (5.x)

Embedding Ovi Maps in your web page

```
HTML <HEAD>

<head>
<script src="http://api.maps.ovi.com/jsl.js" type="text/javascript" charset="utf-8"></script>
</head>
```

Add a div for the map:

```
<body>
<p><center><div id="map" style="height: 360px; width: 640px;"></div></center></p>
```

Start the script and first add the map instance.

```
<script type="text/javascript">


map = new ovi.mapsapi.map.Display(document.getElementById("map"),
  {
      components: [ new ovi.mapsapi.map.component.Behavior(),
                    new ovi.mapsapi.map.component.ZoomBar(),
                    new ovi.mapsapi.map.component.Overview(),
                    new ovi.mapsapi.map.component.TypeSelector(),
                    new ovi.mapsapi.map.component.ScaleBar() ],
      zoomLevel: 3,
                                  eventListener: myListener,
      center: [52.51, 13.4]  // default map center point

  });
```

The function showuseronmap()

```
function showuseronmap()
{
  var my_longitude = 0;
  var my_latitude = 0;
  var temp = "";
  var data = new Array();
  var i = 0;
  var datafile = window.location.href.substring(0,
      window.location.href.lastIndexOf("/")+1)+"inc/testFile.txt";

  var url = new java.net.URL(datafile);
  var connect = url.openConnection();

  var input = new java.io.BufferedReader(new java.io.InputStreamReader(connect.getInputStream()));
  var aLine = "";
```

In the beginning of the script we will set several variables to hold data. The datafile variable points to the file on server in which the location data (latitude and longitude) values are to be read from.

The java.net.URL class represents a URL. There are constructors to create new URLs and methods to parse the different parts of a URL. However the heart of the class are the methods that allow you to get an InputStream from a URL so you can read data from a server. We are utilizing this by providing the class a reference to the datafile variable to define the target for file reading.

(C) Copyright Nokia 2013. All rights reserved.

```
while ((aLine = input.readLine()) != null) {
     data [i++] = aLine;
}
```

This file loop reads the input file line by line until there are no more lines to be read.

```
for (var j = 0; j < data.length ; j++)
     temp += data[j] + "<br>";
     tilap = data[0].split(' ');
```

The following for-loop will iterate through the lines read. It will split each line in two parts, separated a blank space (' '). It is therefore important that the input file has the following format and content e.g.:

32.10000<SPACE>64.000001

The first number, value of geolocation latitude, is separated from the longitude by a blank space.

Basically, there can be many location lines one file, but this example only supports handling of one location line. This would be easily extended to multiple locations to output multiple markers on the map, if needed.

```
// Show a map marker on mylocation

       putMeOnTheMap(parseFloat(tilap[0]),parseFloat(tilap[1]));
       }
       function putMeOnTheMap(p_latitude, p_longitude)
       {
        var MyMarker = [{type: "marker",
      longitude: p_longitude,
      latitude: p_latitude,
                                clickable: true,
      infoTitle: "My last known location",
      infoDescription: p_longitude +","+p_latitude
        }];
```

This part of the code defines a map marker for the Ovi Maps and utilizes the latitude and longitude values we had fetched previously. The values from the file are in string format. Therefore we need to convert them to numeric, to be used as the MyMarker parameters respectively. This is achieved simply by calling the putMeOnTheMap() function using the parseFloat() function. The only function for it is to parses a string and return a floating point number. These values are then passed on to the putMeOnTheMap() function.

```
<TBD>

}
</script>
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.