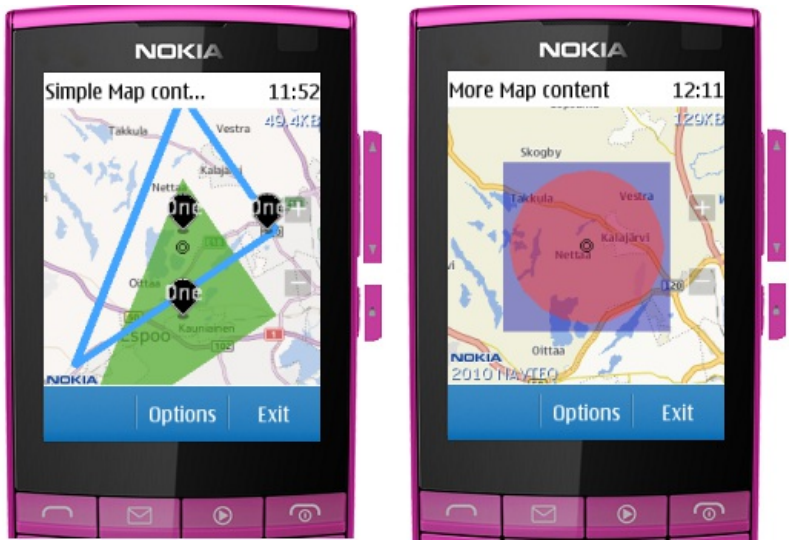
Content Handling with Maps API for Java ME

This article explains how to add different types of content to the Map when using the HERE Maps API for Java ME. Removing content and other content functionality is also dealt with in this article.

13 Oct 2013

Introduction



The HERE Maps API for Java ME makes it possible to easily embed the HERE Maps service into Java ME applications.

`MapMarkers` have already been discussed in the Adding Markers to the Map with Maps API for Java ME article, thus this article will concentrate on other types of content that may be added to the map.

Adding content to the map

In addition to `MapMarkers` the following content types have class definition included with the HERE Maps API for Java ME:

- `Polygon`,
- `Polyline`,
- `Circle`, and
- `Rectangle`

`Polylines` can be constructed by using the `MapFactory`, the constructor takes only one argument, which is an array of points for the `Polyline`. after construction you can use the methods provide by the `MapPolyline` class to modify it, for example you can change the color of the line with `setColor()` method.

```
GeoCoordinate[] polylinCoord = new GeoCoordinate[4];
polylinCoord[0] = new GeoCoordinate(60.27, 24.81, 0);
polylinCoord[1] = new GeoCoordinate(60.35, 24.70, 0);
polylinCoord[2] = new GeoCoordinate(60.19, 24.57, 0);
polylinCoord[3] = new GeoCoordinate(60.27, 24.81, 0);

MapPolyline polyline = mapCanvas.getMapFactory().createMapPolyline(polylinCoord);
polyline.setColor(0xFF43A5FF);

mapCanvas.getMapDisplay().addMapObject(polyline)
```

In a similar manner to `Polylines`, `Polygons` are constructed via the `MapFactory`, and the constructor also takes the array of points:

```
GeoCoordinate[] polygonCoord = new GeoCoordinate[3];
polygonCoord[0] = new GeoCoordinate(60.22, 24.81, 0);
```


(C) Copyright Nokia 2014. All rights reserved.

```
polygonCoord[1] = new GeoCoordinate(60.30, 24.70, 0);
polygonCoord[2] = new GeoCoordinate(60.14, 24.57, 0);

MapPolygon polygon = mapCanvas.getMapFactory().createMapPolygon(polygonCoord);
polygon.setColor(0xAA43A51B);

mapCanvas.getMapDisplay().addMapObject(polygon);
```

The `Circle` and `rectangle` are basically special polygons helper classes. For the `Circle` the constructor takes two arguments, which are the `radius` and `center`. And for the `Rectangle` the only argument taken is bounding box for the rectangle, which is basically area defined by two points, left top point and the bottom right point.

```
MapCircle myCircle = mapCanvas.getMapFactory().createMapCircle(5000.0,new
GeoCoordinate(60.30, 24.70,0));
// ...
GeoBoundingBox rectnaglebox = new GeoBoundingBox(new GeoCoordinate(60.35, 24.60,0), new
GeoCoordinate(60.25, 24.80,0));
MapRectangle rectnagle = mapCanvas.getMapFactory().createMapRectangle(rectnaglebox);
```

Z-ordering content

The Z-ordering content is handled simply by using the `setzIndex()` method for each content that needs to be ordered. Giving a lower value in `setzIndex()` will make the content to be drawn under the content that has higher values. For example if you want `myCircle` to be drawn on top of the `rectangle`, you can do it with following code:

```
int markerZ = 1;
int polyZ = 0;
myCircle.setzIndex(markerZ);
rectangle.setzIndex(polyZ);
```

Hiding content without deleting them

Each content item can be hidden by calling the `setVisible()` method with a `Boolean` argument set to `false` :

```
boolean vissible = !rectnagle.isVisible();
rectnagle.setVisible(vissible);
```

Removing content

Each content added to the Map display with the `addMapObject()` method can be also removed by using the `removeMapObject()` method.

```
mapCanvas.getMapDisplay().addMapObject( myCircle );
```

Thus, if for example, `myCircle` is added with the code above, it can be later on removed from the Map by using the code shown below.

```
mapCanvas.getMapDisplay().removeMapObject(myCircle);
```

Resources

(C) Copyright Nokia 2014. All rights reserved.

Full source code illustrating usage for map content types discussed in this article is available at: File:Java MapContent.zip

Summary

The HERE Maps API for Java ME offers rich functionality which facilitates the integration of all the main HERE Maps features into a Java ME application, with just a few lines of code.

(C) Copyright Nokia 2014. All rights reserved.