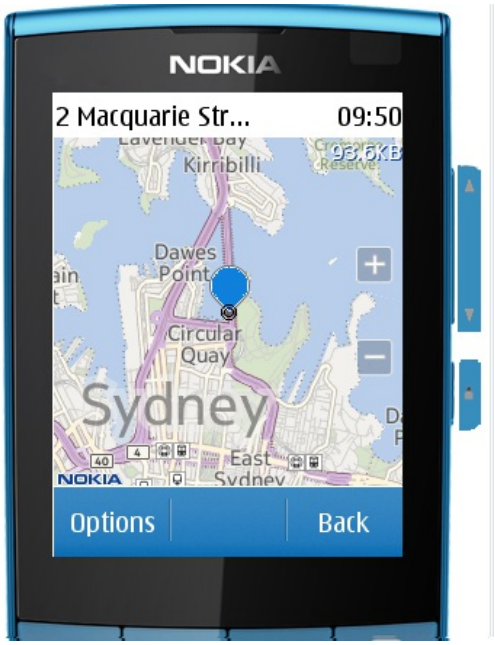
How to display an Address or a Point of Interest on a Map with Java ME

This article describes how to search for an address or a point of interest and display it on the Map. It uses the Map API for Java ME for the translation of the search string into valid coordinates and for the retrieval of the relevant map image.



Introduction

The HERE Maps API for Java ME uses the concept of tiles. The screen is divided into several tiles. When scrolling the map, if the new screen contains tiles that have already been downloaded, these will be used and only the new tiles will be requested via the network. This reduces the amount of generated data traffic and makes the interaction with the user faster. In some platforms, such as Series 40 6th Edition, the default security access level for the underlying network operations by default is set to "Always ask", meaning that many prompts will be generated each time the map needs to be updated. Information on how to change the security access after the application has been installed, and also how to change the default security level via signing the application can be found here: Java Security Domains

Interaction modes for different types of devices

The device requirements for running this example are MIDP2.0 and CLDC 1.1. This covers a very large group of devices from Symbian to Series 40 and from non-touch to Full Touch and Touch and Type. The Map API for Java ME masks away the differences between various devices and provides different interaction modes for each device, without the developer having to worry about the details of the implementation. With respect to zooming, the following interaction modes are supported:

- On non-touch devices, the zoom functionality is provided by the star (*) and hash (#) keys for zooming in and out respectively. No zoom controls are displayed on the screen.
- On full-touch devices, the zoom functionality is provided by the zoom controls displayed on the screen.
- On Touch and Type and full-touch with keyboard devices, the zoom functionality is provided by both the star (*) and hash (#) keys for zooming in and out respectively and the zoom controls displayed on the screen.

With respect to panning (i.e. scrolling or moving the screen around the target location), the following interaction modes are supported:

- On non-touch devices, the navigation keypad is used to move the center of the map up, down, left and right from its current position
- On full-touch and Touch and Type devices, panning is touch driven.
- On full-touch devices with keyboard, such as Nokia E7-00, panning is also supported by the keyboard's arrow keys.

Warning: On S60 5th Edition devices, this application requires that the On Screen Keypad (OSK) is explicitly defined as disabled. In order to disable OSK and use the entire screen for displaying the Map, the following line should be added to the application's file descriptor (jad file):
Nokia-MIDlet-On-Screen-Keypad: no

The search screen

The search screen consists of a `Form`, that contains the `TextField` where the user can type the requested address or point of interest, a hidden info message item that is only used for the display of any potential errors and two `Commands`, an exit and a search `Command`.

(E) Copyright Nokia 2013. All rights reserved.

```
   //Main Search Screen
   display=Display.getDisplay(this);
   mainform=new Form("Address or POI");
   mainform.append(text);
   mainform.append(infomessage);
   mainform.addCommand(searchCmd);
   mainform.addCommand(exitCmd);
   mainform.setCommandListener(this);
   display.setCurrent(mainform);
```

The search `Command` triggers the connection to the Nokia Maps servers. An additional control has been added, so that no operation takes place if the user hasn't typed any text, after selecting the Search option:

```
   if(c==searchCmd)
   {
    String search=text.getString();
    infomessage.setText("");
    if(!search.equals(""))
    {
     showOnMap(search);
    }
   }
```

The map screen

It is quite easy to create a scrollable, dynamic map with zoom functionality with Nokia Maps API for Java ME. We can simply instantiate a new `MapCanvas` object and set it as current:

```
    mapcanvas = new MapCanvas(display);
                        display.setCurrent(mapcanvas);
```

The next problem, is to **geolocate** the search string and instantiate the `geo` object, which is the `GeoCoordinate` of the target location. We can also create a standard balloon marker and attach it to that location. Note that a list of results is retrieved from Nokia's servers as an array of `Location` objects. In this example, we use by default the first available result.

```
  //The search Manager for this search
 GeocodeRequest geocodeRequest = searchFactory.createGeocodeRequest();
 //Retrieves the Point of Interest
 geocodeRequest.geocode(poi, null, this); // this refers to a GeocodeRequestListener
```

The search has been made **asynchronously**, therefore the `GeocodeRequestListener` interface must be implemented, and the app will be informed once the results are supplied.

```
 public void onRequestComplete(GeocodeRequest request, Location[] result) {

         //Stores the results in an array of Location objects
         locationresults = result;
         //The first result is taken
         geo = locationresults[0].getDisplayPosition();
         //Centers the Map at the Point of Interest
         map.setCenter(geo);
         //Displays the marker
         marker = getMapFactory().createStandardMarker(geo);
         map.addMapObject(marker);
         //Sets the zoom level
         map.setZoomLevel(14, 0, 0);
         //Displays the Map on the Screen

         addCommand(exitCmd);
         addCommand(backCmd);
         setCommandListener(this);
         display.setCurrent(this);

     }

  public void onRequestError(GeocodeRequest request, Throwable error) {
 ... etc ...
 }
```

If we need on top of that, our `MapDisplay` to be displayed with a specific zoom level and to be centered around a given coordinate point, we can write the following:

```
    //Centers the Map at the Point of Interest
    mapdisplay.setCenter(geo);
    //Sets the zoom level
           mapdisplay.setZoomLevel(14,0,0);
```

Finally, we need to set the `MapCanvas` we have just created as current:

```
           display.setCurrent(this);
```

The MIDlet source

```
 import com.nokia.maps.common.ApplicationContext;
 import javax.microedition.lcdui.Display;
 import javax.microedition.midlet.MIDlet;
 import javax.microedition.midlet.MIDletStateChangeException;

 public class ShowPOIonMapMidlet extends MIDlet {
```


(c) Copyright Nokia 2013. All rights reserved.

```
    protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {}

    protected void pauseApp() {}

    protected void startApp() throws MIDletStateChangeException {

        // Please initialise the appId and token first
        ApplicationContext.getInstance().setAppID("...");
        ApplicationContext.getInstance().setToken("...");

        Display display = Display.getDisplay(this);
        ShowPOIonMap md = new ShowPOIonMap(display, this);
    }
}
```

```
import com.nokia.maps.common.GeoCoordinate;
import com.nokia.maps.common.Location;
import com.nokia.maps.map.MapCanvas;
import com.nokia.maps.map.MapStandardMarker;

import com.nokia.maps.search.GeocodeRequest;
import com.nokia.maps.search.GeocodeRequestListener;
import com.nokia.maps.search.SearchFactory;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;

public class ShowPOIonMap extends MapCanvas implements CommandListener, GeocodeRequestListener {

    MIDlet midlet;
    Form mainform;

    GeoCoordinate geo;
    TextField text = new TextField("Type Address or POI", "", 100, TextField.ANY);
    StringItem infomessage = new StringItem("", "");
    Command exitCmd = new Command("Exit", Command.EXIT, 0);
    Command backCmd = new Command("Back", Command.BACK, 0);
    Command searchCmd = new Command("Search", Command.OK, 1);
    Location[] locationresults;
    MapStandardMarker marker;
    private SearchFactory searchFactory = SearchFactory.getInstance();

    public ShowPOIonMap(Display display, MIDlet midlet) {
        super(display);
        this.midlet = midlet;
        //Main Search Screen

        mainform = new Form("Address or POI");
        mainform.append(text);
        mainform.append(infomessage);
        mainform.addCommand(searchCmd);
        mainform.addCommand(exitCmd);
        mainform.setCommandListener(this);
        display.setCurrent(mainform);
    }

    public void onRequestComplete(GeocodeRequest request, Location[] result) {

        //Stores the results in an array of Location objects
        locationresults = result;
        //The first result is taken
        geo = locationresults[0].getDisplayPosition();
        //Centers the Map at the Point of Interest
        map.setCenter(geo);
        //Displays the marker
        marker = getMapFactory().createStandardMarker(geo);
        map.addMapObject(marker);
        //Sets the zoom level
        map.setZoomLevel(14, 0, 0);
        //Displays the Map on the Screen

        addCommand(exitCmd);
        addCommand(backCmd);
        setCommandListener(this);
        display.setCurrent(this);

    }

    public void onRequestError(GeocodeRequest request, Throwable error) {
        locationresults = null;
        setTitle("");
        Alert a = new Alert("ERROR", error.toString(), null, AlertType.INFO);

        a.setTimeout(Alert.FOREVER);
        display.setCurrent(a, this);

    }

    public void showOnMap(String poi) {
        try {
            //The search Manager for this search
            GeocodeRequest geocodeRequest = searchFactory.createGeocodeRequest();
            //Retrieves the Point of Interest
            geocodeRequest.geocode(poi, null, this);
            setTitle(poi);

        } catch (RuntimeException e) {
            infomessage.setText("Error:" + e.getMessage());
        }
    }

    public void commandAction(Command c, Displayable d) {
        if (c == searchCmd) {
            String search = text.getString();
            infomessage.setText("");
            if (!search.equals("")) {
                showOnMap(search);
            }
        }
        if (c == exitCmd) {
            midlet.notifyDestroyed();
```


(c) Copyright Nokia 2013. All rights reserved.

```
        }
        if (c == backCmd) {
            display.setCurrent(mainform);
        }
    }

    public void onMapUpdateError(String description, Throwable detail, boolean critical) {
    }

    public void onMapContentComplete() {
    }
}
```

Resources

- The example's source code can be downloaded from here: File:NokiaMapsShowPOIonMapSource.zip
- The example's installation binary files can be downloaded from here: File:NokiaMapsShowPOIonMapBinaries.zip

(c) Copyright Nokia 2013. All rights reserved.

 (c) Copyright Nokia 2013. All rights reserved.