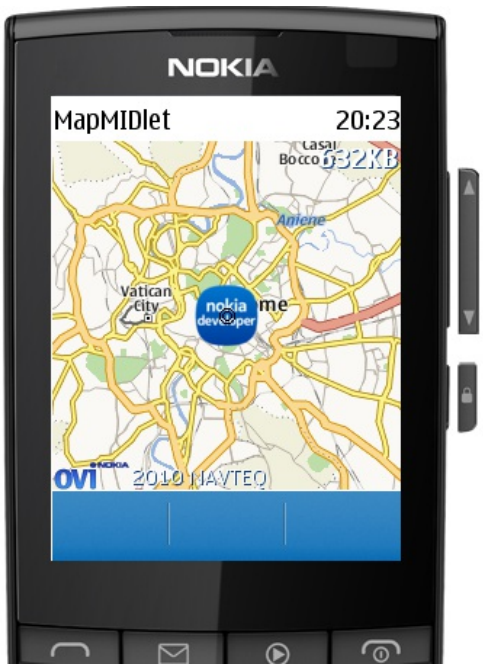
Geocoding an address with Maps API for Java ME

This article explains how to geocode and show an address by using the Maps API for Java ME

Introduction



The [Maps API for Java ME makes it possible to easily embed the Nokia Maps service into Java ME applications. The API includes geocoding functionality, which allows you to retrieve the geographical coordinates associated with a given address or location.

SearchFactory and GeocodeRequestListener

The `SearchFactory` class (available in the `com.nokia.maps.search` package) offers various services, including geocoding, reverse geocoding, and place search. When *geocoding* operations are performed in an **asynchronous** manner, a `GeocodeRequestListener` is notified when they are completed or when an error has occurred.

The GeocodeMIDlet

A base MIDlet to show a `MapCanvas` is as follows:

```
public class GeocodeMIDlet extends MIDlet {

        MapCanvas mapCanvas = null;

        protected void startApp() throws MIDletStateChangeException {

                ApplicationContext ctx = ApplicationContext.getInstance();

                ctx.setAppID("MyAppId");
                ctx.setToken("MyToken");

                Display display = Display.getDisplay(this);
                mapCanvas = new MapCanvas( display  );
                display.setCurrent(mapCanvas);
        }
        protected void destroyApp(boolean unconditional)
                        throws MIDletStateChangeException {

        }
        protected void pauseApp() {
        }
}
```

Implementing the GeocodeRequestListener

Before actually performing a geocode operation, it is necessary to implement the `GeocodeRequestListener` interface. This can be done by implementing two different methods:

- `onRequestError(GeocodeRequest origin, Throwable error)`
- `onRequestComplete(GeocodeRequest origin, Location[] results)`

(C) Copyright Nokia 2013. All rights reserved.

First, the **implements** clause has to be added to the `MapMIDlet` declaration:

```
public class GeocodeMIDlet extends MIDlet implements GeocodeRequestListener{

[...]

}
```

Handling the geocode result

If the geocode operation successfully completes, the `onRequestComplete()` method of the `GeocodeRequestListener` interface is called, with the `GeocodeRequest` instance that performed the geocode operation passed as an argument along with the results. For the purpose of this example, the first returned `Location` is used but in a real app it would be a good practice to let the user manually choose one of the returned locations.

```
public void onRequestComplete(GeocodeRequest request, Location[] result)
{
    if(result.length > 0)
    {
        Location firstLocation = result[0];

        GeoCoordinate position = firstLocation.getDisplayPosition();
    }
}
```

Create and show a MapMarker

Now that the `GeoCoordinate` object associated with the geocoded location is available, it can be used to create a `MapMarker` for the given location. Markers, as all other map objects, must be created by using a `MapFactory` instance, that can be obtained through the `getMapFactory()` method of the `MapCanvas` class.

The `MapMarker` class allows icons to be created with custom images. To properly center the icon over the marker coordinates, an anchor `Point` can be defined to set the offsets (in pixels) used to draw the icon itself.

The `onRequestComplete()` method can be modified as follows:

```
 public void onRequestComplete(GeocodeRequest request, Location[] result) {
    if(result.length > 0)
    {
        Location firstLocation = result[0];

        GeoCoordinate position = firstLocation.getDisplayPosition();

        try {
            MapFactory mapFactory = mapCanvas.getMapFactory();

            MapMarker marker = mapFactory.createMapMarker(position, Image.createImage("/nokiaicon.png"));

            marker.setAnchor(new Point(- 24, - 24));

            mapCanvas.getMapDisplay().addMapObject(marker);
        }
        catch(Exception e)
        {
        }
    }
}
```

As the icon used in this example measures 48x48 pixels, the marker defines offsets of -24 pixels on each side. What is left to do now is to center the map over the geocoded location, and this can be easily performed with the `MapDisplay.setCenter()` method, as shown in the code snippet below:

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

    if(result.length > 0)
    {
        Location firstLocation = result[0];

        GeoCoordinate position = firstLocation.getDisplayPosition();

        mapCanvas.getMapDisplay().setCenter(firstLocation.getDisplayPosition());
    }
}
```

Handling search errors

The `GeocodeRequestListener` interface defines the `onRequestError()` that handles errors raised during a search operation. The `Throwable` object passed as argument can be used to obtain more information about the error that was generated (and to accordingly show an error to the user):

```
public void onRequestError(GeocodeRequest request, Throwable error) {
    Alert alertView = new Alert("Search request error: " + error.getMessage());
    Display.getDisplay(this).setCurrent(alertView);
}
```

Using the geocode feature

Now that a `GeocodeRequestListener` has been fully implemented, the `GeocodeRequest` can be used to perform a geocode operation. To do this, three steps are necessary:

(C) Copyright Nokia 2013. All rights reserved.

- Obtain the `SearchFactory` instance through its `getInstance()` method.
- Create an instance of a `GeocodeRequest`'
- Geocode an address through the `geocode()` method, passing in the `GeocodeRequestListener` as the final parameter.

The `startApp()` method of the `MapMIDlet` can be modified as follows:

```
protected void startApp() throws MIDletStateChangeException {

        ApplicationContext ctx = ApplicationContext.getInstance();

        ctx.setAppID("MyAppId");
        ctx.setToken("MyToken");

        Display display = Display.getDisplay(this);
        mapCanvas = new MapCanvas( display  );
        display.setCurrent(mapCanvas);

        SearchFactory searchFactory = SearchFactory.getInstance();
        GeocodeRequest geocodeRequest = searchFactory.createGeocodeRequest();
        geocodeRequest.geocode("Rome, Italy", null, this);
}
```

Resources

A sample MIDlet that shows the above code in action is available here: File:GeocodeMIDlet.zip

Full source code of the `GeocodeMIDlet` is available here: File:GeocodeMIDlet.java.zip

Summary

The Maps API for Java ME offers rich functionality that allows all the main Nokia Maps features to be integrated into a Java ME application, with just a few lines of code.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.