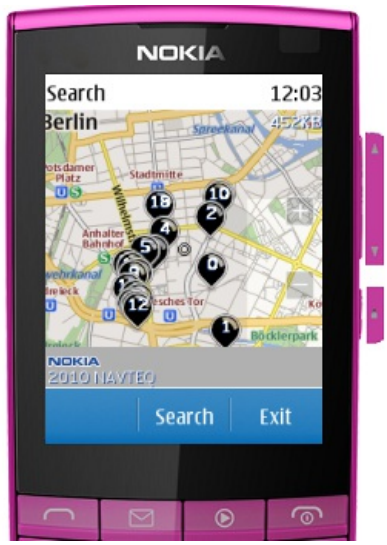
Archived:Search with Maps API for Java ME

This article explains how to use search services provided with Maps API for Java ME API.

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

The article is believed to be still valid for the original topic scope.

Introduction



The Maps API for Java ME makes it easy to embed the Nokia Maps service into Java ME applications. This API also includes search service, with which you can search places.

SearchFactory and PlaceSearchRequestListener

Similarly to Geo services, the search service is handled with the `SearchFactory` class, and as with other services there are methods for **synchronous** as well as for **asynchronous** use of the search service. With **asynchronous** search, results are given back with methods defined with `PlaceSearchRequestListener` interface.

And if the we want the search to be handled **asynchronously**, then the required `PlaceSearchRequestListener` interface needs to be implemented in the search class (or the class taking care of the results), this interface defines the following 2 methods:

- `onRequestComplete(PlaceSearchRequest request, Place[] result)`
- `onRequestError(PlaceSearchRequest request, java.lang.Throwable error)`

And to include this interface in your midlet implementation you need to use the `implements` clause. Basic code that handles this is shown below:

```
public class MapSearchClass extends MIDlet
implements PlaceSearchRequestListener, …
{
public void onRequestComplete(PlaceSearchRequest request, Place[] result) {
    ...
    }
public void onRequestError(PlaceSearchRequest request, java.lang.Throwable error) {
    ...
    }
}
```


(C) Copyright Nokia 2014. All rights reserved.

Starting the query

Before the service can be used you need to get the `SearchFactory` instance, which is implements the service; this is handled by simply calling the `getInstance()` method from `SearchFactory`:

```
sm = SearchFactory.getInstance();
```

Then you need to construct the actual search request by calling the `createPlaceSearchRequest()` method from the search factory. Then the actual search can be started by calling methods implemented in the `PlaceSearchRequest()`. Note that since we implemented the `PlaceSearchRequestListener` in the midlet, we can give this as a third argument for asynchronous search requests.

```
GeoCoordinate topLeft = mapCanvas.getMapDisplay().pixelToGeo(new Point(0,0));
GeoCoordinate btRight = mapCanvas.getMapDisplay().pixelToGeo(new
Point(mapCanvas.getMapDisplay().getWidth(),mapCanvas.getMapDisplay().getHeight()));
GeoBoundingBox bboxx  = new GeoBoundingBox(topLeft,btRight);

PlaceSearchRequest Pls = sm.createPlaceSearchRequest();
Pls.search(geostr, bboxx,this);
```

Handling the result

Successful completion of the asynchronous search will be reported by calling the `onRequestComplete()` method implemented for the `PlaceSearchRequestListener` interface. The `onRequestComplete()` method will be called with a `PlaceSearchRequest` as the first argument, as well as an array of `Places` as the second argument, the resultant `Places` can then be retrieved directly from the array.

```
public void onRequestComplete(PlaceSearchRequest request, Place[] result){
 for(int i = 0; i < result.length; i++){
        Place firstPlace = result[i];
                Location[] locArr = firstPlace.getLocations();
 ...
 }
}
```

The query could also be finished with error, and to handle those you must also implement the `onRequestError()` method, in this example this method simply informs the user of the possible problem.

```
public void onRequestError(PlaceSearchRequest request, java.lang.Throwable error) {
 Alert alertView = new Alert("Search request error: " + error.getMessage());
 display.setCurrent(alertView);
}
```

Resources

Full source code of the Search and Places search is available at: File:Java MapSearch.zip

Summary

The Java ME Location API offers rich functionalities which make it possible to integrate all the main Ovi Maps features in a Java ME application, with just a few lines of code.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.