Using MGMaps Library with J2MEPolish

This tutorial describes how build **MGMaps Library** applications using **Eclipse**, **Ant**, **Sun Wireless ToolKit** and **J2MEPolish.**

Preconditions:

1) **J2MEPolish** installed (it is recommended that you install **J2MEPolish** into directory path without spaces, **WTK** preverifier does not like spaces).
2) **WTK** installed
3) **Proguard** obfuscator installed
4) **Eclipse** with **Ant** plug-in installed

You can also use different IDE and so that the Ant build script would remain the same. Also since this sample build script uses J2MEPolish build target in build script which is highly customizable, you can also use different obfuscator, preverifier and etc – about that you'll find more information from J2MEPolish documentation: http://www.j2mepolish.org/cms/leftsection/documentation/building.html .

Creating and building the project:

1. Create a new project in **Eclipse** and some midlet source files that use **MGMaps Library** and **J2MEPolish** functionality, or import **J2MEPolishSample** project into **Eclipse**.
2. Create an **Ant** build script for your project.
   You can use **J2MEPolishSample** *build.xml* sample build script and modify it according to your own project (if building some other project than **J2MEPolishSample** project).

   If you are building **J2MEPolishSample** project, then just modify copy *build.properties_template* file to *build.properties* and modify *${polish.home}* and *${maps.lib}* values to point to correct locations.

   If you are building a new project then in addition to *build.properties* file you also have to modify following parts in sample build script:

```
<project
      Modify build project name:
      name="J2MEPolishSample"
      [...]
      Modify midlet name:
      <property name="midlet.name" value="Polish Map" />
      [...]
      Modify target devices (from J2MEPolish database):
      <property name="device" value="Sony-Ericsson/JavaPlatform6"/>
```

```
<property name="devices" value="Sony-Ericsson/JavaPlatform6"/>
      [...]
Modify deploy-url:
<target name="setdeploy"
      description="Call this target first to set the OTA download-URL,
      e.g. ant setdeploy j2mepolish">

            <property name="deploy-url"
            value="http://www.company.com/download/" />
</target>
[...]

<target name="j2mepolish" depends="init"
      description="This is the controller for the J2ME build process.">
      <j2mepolish>
            Modify midlet parameters:
            <info

                  name="Polish Map"
                  version="0.8.0"
                  description="J2MEPolish Sample Application"
                  vendorName="Nutiteq"
                  icon="/midletIcon.PNG"
                  jarName="J2MEPolishMap_${polish.vendor}_${polish.name
                  }_${polish.locale}.jar"
                  jarUrl="${deploy-url}${polish.jarName}"

            />
      [...]

      </j2mepolish>
</target>

[...]

Modify preverifier classpath if needed (if you are using some
additional external packages). If some of those paths contain spaces,
preverify may fail in which case move the file into new path without
spaces.
<target name="preverify">
      <exec executable="${wtk.home}/bin/preverify">
            <arg line="-classpath
            ${wtk.home}/lib/cldcapi11.jar;${wtk.home}/lib/midpapi20.jar
            ;${wtk.home}/lib/jsr179.jar;${wtk.home}/lib/jsr75.jar;${wtk
            .home}/lib/jsr184.jar;${wtk.home}/lib/mmapi.jar;${wtk.home}
            /lib/jsr082.jar;${lib.dir}/j2me-ext-motorola-
            1.0.jar;${lib.dir}/j2me-ext-siemens-
            1.0.jar;${polish.home}/lib/enough-j2mepolish-
            client.jar;${lib.dir}/nokia-ui.jar"/>
            <arg line="-d ${polish.preverify.target}" />
            <arg line="-target CLDC1.1" />
            <arg line="${polish.preverify.source}" />
      </exec>
</target>
[...]
```

```
</project>
```

In order to build the project you must run at least „*j2mepolish*“ target. In **J2MEPolishSample** project there are different targets created to build debug version and release version of the project:

```
<target name="buildJ2MEPolishSample_localTest" depends="clean, test,
j2mepolish"/>
<target name="buildJ2MEPolishSample_releaseVersion" depends="clean,
obfusc, j2mepolish"/>
```

„*Clean*“ target has to be run each time the target device has been changed, in debug version the target „*test*“ is also run, which sets the „*test*“ property to true in which case different target device can be used when building debug version.
When building release version, then the property „*obfuscate*“ is set to true and code is run through **Proguard** obfuscator

NOTE: If you don’t use the sample build script, but create an new one from scratch, then don’t forget to add

```
binaryLibraries="${maps.lib}"
```

reference to MGMaps Library JAR-file in J2MEPolish <build> target.

Generic recommendations:

If you want your application to use J2MEPolish style commands (and J2MEPolish built softkey values according to selected device) instead of J2ME native commands, you should use MapItem instead of MapComponent in your applications, because MapItem can be placed on a Form, which is high-level UI element and therefore customized by J2MEPolish, while MapComponent is drawn directly on Canvas, which is low-level UI element which will not be customized by J2MEPolish.

Using **Canvas** and **MapComponent** (J2ME native look commands/menu):		Using **MapItem** and **Form** (J2MEPolish look commands/ menu):