Porting MIDlets from Symbian to Nokia Asha

This article provides instructions on how to port existing MIDlets written for Symbian to Nokia Asha software platform.

19 May 2013

Memory and installation package constraints

One of the biggest challenges when porting existing Symbian MIDlets to Nokia Asha software platform, is to ensure that the application does not exceed the allowed limit for memory consumption. The Symbian platform does not cap the Java heap and it doesn't set a size limitation for the installation package (the .jar file). Applications running on Nokia Asha software platform however, do not have more than 3 MB of available Java heap at their disposal. The latest Series 40 full touch devices, set a similar limitation of 2 MB to the Java heap. Although there is no limit for the installation package on Asha software platform, it is recommended to use a .jar file that doesn't exceed 5 MB in size. If the same installation package targets Series 40 devices, it should not exceed 2 MB. For more information about the Java heap and installation package limits per device, see the Device Specifications page.

The most common reason why the Java heap can be easily consumed is the presence of Image instances. Every time an Image object is instantiated on Nokia Asha software platform, it is represented as a bitmap in the Java heap. For more information on how Image instances consume the available memory see the Memory Usage section in Java Developer's Library. In order to avoid Out Of Memory errors, when opening images from the local drive, use the Image Scaling API, to downscale the image. This can be proven particularly helpful, when opening high resolution images.

Orientation

While high level UI components scale automatically on Symbian, and low level Canvas instances are being repainted, by automatically calling the `paint()` method, when a change in the orientation of the device occurs, the same operation is not implemented automatically by the Application Management Software on the Asha Software platform. The developer needs to use the `OrientationListener`, as well as the `Nokia-MIDlet-App-Orientation` JAD attribute in order to achieve the same functionality. First you need to add the `OrientationListener` to your MIDlet, and then you have to implement its `displayOrientationChanged` method as follows:

```
import com.nokia.mid.ui.orientation.Orientation;
import com.nokia.mid.ui.orientation.OrientationListener;
…
  protected void startApp()
    throws MIDletStateChangeException  {
    Orientation.addOrientationListener(this);
  }
  ...
  public class YourMIDlet extends MIDlet implements OrientationListener{
    /** Orientation listener callback */
    public void displayOrientationChanged( int newDisplayOrientation ){

    /** Check display orientation */
    switch( newDisplayOrientation ){
      case Orientation.ORIENTATION_PORTRAIT:
      case Orientation.ORIENTATION_PORTRAIT_180:
        /** Change MIDlet UI orientation to portrait */
        Orientation.setAppOrientation(Orientation.ORIENTATION_PORTRAIT);
        break;

      case Orientation.ORIENTATION_LANDSCAPE:
      case Orientation.ORIENTATION_LANDSCAPE_180:
        /** Change MIDlet UI orientation to landscape */
        Orientation.setAppOrientation(Orientation.ORIENTATION_LANDSCAPE);
        break;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
  }
}
```

Then in the application descriptor file of your project you need to add the following attribute:

```
Nokia-MIDlet-App-Orientation: manual
```

Hiding the Virtual Keyboard for Canvas based applications

Similarly to Symbian devices, Nokia Asha software platform displays by default a Virtual Keyboard for Canvas based applications. The Virtual Keyboard is present in the Option's Menu. The behavior of the On-Screen Keypad (OSK) varies per platform version with the latest platforms (starting from Symbian^3) being able to automatically hide it, if touch interaction is implemented via the methods `pointerPressed()`, `pointerReleased()` and `pointerDragged()`. In order to enforce the hiding of the OSK, irrespective of platform version or the presence of touch interaction, the following attribute is used on Symbian devices:

Nokia-MIDlet-On-Screen-Keypad: no

This attribute is not supported on Series 40 devices. In order to enforce the hiding of the Virtual Keyboard on Asha software platform, you need to use this snippet:

```
String keyboardType = System.getProperty("com.nokia.keyboard.type");
if (keyboardType.equals("OnekeyBack") || keyboardType.equals("None")) {
    // Series 40 full touch or Nokia Asha software platform detected
    // Your code here
    com.nokia.mid.ui.VirtualKeyboard.hideOpenKeypadCommand(true);
}
```

Adding a CategoryBar in ported Symbian MIDlets with parallel views

If you are instantiating several `Form` objects that represent different parallel views of your application, that are alive throughout the life cycle of your MIDlet, you can use the `CateogyBar` instead on Nokia Asha software platform, to switch among them. Let's assume you would like to port to Asha software platform a Symbian MIDlet, that contains these four views:

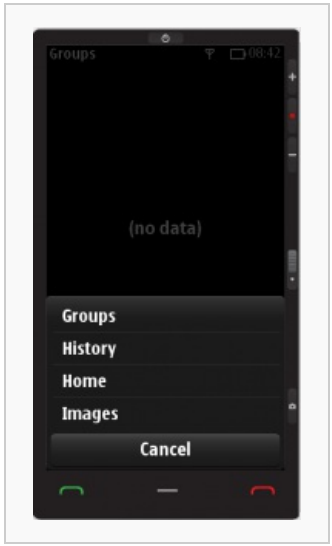
- "Groups"
- "History"
- "Home"
- "Images"

```
public class CategoryBarPortedSymbian
    extends MIDlet
    implements CommandListener {

    Command Groups = new Command("Groups", Command.OK, 1);
    Command History = new Command("History", Command.OK, 2);
    Command Home = new Command("Home", Command.OK, 3);
    Command Images = new Command("Images", Command.OK, 4);

    Form groupForm;
    Form historyForm;
    Form homeForm;
    Form imagesForm;
    Display display;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    protected void destroyApp(boolean unconditional)
            throws MIDletStateChangeException {
        // TODO Auto-generated method stub
    }

    protected void pauseApp() {
        // TODO Auto-generated method stub
    }

    protected void startApp() throws MIDletStateChangeException {
        Form groupForm = new Form("Groups");  // default view
        Form historyForm = new Form("History");
        Form homeForm = new Form("Home");
        Form imagesForm = new Form("Images");

        groupForm.addCommand(Groups);
        groupForm.addCommand(History);
        groupForm.addCommand(Home);
        groupForm.addCommand(Images);


        display = Display.getDisplay(this);
        display.setCurrent(groupForm);
    }

    public void commandAction(Command c, Displayable d) {
        //handle here the change of views
    }
}
```

The Group (default) view



Switching among views is performed with Commands

In the snippet above, you would have to use the method `display.setCurrent()` withing the `commandAction()` and give as argument the Form that represents the view that the user selected by pressing one of the respective Commands. This is how you can implement a similar 4-element CategoryBar MIDlet on Asha software platform:

```
public class CategoryBarSimple
 extends MIDlet
 implements ElementListener {
```

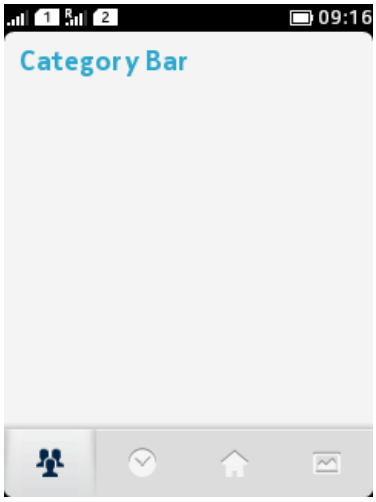

(C) Copyright Nokia 2014. All rights reserved.

```
  Form mainForm;
  CategoryBar bar;
  Image[] unselectedIcons;
  Image[] selectedIcons;
  Command okCmd = new Command("OK", Command.OK, 1);
  Command cancelCmd = new Command("CANCEL", Command.CANCEL, 2);

  protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
   // TODO Auto-generated method stub
  }

  protected void pauseApp() {
   // TODO Auto-generated method stub
  }

  protected void startApp() throws MIDletStateChangeException {
   mainForm = new Form("Category Bar");
   Display.getDisplay(this).setCurrent(mainForm);
   try {

    unselectedIcons = new Image[6];

    InputStream ios1 = this.getClass().getResourceAsStream("qgn_catbar_groups.png");
    InputStream ios2 = this.getClass().getResourceAsStream("qgn_catbar_history.png");
    InputStream ios3 = this.getClass().getResourceAsStream("qgn_catbar_home.png");
    InputStream ios4 = this.getClass().getResourceAsStream("qgn_catbar_images.png");

    unselectedIcons[0] = Image.createImage(ios1);
    unselectedIcons[1] = Image.createImage(ios2);
    unselectedIcons[2] = Image.createImage(ios3);
    unselectedIcons[3] = Image.createImage(ios4);

   } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   }

   String[] labels = { "Groups", "History", "Home", "Images"};
   bar = new CategoryBar(unselectedIcons, null, labels);

   bar.setElementListener(this);
   bar.setVisibility(true);

  }

  public void notifyElementSelected(CategoryBar bar, int selectedIndex) {
   //To be implemented
  }
 }
```

Notice, that CategoryBar elements are always visible at the Toolbar section of the screen. You can add custom images for branding each element and you can customize the highlight color of the CategoryBar.

This is how the MIDlet looks on Asha software platform, provided that we have added the appropriate .png files into the resource folder:

(C) Copyright Nokia 2014. All rights reserved.



Persistent Storage and retrieval of Record Store entries on Asha Software Platform

There is a significant difference between Symbian and the Asha software platform when it comes to retrieving persistent storage entries. When a MIDlet stores data to the Persistent Storage (RecordStore), there is an index or RecordStoreId associated with the entry.On Symbian devices, retrieving the entries of a RecordStore with the `nextRecordId()` method is performed according to the entry order (i.e. last one added retrieved last). On both Series 40 and Asha software platforms, however, the retrieval is performed with the reversed order in which the entries were added (i.e. last one added retrieved first). If the retrieval order matters, we need to always ensure that the RecordStore is traversed in the same order in both Nokia Asha platform and Symbian. This can be done as follows:

```
RecordStore rs = RecordStore.openRecordStore("test",
true);
RecordEnumeration en;
en = rs.enumerateRecords(null, null, false);
en.reset();
int index = en.nextRecordId();
//Series 40
if(index > 1) {
  for(int i = 1; i <=index; i++) {
    rs.getRecord(i);
  }
}
//Symbian
else {
  while (en.hasNextElement()) {
    int RecId = en.nextRecordId();
    rs.getRecord(RecId);
  }
}
```

Porting Location aware MIDlets to Asha software platform

The Nokia Asha platform, like Symbian, supports JSR-179 (the Location API). On Symbian devices, the retrieval of the Location Provider is done based on defining Criteria that among others include a desired horizontal and vertical accuracy or the request to also retrieve speed, besides coordinates. Criteria-based location retrieval is also supported on Asha software platform provided that an external Bluetooth GPS module is used. For applications that do not require precise and on the fly location updates, the current (approximate) location is performed by utilizing Cell ID information in Online or WLAN modes. Cell ID location retrieval is only supported by Asha software platform and Series 40 devices. When porting from Symbian, you need to modify the way you retrieve the location provider.

(C) Copyright Nokia 2014. All rights reserved.

This is a sample snippet on how the `QualifiedCoordinates` can be retrieved on Symbian, using the Assistant GPS (A-GPS) location method, where the device will utilize the data connection (Internet) to contact an assistance server, by allowing costs to occur:

```
public class LocationMIDlet
  extends MIDlet
  implements LocationListener {
  …
  Criteria criteria = new Criteria();
  criteria.setSpeedAndCourseRequired(false)
  criteria.setCostAllowed(true)
  criteria.setPreferredPowerConsumption(Criteria.POWER_USAGE_LOW)
  LocationProvider lp = LocationProvider.getInstance(criteria);
  lp.setLocationListener(this, -1, -1, -1);
  …
  public void locationUpdated(LocationProvider provider, Location location) {
    if(location.isValid()) {
      QualifiedCoordinates c = loc.getQualifiedCoordinates();
    }
  }
}
```

In order to get the same pair of `QualifiedCoordinates` using Cell ID based location retrieval on Nokia Asha platform, you need to use this snippet:

```
//Specify the retrieval method to Online/Cell-ID
int[] methods = {(Location.MTA_ASSISTED | Location.MTE_CELLID | Location.MTE_SHORTRANGE
| Location.MTY_NETWORKBASED)};
// Retrieve the location provider
LocationProvider provider = LocationUtil.getLocationProvider(methods, null);
Location location = provider.getLocation(50000);
QualifiedCoordinates coordinates = location.getQualifiedCoordinates();
```


(C) Copyright Nokia 2014. All rights reserved.