Getting Started with Nokia Asha SDK Templates

This article explains how to get started with code templates provided with Nokia Asha SDK.

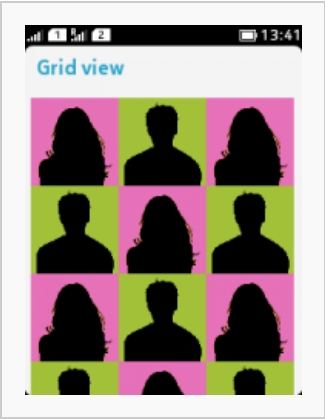
09 Jun 2013

Introduction

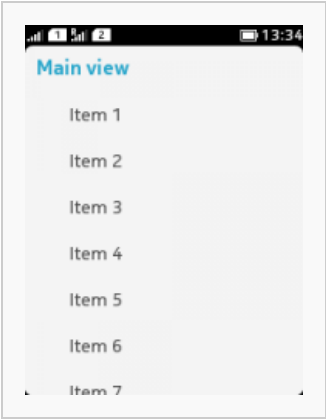
The Nokia Asha SDK 1.0 provides a number of code templates to help developers quickly get started with Java ME app development. The templates from the previous SDK (Nokia SDK 2.0 for Java) are still available and have been updated to support the new Nokia Asha software platform 1.0. In addition four new user interface (UI) oriented templates have been added, along with an advanced UI example: "Composed UI".

The new UI templates are very handy when dealing with list based data such as galleries, phone books etc. and you can get a running start in your app development using the templates; just throw away what you don't need and implement more to add value.

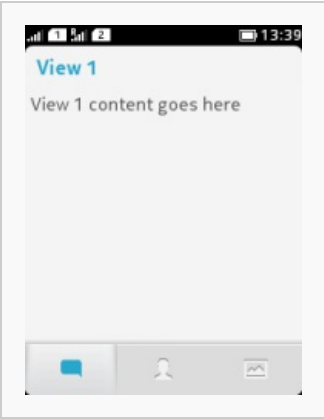
The article briefly introduces the new UI templates in the SDK and demonstrates how to use a template to create a gallery app for Nokia Asha software platform. The only prerequisite for this article is basic understanding of Java ME development.



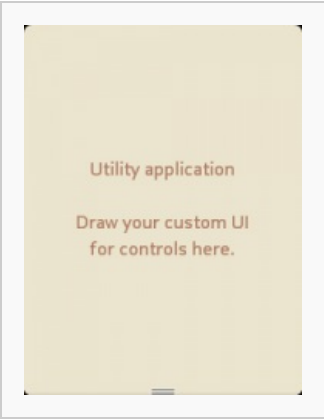
Grid UI



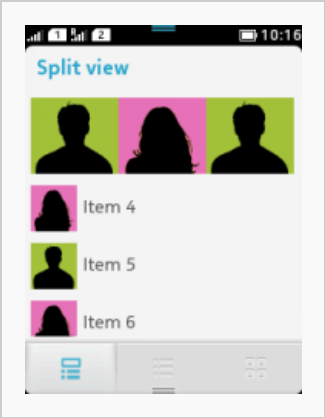
List UI



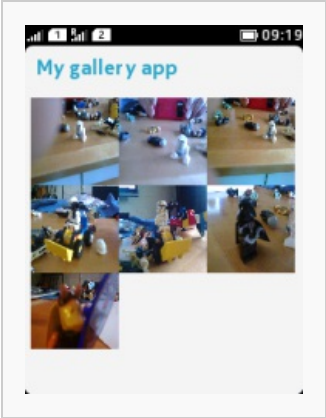
Tabbed UI



Utility template



Composed UI



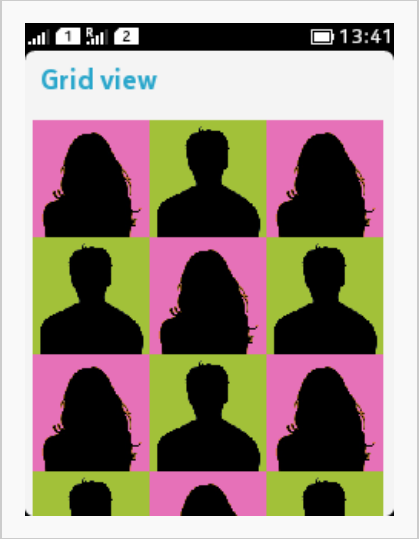
Gallery app

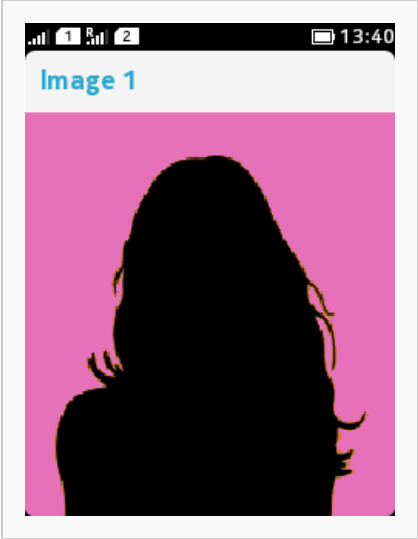
New UI templates in Nokia Asha SDK 1.0

Grid UI

Grid UI shows your data items in a grid and provide drill-down navigation. Tap the item in the grid to see the details. Use the back button to navigate back. Simple. We will use this template to implement our simple gallery app.

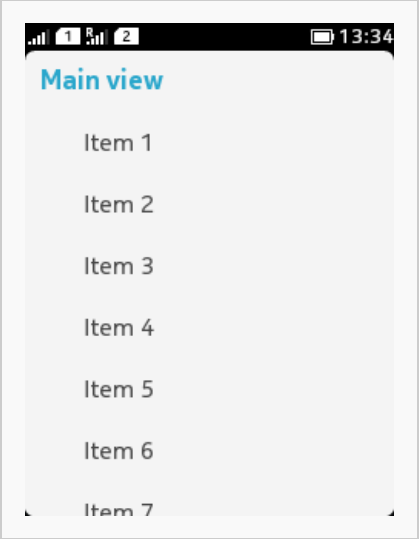
(C) Copyright Nokia 2014. All rights reserved.

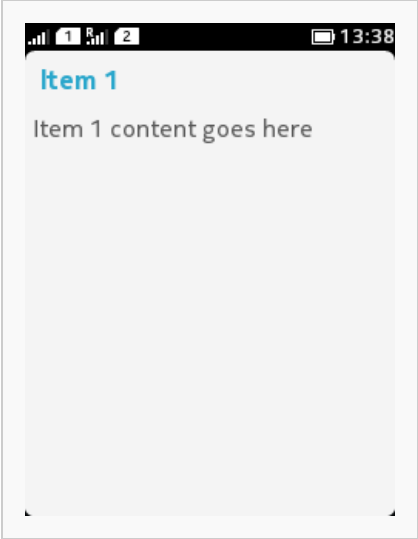




List UI

List UI utilises the LCDUI List component to show the data.

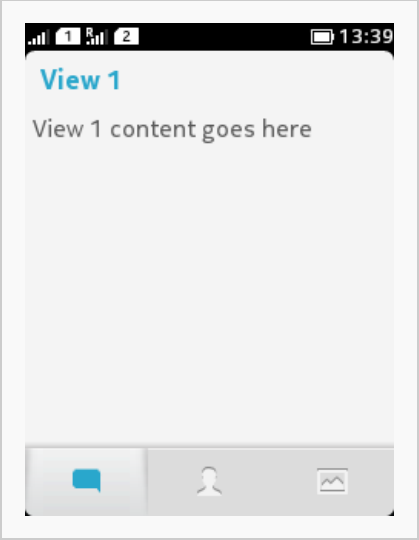


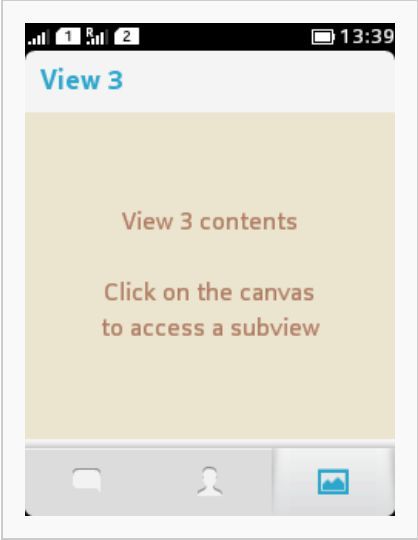


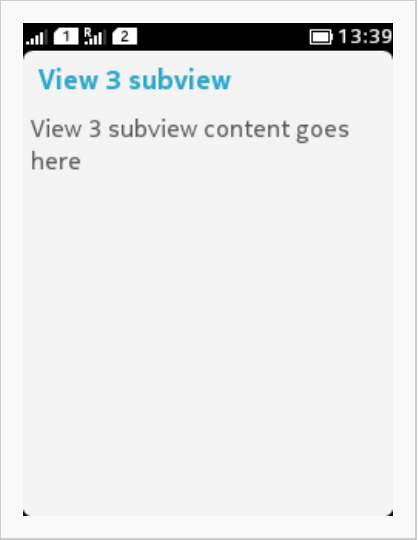
Again, tapping an item opens the details view and pressing the back button brings you back. The last item in the list associated with Canvas based details view while the others are Form based.

Tabbed UI

Tabbed UI shows how to use the CategoryBar of Nokia UI API for providing tab based navigation. The last view (top-right) is `Canvas` based and has a drill-down navigation to a sub view.



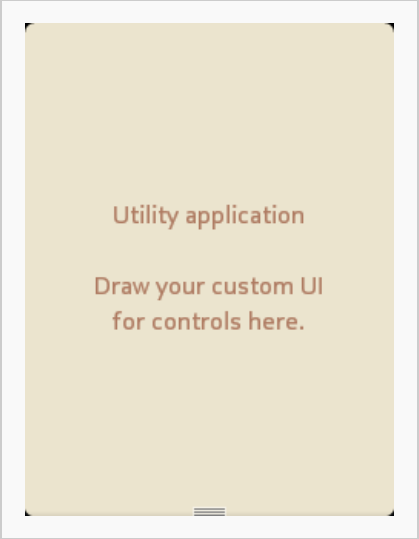


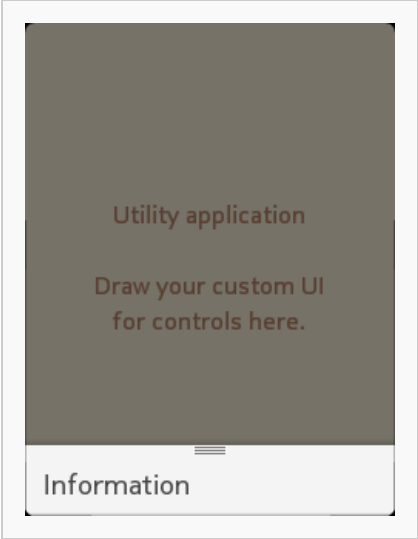


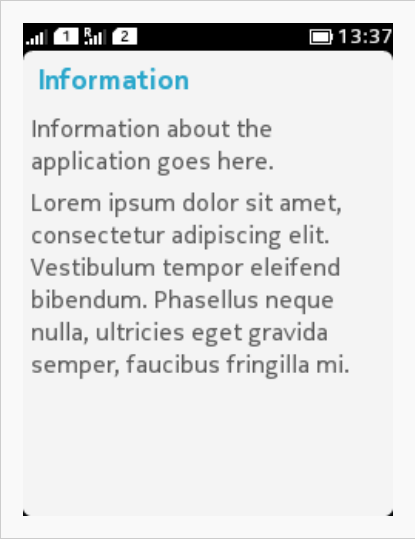
(C) Copyright Nokia 2014. All rights reserved.

Utility

Utility template is convenient for small utility apps with a single main view. The main view is `Canvas` based. The help or about page, where you usually have usage instructions and vendor information, is `Form` based.





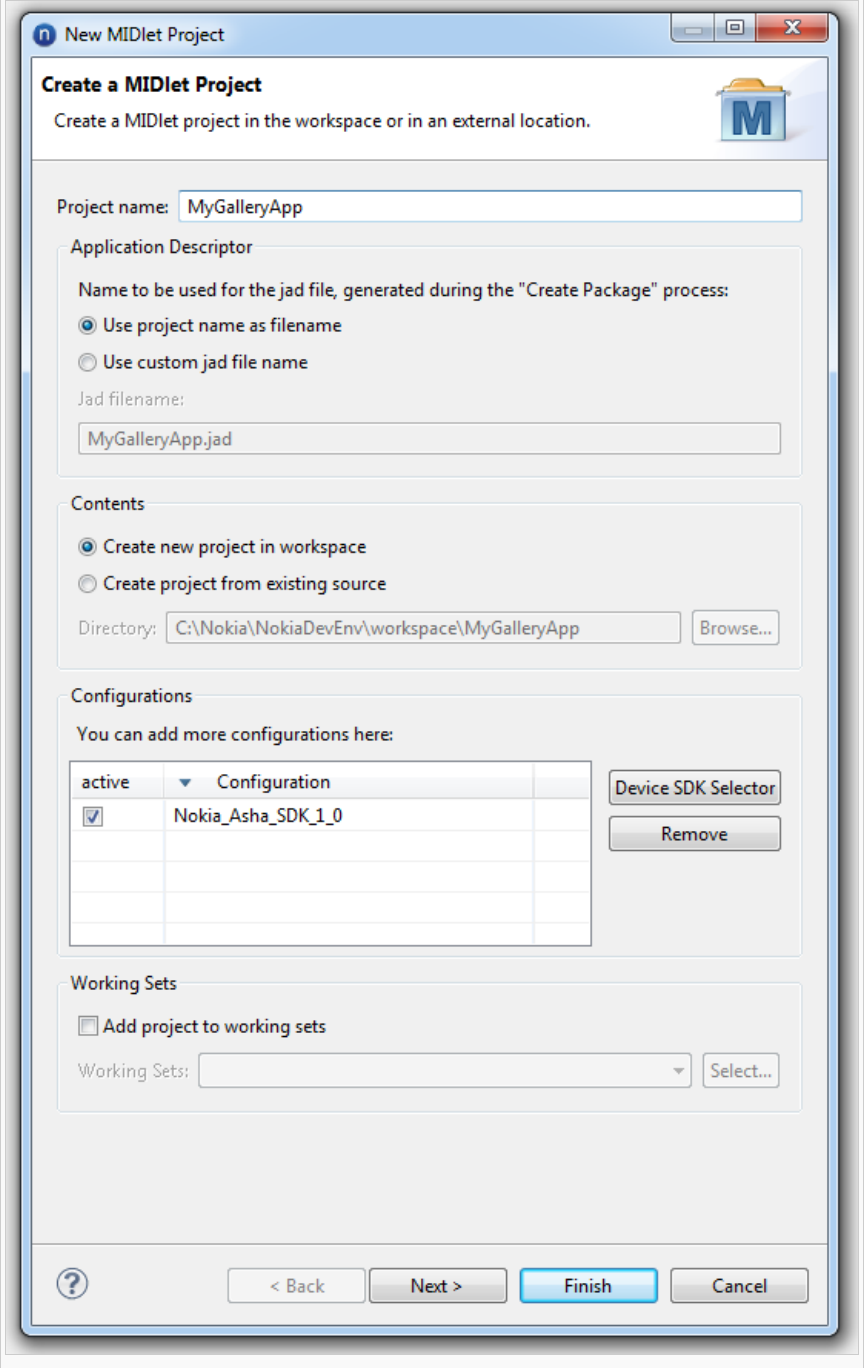


Creating a gallery app using Grid UI template

Creating a project using a template and running it in the emulator

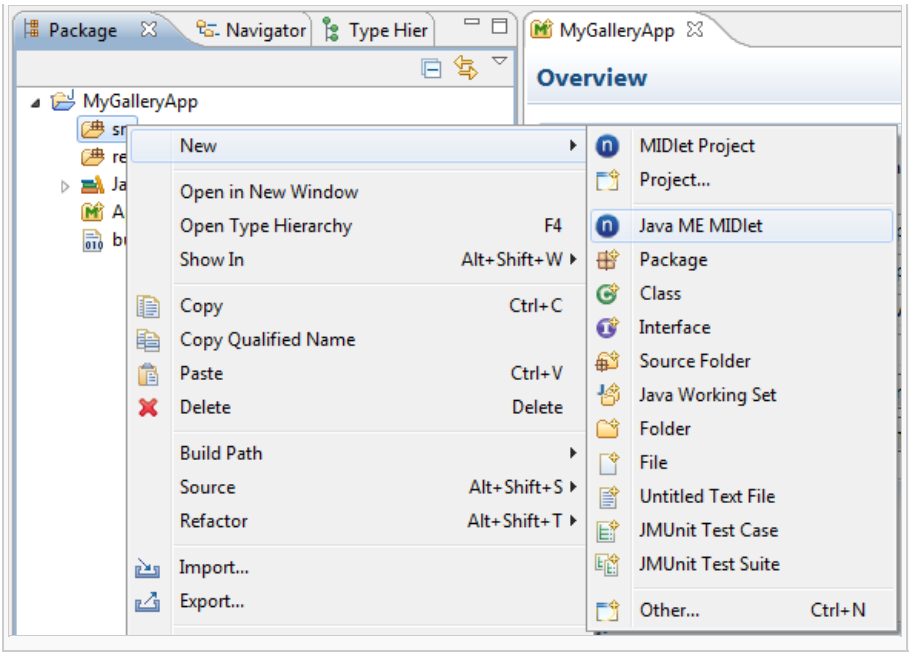
1. **Step 1. Create a new MIDlet project.** Input the name and click Finish.

(C) Copyright Nokia 2014. All rights reserved.

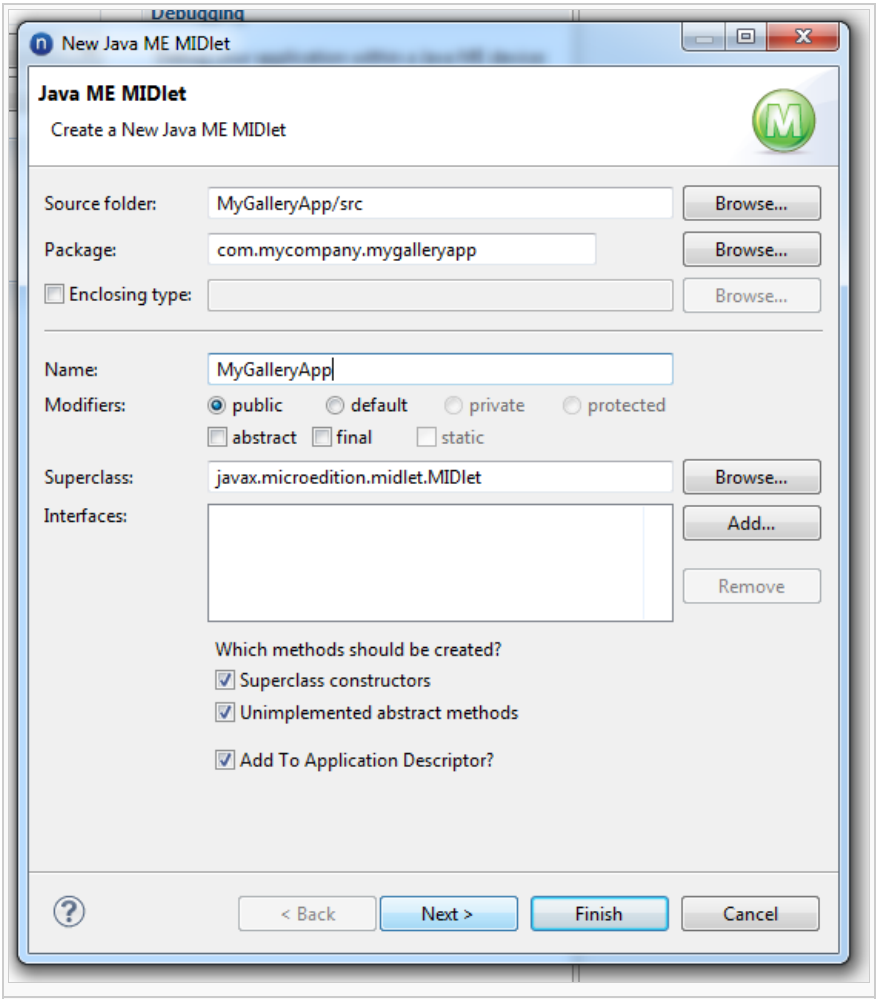


2. **Step 2. Create a new MIDlet.** The created project does not have any source files yet. Open the wizard for creating a new MIDlet.

(C) Copyright Nokia 2014. All rights reserved.

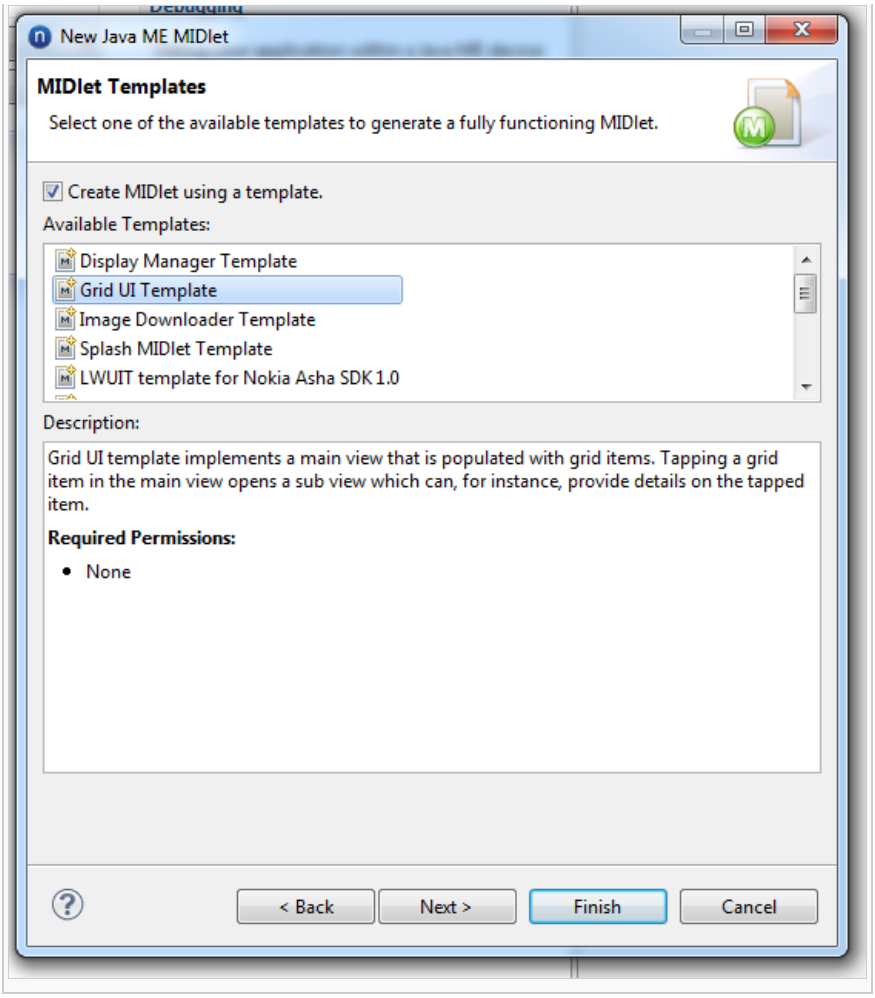


3. **Step 3. Define the package and the name for the new MIDlet.** It is recommended to define a package i.e. using the default package (root) is discouraged. A good name for the MIDlet is usually the name of your app or, for instance, "Main" since it will define the main class of the MIDlet application. After you have named the package and the MIDlet, click Next instead of Finish.

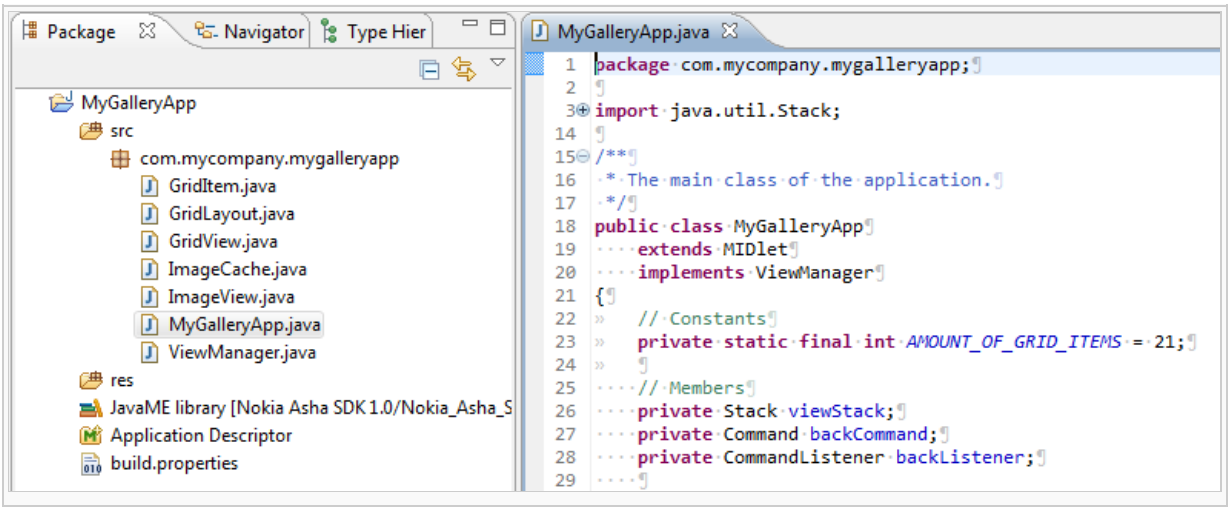


4. **Step 4. Select a template.** Select "Create MIDlet using a template." and select the template. In this case we will be using Grid UI. Finally, click Finish.

(C) Copyright Nokia 2014. All rights reserved.

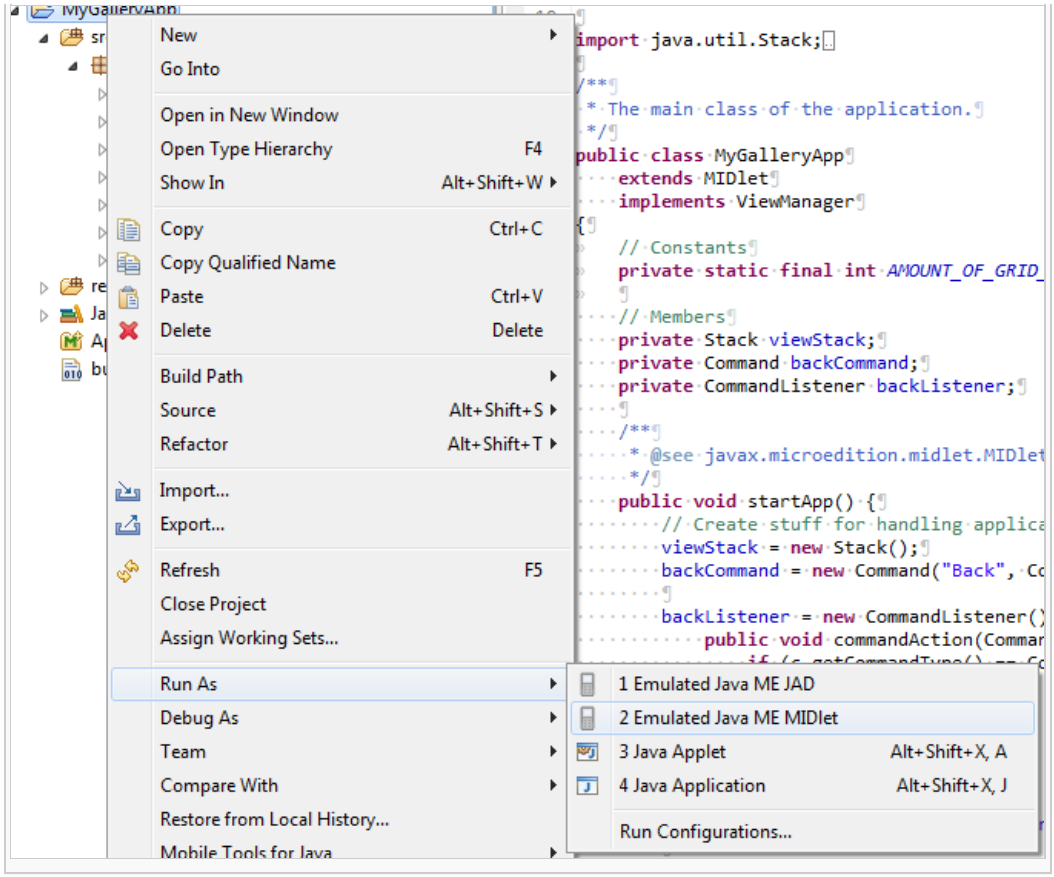


5. **Step 5. Admire your newly created project.** Congratulations, you now have a working MIDlet project.

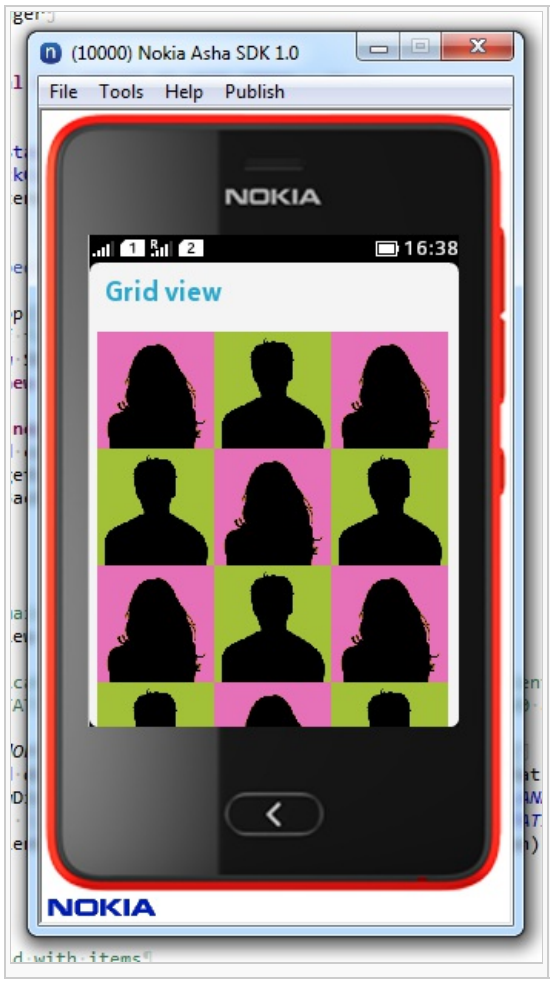


6. **Step 6. Run the MIDlet in the emulator.** To make sure we have everything in place, let's run the MIDlet. Just for kicks.

(C) Copyright Nokia 2014. All rights reserved.



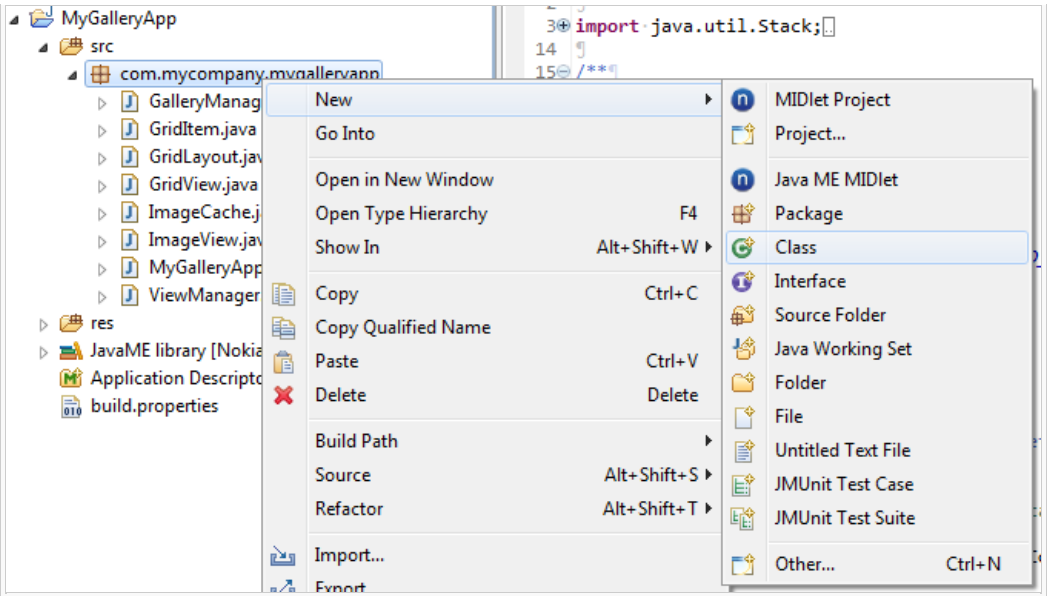
7. **Step 7. Behold the new MIDlet running in the emulator.** And how about that! It's running perfectly.



Implementing the gallery

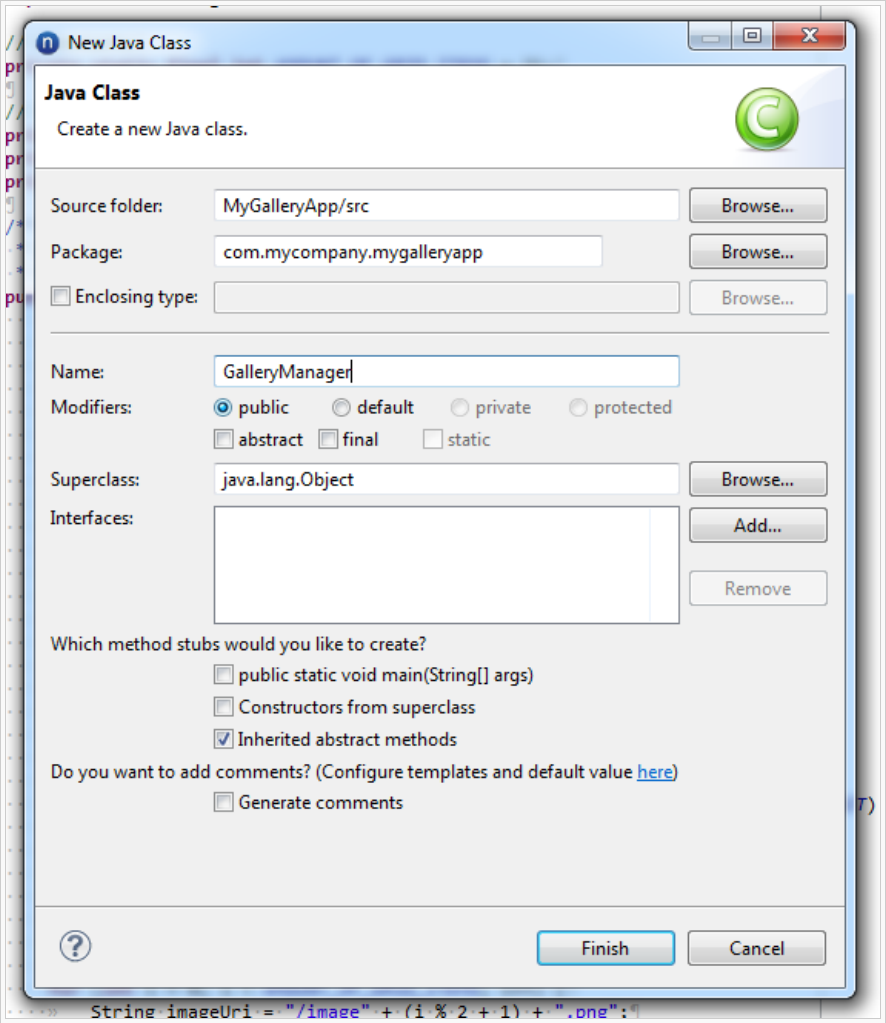
First we need to create a class which will manage retrieving the image file URIs from the photos folder.

(C) Copyright Nokia 2014. All rights reserved.



Create a class

Let's simply name it `GalleryManager`:



Name it GalleryManager class

Then we write some code. Let's keep it simple. Just fetch the images with `.jpeg` prefix from the memory card. These will include the images captured by camera. Below is the full content of the `GalleryManager.java`. You may notice that we have a nasty dependence implemented here: Our new `GalleryManager` creates the grid items and calls the method of the `GridView` directly. Normally we would have a listener interface for the gallery manager, but here a shortcut is taken for the sake of simplicity.

```
package com.mycompany.mygalleryapp;
```


(C) Copyright Nokia 2014. All rights reserved.

```
import java.io.IOException;
import java.util.Enumeration;
import java.util.Vector;

import javax.microedition.io.Connector;
import javax.microedition.io.file.FileConnection;

public class GalleryManager {
    private static final String PHOTOS_FOLDER_URI =
System.getProperty("fileconn.dir.photos");
    private static final String JPEG_IMAGE_FILE_FILTER = "*.jpeg";
    private Vector imageFileURIs = new Vector();

    public synchronized void getImageFileListAsync(final GridView gridView) {
        System.out.println(PHOTOS_FOLDER_URI);

        new Thread() {
            public void run() {
                try {
                    FileConnection fileConnection =
                        (FileConnection)Connector.open(PHOTOS_FOLDER_URI,
Connector.READ);
                    Enumeration fileList = fileConnection.list(JPEG_IMAGE_FILE_FILTER,
false);

                    while (fileList.hasMoreElements()) {
                        final String fileUri =
                            PHOTOS_FOLDER_URI + (String)fileList.nextElement();
                        System.out.println(fileUri);
                        imageFileURIs.addElement(fileUri);
                    }

                    fileConnection.close();

                    // Populate grid with the images
                    final int count = imageFileURIs.size();
                    System.out.println(count + " images found.");
                    Vector gridItems = new Vector();

                    for (int i = 0; i < count; ++i) {
                        GridItem gridItem =
                            new GridItem((String)imageFileURIs.elementAt(i), 230 / 3,
230 / 3);
                        gridItems.addElement(gridItem);
                    }

                    gridView.populateList(gridItems);
                }
                catch (IOException e) {
                    e.printStackTrace();
                }
                catch (SecurityException e) {
                    e.printStackTrace();
                }
            }
        }.start();
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
}
```

There are two important things to notice when we are loading images from the phone memory card instead of local images in the application itself. First, this has to be done in a separate thread so that we won't block the UI thread. Second, loading the images cannot be done with `Image.createImage()` method. `Image.createImage()` is used for loading local images (bundled with the application), but here we are accessing image files in the filesystem and thus, we need to use the FileConnection class. Therefore, let's add the appropriate method to `ImageCache` class:

```
    public static Image loadImageFromPhone(final String uri)
        throws IOException
    {
        Image image = null;

        try {
            FileConnection fconn = (FileConnection)Connector.open(uri);
            InputStream in = fconn.openInputStream();
            image = Image.createImage(in);
            in.close();
            fconn.close();
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
        }

        return image;
    }
```

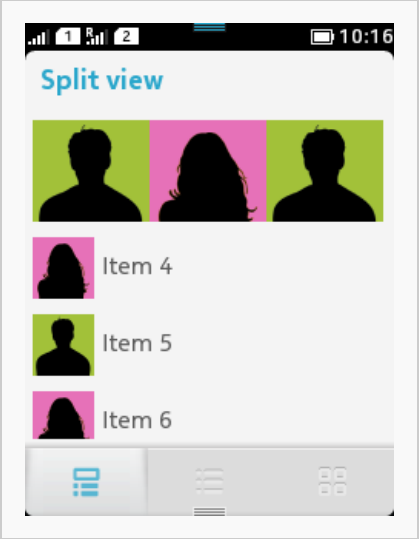
Add couple of more minor changes here and there, and we have a working application:

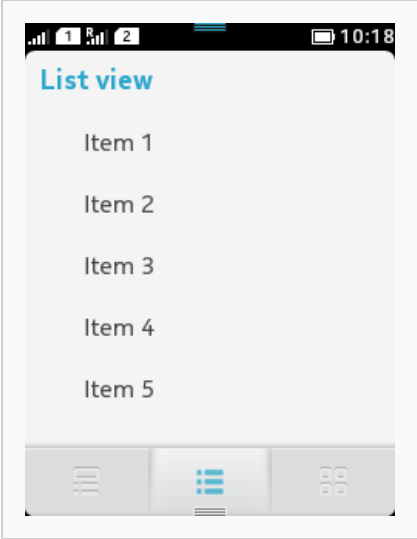
The full source code with `.jad` and `.jar` file is here: File:MyGalleryApp.zip.

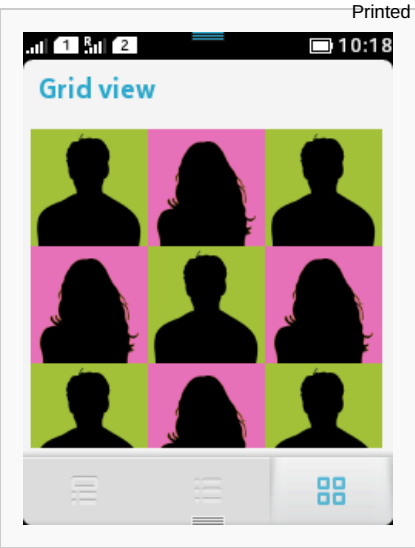
Composed UI

Composed UI resembles the other UI templates, but since it is a lot more complex, it has been published as an example instead. The example combines tab based navigation with list and grid based drill-down navigation model. The three main views - split, list and grid - are shown in the screenshots below.

(C) Copyright Nokia 2014. All rights reserved.







Besides the UI features, Composed UI demonstrates advanced image and record store management. Download source code and binaries from the project and check out the documentation.

Summary

The templates in the SDK are great assets for getting a quick start in Nokia Asha Java ME development. In addition to these new UI templates, the existing Series 40 templates have been ported to support the new Nokia Asha software platform too. For more advanced resources, see the code examples in the SDK and Nokia Developer website. See also Nokia Asha Design Guidelines for how to utilise best the new fresh Nokia Asha UI.

(C) Copyright Nokia 2014. All rights reserved.