Introduction to Lightweight User Interface Toolkit

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee (05 Jul 2012)
LWUIT now "properly" supported on Series 40 - see https://github.com/nokia-developer/lwuit-for-series-40.The article needs to be updated in line with this - and possible all content removed except pointer to current LWUIT resources.

- Lightweight User Interface Toolkit is a framework created by Sun to overcome the lack of rich GUI components available in the J2ME platform. The structural concept is based on Swing/AWT component type, using the techniques of JavaSE for mobile devices. LWUIT does not have compatibility with the canvas library, so there's no way to use drawings of a canvas interface developed by LWUIT. LWUIT toolkit supports CLDC1.1 MIDP2.0/CDC PBB / SE.
- Under the AWT library, the LWUIT uses the standard composite design pattern, which the father node of the tree of inheritance is the Component and the son Container is able to include another container, thus enriching the variability of GUIs in applications. The representative image of the tree of inheritance is shown below:
- In Swing components, it uses the layout techniques used in the JavaSE platform, and Themes, whose using together with Style, area capable of create a pattern GUI for an entire application. However, this pattern can be violated by changing the style of the component. (e.g. o.getTitleStyle().SetBgColor(0×00ffff)).
- One interesting detail to be mentioned about the Themes used in LWUIT is that they can be loaded in runtime. For this, it is necessary only to load a resource containing the themes and choose one theme to change the whole GUI style of the current application. A snippet of code to illustrate this execution will be represented below:

```
try {
    Resources res = Resources.open("/theme.res");
    UIManager.getInstance().setThemeProps(res.getTheme("default"));
} catch (IOException e) {
    e.printStackTrace();
}
```

- To create a Theme, there is a very interesting tool available by LWUIT toolkit, calling ResourceEditor. It is inside the LWUT_20080814\util\ folder (ResourceEditor.jar or ResourceEditor.exe). With this available tool you can create a pattern for the various components and check them (observe how it will be drawn in the screen) in a layout available in a right side bar application.
- Another very interesting feature of LWUIT is the technical transition of the screens. While you change from one container to another, there is a transition between them. That transition can be implemented with animation 2D or 3D (pre-requisites: JSR 184 Mobile 3D Graphics API) through the classes com.sun.lwuit.animations.CommonTransitions and com.sun.lwuit.animations.Transition3D. For instance, you can create a transition that simulates a rotating cube to right or left.
- In this article, I will implement a small example using the resources of a theme, layout and transition to serve as the basis for your future applications. First of all, you can download the library LWUIT. Finally, you may download the Theme (which we will use). (Note: See links at end of article. Copy the file to the resource directory / res of your project).

```
import java.io.IOException;

import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

import com.sun.lwuit.*;
import com.sun.lwuit.animations.Transition3D;
import com.sun.lwuit.events.ActionEvent;
import com.sun.lwuit.events.ActionListener;
import com.sun.lwuit.plaf.UIManager;
import com.sun.lwuit.util.Resources;
```


(C) Copyright Nokia 2014. All rights reserved.

```
public class FirstLWUITExample extends MIDlet implements ActionListener {

private static final String URL_THEME = "Theme_Default";
private static final String URL_RESOURCE = "/theme_default.res";
static Form[] forms;
private Command back = new Command("&lt;&lt;&lt;");
private Command next = new Command("&gt;&gt;&gt;");

/*

This constructor executes the static method com.sun.lwuit.Display.init() passing a
MIDlet reference as a
parameter. Next, the method setupTheme() is called. All it will do is to import the
theme and create 10
Objects FormNumbered and indexed in a primitive array.


*/

public FirstLWUITExample() {
    Display.init(this);
    this.setupTheme();
    forms = new Form[10];
    for (int i = 0; i < 10; i++) {
         forms[i] = new FormNumbered(i, this);
    }
}

/*

This method imports a resource by the method com.sun.lwuit.Resource.open (), whose
parameter is a
string, an URL of the file *.res. Shortly after, we recover the instance of
com.sun.lwuit.UIManager and
set the theme selecting the Theme we desire.

*/

private void setupTheme() {
    if (!((URL_RESOURCE == null) || (URL_THEME == null))) {
        try {
            Resources res = Resources.open(URL_RESOURCE);
            UIManager.getInstance().setThemeProps(res.getTheme(URL_THEME));
       } catch (IOException e) {
           System.err.println("Falha na importacao do resource/theme");
           e.printStackTrace();
      }
    } else
          System.err.println("URL null");
    }

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

protected void pauseApp() {
```


(C) Copyright Nokia 2014. All rights reserved.

```
// TODO Auto-generated method stub
}

/*

In the startApp() method of the Midlet class, we chose the FormNumbered indexed in the
position 0 of the
primitive array and runs the method show() to display it in the screen.


*/

protected void startApp() throws MIDletStateChangeException {
    forms[0].show();
}

/*

This method belongs to the interface ActionListener. It treats the events initiated by
commands "back"
and "next" from the class FormNumbered. If triggered the next command, moving a position
of the array
with a cube-shaped transition turning to the left. If the command is "back", returning a
position of
the array and transition, a cube spins to the right. In both cases, the arrays acts in
circles.


*/

public void actionPerformed(ActionEvent event) {
    Command eventCmd = event.getCommand();
    Form f = Display.getInstance().getCurrent();
    int num = Integer.parseInt(f.getTitle());
    System.out.println(num);
    boolean sentido = false;
    if (eventCmd == back) {
        sentido = true;
        num = (num > 0) ? num - 1 : 9;
    } else if (eventCmd == next) {
        sentido = false;
        num = (num < 9) ? num + 1 : 0;
    }

    f.setTransitionOutAnimator(Transition3D.createCube(300, sentido));
    forms[num].show();
}

/*

I created an internal class that extends the Class com.sun.lwuit.Form.


*/

class FormNumbered extends Form {
```

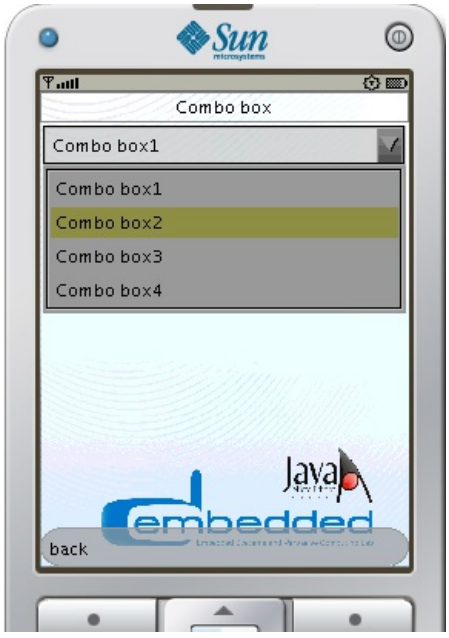

(C) Copyright Nokia 2014. All rights reserved.

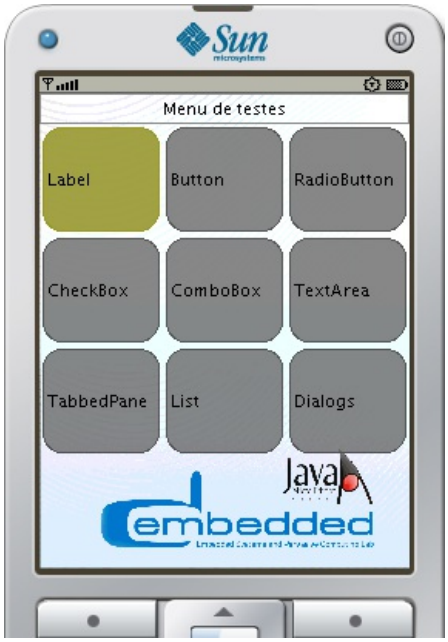
```
/*

The constructor calls the super class constructor passing as parameters: a numerical
value as String
object. Next, add two command, one to back and the other to follow.And sets a listener
to the events of
controls.


*/
public FormNumbered(int num, FirstLWUITExample app) {
    super(String.valueOf(num));
    this.addCommand(back);
    this.addCommand(next);
    this.setCommandListener(app);
}
}
}
```

These are another examples using interesting components of the LWUIT:





(C) Copyright Nokia 2014. All rights reserved.

- Picture1: A CheckBox list some options that you can select, in each selection is send a event as in a Button.
- Picture2: A ComboBox open a drop-down list that you can select just one option. It works as a List.
- Picture3: A GridLayout, where the components are placed in a matriz, n lines and m columns.
- Picture4: A TabbedPane contains any tabs, whose, each one, contains a Container.

Links

- Link site LWUIT official

(C) Copyright Nokia 2014. All rights reserved.