How to filter a list dynamically while typing with LCDUI - Part 3 (CustomItem List)

This article describes how to create a touch friendly incremental search UI with high level and low level LCDUI components.

This is the last of a three part article series that deal with this task. Part 1 focuses on Multiple Selection Lists while Part 2 deals with Single Selection Lists respectively. There are more available options, when implementing a similar non-touch interface, such as using `StringItem` objects for each item on the list and setting an `itemCommandListener` on each of them. Doing the same however for a Touch UI with high level LCDUI components, where a single tap can capture the user's selection, restricts the developer's available options. This series of articles describe these available options and point out each option's advantages and drawbacks. Check the links at the end of the article for more information.

Introduction

An Incremental Search UI is a method to progressively search for and filter through text. This kind of filter-as-you-type interface is becoming popular nowadays due to the ability to provide extremely fast search results, before even typing is over. Not only does the user no longer need to type the entire search string and hit the search button, but also scrolling down through long lists becomes redundant. While eSWT provides this functionality in the form of a `ShortedList` with FILTER option, this cannot be implemented on low end devices, like Series 40, because eSWT is supported only by Symbian.

This article describes an alternative way to create a dynamic filter-as-you-type list, by using a combination of high-level LCDUI components and low-level CustomItems that can receive touch events and support the `pointerPressed()` method. This option might eventually be the most user friendly in terms of UI for Series 40 devices when creating dynamic lists, compared to using a high level `ChoiceGroup` as described in parts 1 and 2. Using `CustomItems` however require some additional logic so that

- the application remains backward compatible with non Touch and Type devices
- the drawing of low-level `CustomItems` should be done in such a way, so that the active theme's colors do not make the `CustomItem` invisible

Backward compatibility with non Touch and Type

```
import com.nokia.mid.ui.LCDUIUtil;
```

This example makes use of the LCDUIUtil class. This means that only a touch-enabled Series 40 SDK can be used, i.e. the Nokia SDK 1.0 for Java or later.

(C) Copyright Nokia 2013. All rights reserved.

However it is possible in run time, to bypass the LCDUIUtil class, meaning that we can run the compiled MIDlet on non touch Series 40 devices by checking whether the current device supports touch.

If the device is a touch-enabled Series 40, then we need to enable the direct touch trait as follows:

```
 if(supportsS40Touch()) {
 //enables the single touch trait
 LCDUIUtil.setObjectTrait(item, "nokia.ui.s40.item.direct_touch", new Boolean(true));
}
```

The direct touch trait enables the user to pick an item from the list by tapping it once. By default, this trait is disabled, therefore the default behavior on Series 40 touch devices is double tapping for selecting an item. The first tap is used to highlight the item and the second to select it. We find however more user-friendly, easier and faster enabling the single tap for selection.

If the LCDUIUtil class is not found on the device, either a `NoClassFoundError` will occur or an exception will be thrown. This is how we can identify if the device is a Series 40 touch-enabled or a Series 40 older non touch:

```
//Checks if the device is Series 40 Touch and Type
   public static boolean supportsS40Touch()
   {
       try {
           Class.forName("com.nokia.mid.ui.LCDUIUtil");
           return true;
       }
       catch (NoClassDefFoundError e) {
        return false;
       }
       catch(Exception e){
        return false;
       }
   }
```

If the device is non-touch, we should add a Select Command on the `CustomItem` so that every time the user highlights the item, a select option should appear. Also an `ItemCommandListener` should be set on the item:

```
         //If the device is Series 40 Touch and Type
         if(supportsS40Touch()) {
          //enables the single touch trait
          LCDUIUtil.setObjectTrait(item, "nokia.ui.s40.item.direct_touch", new Boolean(true));
         }
         //If the device is older Series 40 with keyboard
         else {
          //adds a Select Command as Middle Softkey
          item.addCommand(selectCmd);
          item.setItemCommandListener(this);
         }
```

When the user chooses the "Select" command on a highlighted item, on a non-touch Series 40 device the following code is executed:

```
public void commandAction(Command c, Item item) {
 if (c == selectCmd) {
  TouchItem touchItem = (TouchItem) item;
  touchItem.pointerPressed(1, 1);
 }
```

This triggers the same part of the code that would have been executed, had the device been touch-enabled and the user tapped on the item.

Painting the CustomItem

A `CustomItem` is practically a canvas object, but it can be appended to a `Form` as an item. While the Form's interaction with the user is high-level and controlled by the device, what the item displays is low-level. The developer should ensure that the currently active background color, is not the same as the canvas' foreground color. Instead, the theme's foreground color should also be forced as a foreground color for the text displayed within the `CustomItem`:

This is how we can get the theme's foreground color:

```
public int getForegroundColor() {
 return Display.getDisplay(this).getColor(Display.COLOR_FOREGROUND);
}
```

and this is how we can apply it as a foreground color for the text within the item:

```
    public void paint(Graphics g, int width, int height) {

     g.setColor(midlet.getForegroundColor());
     g.drawRect(0, 0, width-2, height-2);
     g.drawString(label,(width-g.getFont().stringWidth(label)) / 2,
                        (height - g.getFont().getHeight()) / 2, 0);
    }
```

The MIDlet's code

```
import java.util.Vector;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
```


(C) Copyright Nokia 2013. All rights reserved.

```
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.ItemCommandListener;
import javax.microedition.lcdui.ItemStateListener;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

import com.nokia.mid.ui.LCDUIUtil;


public class ShortedSearch
 extends MIDlet
 implements CommandListener, ItemStateListener, ItemCommandListener  {

 TextField searchText;
 Form mainForm;
 String[] listItems;
 String selectedChoice;
 Alert alert;

 Command exitCmd = new Command("Exit", Command.EXIT, 0);
 Command selectCmd = new Command("Select", Command.OK, 0);

 protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
  //To Do
 }
 protected void pauseApp() {
  //To Do
 }
 protected void startApp() throws MIDletStateChangeException
 {
  mainForm = new Form("List of Countries");
  searchText = new TextField("Search", "", 15, TextField.ANY);
  mainForm.append(searchText);
  //This initializes the list of countries
  initList();

  //This populates the form with CustomItems
  updateList(listItems);
  //The MIDlet's commands
  mainForm.addCommand(exitCmd);
  mainForm.setCommandListener(this);
  //For listening to updates in the search TextField
  mainForm.setItemStateListener(this);
  Display.getDisplay(this).setCurrent(mainForm);
 }

 public void itemStateChanged(Item item)
 {
  String label = item.getLabel();
  //when the textfield is modified, i.e. text is added or deleted
  if(label.equals("Search"))
  {

   String typedText = searchText.getString();
   //if the search string is not the empty string
   if(!typedText.equals(""))
   {
    String[] newList = FilterList(listItems, typedText);
    //if there is at least a match
    if(newList != null){
     updateList(newList);
    }
    //if there is no match, the list should be emptied
    else {
     updateList(new String[0]);
    }
   }
   //if the search string is the empty string, the list should be set back to default
   else {
    updateList(listItems);
   }
  }

 }
 //Updates the list according to the input array
 public void updateList(String[] newItems)
 {
  //Deletes previously appended items
  for(int i = mainForm.size()-1; i>=1 ;i--) {
   mainForm.delete(i);
  }
  //Creates and appends new items
  for(int i = 0; i < newItems.length; i++) {

          TouchItem item = new TouchItem(newItems[i], 50, this);

          //If the device is Series 40 Touch and Type
          if(supportsS40Touch()) {
           //enables the single touch trait
           LCDUIUtil.setObjectTrait(item, "nokia.ui.s40.item.direct_touch", new Boolean(true));
          }
          //If the device is older Series 40 with keyboard
          else {
           //adds a Select Command as Middle Softkey
           item.addCommand(selectCmd);
           item.setItemCommandListener(this);
          }
          mainForm.append(item);
  }
 }
 //returns a new list of items based on the search text
 public String[] FilterList(String[] list, String filterText)
 {
  //makes the search non-case sensitive
  String textUpper = filterText.toUpperCase();
     String textLower = filterText.toLowerCase();
     char[] firstText = new char[1];
     char[] restText = new char[textLower.length()-1];
     textUpper.getChars(0, 1, firstText, 0);
     textLower.getChars(1, textLower.length(), restText, 0);
     String firstLetter = new String(firstText);
     String remainingLetters = new String(restText);
     filterText = firstLetter + remainingLetters;

     //a vector that stores the matching strings
  Vector matchingStrings = new Vector();
```


(C) Copyright Nokia 2013. All rights reserved.

```
  String[] result = null;
  for(int i = 0; i< list.length; i++) {
   //if the i item of the list starts with the search text
   if(list[i].indexOf(filterText, 0)==0)
    matchingStrings.addElement(list[i]);
  }
  int vectorSize = matchingStrings.size();
  //if at least one item is found in the list that starts with the search text
  if(vectorSize>0)
  {
   result=new String[vectorSize];
   for(int i = 0; i< vectorSize; i++)
   {
    result[i] = (String) matchingStrings.elementAt(i);
   }
  }
  return result;
 }
 //The default list of countries
 public void initList()
 {
  listItems = new String[196];
  listItems[0] = "Afghanistan";
  listItems[1] = "Albania";
  listItems[2] = "Algeria";
  listItems[3] = "Andorra";
  listItems[4] = "Angola";
  listItems[5] = "Antigua & Deps";
  listItems[6] = "Argentina";
  listItems[7] = "Armenia";
  listItems[8] = "Australia";
  listItems[9] = "Austria";
  listItems[10] = "Azerbaijan";
  listItems[11] = "Bahamas";
  listItems[12] = "Bahrain";
  listItems[13] = "Bangladesh";
  listItems[14] = "Barbados";
  listItems[15] = "Belarus";
  listItems[16] = "Belgium";
  listItems[17] = "Belize";
  listItems[18] = "Benin";
  listItems[19] = "Bhutan";
  listItems[20] = "Bolivia";
  listItems[21] = "Bosnia Herzegovina";
  listItems[22] = "Botswana";
  listItems[23] = "Brazil";
  listItems[24] = "Brunei";
  listItems[25] = "Bulgaria";
  listItems[26] = "Burkina";
  listItems[27] = "Burundi";
  listItems[28] = "Cambodia";
  listItems[29] = "Cameroon";
  listItems[30] = "Canada";
  listItems[31] = "Cape Verde";
  listItems[32] = "Central African Rep";
  listItems[33] = "Chad";
  listItems[34] = "Chile";
  listItems[35] = "China";
  listItems[36] = "Colombia";
  listItems[37] = "Comoros";
  listItems[38] = "Congo";
  listItems[39] = "Congo {Democratic Rep}";
  listItems[40] = "Costa Rica";
  listItems[41] = "Croatia";
  listItems[42] = "Cuba";
  listItems[43] = "Cyprus";
  listItems[44] = "Czech Republic";
  listItems[45] = "Denmark";
  listItems[46] = "Djibouti";
  listItems[47] = "Dominica";
  listItems[48] = "Dominican Republic";
  listItems[49] = "East Timor";
  listItems[50] = "Ecuador";
  listItems[51] = "Egypt";
  listItems[52] = "El Salvador";
  listItems[53] = "Equatorial Guinea";
  listItems[54] = "Eritrea";
  listItems[55] = "Estonia";
  listItems[56] = "Ethiopia";
  listItems[57] = "Fiji";
  listItems[58] = "Finland";
  listItems[59] = "France";
  listItems[60] = "Gabon";
  listItems[61] = "Gambia";
  listItems[62] = "Georgia";
  listItems[63] = "Germany";
  listItems[64] = "Ghana";
  listItems[65] = "Greece";
  listItems[66] = "Grenada";
  listItems[67] = "Guatemala";
  listItems[68] = "Guinea";
  listItems[69] = "Guinea-Bissau";
  listItems[70] = "Guyana";
  listItems[71] = "Haiti";
  listItems[72] = "Honduras";
  listItems[73] = "Hungary";
  listItems[74] = "Iceland";
  listItems[75] = "India";
  listItems[76] = "Indonesia";
  listItems[77] = "Iran";
  listItems[78] = "Iraq";
  listItems[79] = "Ireland {Republic}";
  listItems[80] = "Israel";
  listItems[81] = "Italy";
  listItems[82] = "Ivory Coast";
  listItems[83] = "Jamaica";
  listItems[84] = "Japan";
  listItems[85] = "Jordan";
  listItems[86] = "Kazakhstan";
  listItems[87] = "Kenya";
  listItems[88] = "Kiribati";
  listItems[89] = "Korea North";
  listItems[90] = "Korea South";
  listItems[91] = "Kosovo";
  listItems[92] = "Kuwait";
  listItems[93] = "Kyrgyzstan";
  listItems[94] = "Laos";
  listItems[95] = "Latvia";
  listItems[96] = "Lebanon";
  listItems[97] = "Lesotho";
```


(C) Copyright Nokia 2013. All rights reserved.

```
 listItems[98] = "Liberia";
 listItems[99] = "Libya";
 listItems[100] = "Liechtenstein";
 listItems[101] = "Lithuania";
 listItems[102] = "Luxembourg";
 listItems[103] = "Macedonia";
 listItems[104] = "Madagascar";
 listItems[105] = "Malawi";
 listItems[106] = "Malaysia";
 listItems[107] = "Maldives";
 listItems[108] = "Mali";
 listItems[109] = "Malta";
 listItems[110] = "Marshall Islands";
 listItems[111] = "Mauritania";
 listItems[112] = "Mauritius";
 listItems[113] = "Mexico";
 listItems[114] = "Micronesia";
 listItems[115] = "Moldova";
 listItems[116] = "Monaco";
 listItems[117] = "Mongolia";
 listItems[118] = "Montenegro";
 listItems[119] = "Morocco";
 listItems[120] = "Mozambique";
 listItems[121] = "Myanmar, {Burma}";
 listItems[122] = "Namibia";
 listItems[123] = "Nauru";
 listItems[124] = "Nepal";
 listItems[125] = "Netherlands";
 listItems[126] = "New Zealand";
 listItems[127] = "Nicaragua";
 listItems[128] = "Niger";
 listItems[129] = "Nigeria";
 listItems[130] = "Norway";
 listItems[131] = "Oman";
 listItems[132] = "Pakistan";
 listItems[133] = "Palau";
 listItems[134] = "Panama";
 listItems[135] = "Papua New Guinea";
 listItems[136] = "Paraguay";
 listItems[137] = "Peru";
 listItems[138] = "Philippines";
 listItems[139] = "Poland";
 listItems[140] = "Portugal";
 listItems[141] = "Qatar";
 listItems[142] = "Romania";
 listItems[143] = "Russian Federation";
 listItems[144] = "Rwanda";
 listItems[145] = "St Kitts & Nevis";
 listItems[146] = "St Lucia";
 listItems[147] = "Saint Vincent & the Grenadines";
 listItems[148] = "Samoa";
 listItems[149] = "San Marino";
 listItems[150] = "Sao Tome & Principe";
 listItems[151] = "Saudi Arabia";
 listItems[152] = "Senegal";
 listItems[153] = "Serbia";
 listItems[154] = "Seychelles";
 listItems[155] = "Sierra Leone";
 listItems[156] = "Singapore";
 listItems[157] = "Slovakia";
 listItems[158] = "Slovenia";
 listItems[159] = "Solomon Islands";
 listItems[160] = "Somalia";
 listItems[161] = "South Africa";
 listItems[162] = "South Sudan";
 listItems[163] = "Spain";
 listItems[164] = "Sri Lanka";
 listItems[165] = "Sudan";
 listItems[166] = "Suriname";
 listItems[167] = "Swaziland";
 listItems[168] = "Sweden";
 listItems[169] = "Switzerland";
 listItems[170] = "Syria";
 listItems[171] = "Taiwan";
 listItems[172] = "Tajikistan";
 listItems[173] = "Tanzania";
 listItems[174] = "Thailand";
 listItems[175] = "Togo";
 listItems[176] = "Tonga";
 listItems[177] = "Trinidad & Tobago";
 listItems[178] = "Tunisia";
 listItems[179] = "Turkey";
 listItems[180] = "Turkmenistan";
 listItems[181] = "Tuvalu";
 listItems[182] = "Uganda";
 listItems[183] = "Ukraine";
 listItems[184] = "United Arab Emirates";
 listItems[185] = "United Kingdom";
 listItems[186] = "United States";
 listItems[187] = "Uruguay";
 listItems[188] = "Uzbekistan";
 listItems[189] = "Vanuatu";
 listItems[190] = "Vatican City";
 listItems[191] = "Venezuela";
 listItems[192] = "Vietnam";
 listItems[193] = "Yemen";
 listItems[194] = "Zambia";
 listItems[195] = "Zimbabwe";
}

//Displays alert with selected country after the user selects one
public void showSelected(String selectedItem) {
 Alert alert = new Alert("Info", selectedItem, null,AlertType.INFO);
       alert.setTimeout(3000);
       Display.getDisplay(this).setCurrent(alert, mainForm);
}
//used for correctly drawing the CustomItems
public int getForegroundColor() {
 return Display.getDisplay(this).getColor(Display.COLOR_FOREGROUND);
}
//Checks if the device is Series 40 Touch and Type
   public static boolean supportsS40Touch()
   {
       try {
           Class.forName("com.nokia.mid.ui.LCDUIUtil");
           return true;
       }
       catch (NoClassDefFoundError e) {
        return false;
       }
```


(C) Copyright Nokia 2013. All rights reserved.

```
        catch(Exception e){
         return false;
        }
    }

 public void commandAction(Command c, Displayable d) {
  if(c == exitCmd) {
   notifyDestroyed();
  }
 }
 public void commandAction(Command c, Item item) {
  if (c == selectCmd) {
   TouchItem touchItem = (TouchItem) item;
   touchItem.pointerPressed(1, 1);
  }

 }
}
```

The CustomItem's class

```
import javax.microedition.lcdui.CustomItem;
import javax.microedition.lcdui.Graphics;

class TouchItem extends CustomItem {

    String label;
    int itemsHeight;
    ShortedSearch midlet;

    public TouchItem( String label, int itemsHeight, ShortedSearch midlet) {
        super( "" );
        this.label = label;
        this.itemsHeight = itemsHeight;
        this.midlet = midlet;
    }

    public void paint(Graphics g, int width, int height) {

     g.setColor(midlet.getForegroundColor());
     g.drawRect(0, 0, width-2, height-2);
     g.drawString(label,(width-g.getFont().stringWidth(label)) / 2,
                        (height - g.getFont().getHeight()) / 2, 0);
    }

    protected int getMinContentWidth() {
        return 250;
    }

    protected int getMinContentHeight() {
        return itemsHeight;
    }

    protected int getPrefContentWidth(int arg0) {
        return 250;
    }

    protected int getPrefContentHeight(int arg0) {
        return itemsHeight;
    }

    protected void pointerDragged(int x, int y) {
     //To do
    }
    protected void pointerPressed(int x, int y) {
  midlet.showSelected(label);
 }
 protected void pointerReleased(int x, int y) {
  //To do
 }
    protected void keyPressed(int keyCode) {
     midlet.showSelected(label);
    }
}
```

Resources

The source code of this MIDlet is available for download from here: File:ShortedSearchSource.zip

The binary files of this MIDlet are available for download from here: File:ShortedSearchBinaries.zip

See also

- How to filter a list dynamically while typing with LCDUI - Part 1 (Multiple Selection)
- How to filter a list dynamically while typing with LCDUI - Part 2 (Single Selection)
- eSWT's ShortedList with FILTER option
- LCDUIUtil

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.