How to filter a list dynamically while typing with LCDUI - Part 1 (Multiple Selection)

This article describes how to create a touch friendly incremental search UI with high level LCDUI components. This is the first of a three part article series that deal with this task. Part 1 focuses on Multiple Selection Lists while Part 2 and Part 3 deal with Single Selection Lists and CustomItem 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 high level LCDUI components and more precisely Multiple Selection Lists.

Main Components and Implementation Design for the Filtered Search

There are two main components in this MIDlet. A TextField and a ChoiceGroup of BUTTON or MULTIPLE type. Both can generate ItemStateChanged events as long as an ItemStateListener is set to the Form that contains the components.While the user types the search string in the TextField, this triggers the filtering in the list of results. If the search string matches the beginning of an item in the list, the list is updated with only the matched items, otherwise it is emptied. If the search string becomes the empty string, that means the user has deleted any previous searches and needs to start over, therefore the original list of items should be shown:

```
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 empty the list
  else
```


(C) Copyright Nokia 2014. All rights reserved.

```
      updateList(new String[0]);
    }
    //if the search string is the empty string
    //set the list back to the default
    else
    {
     updateList(listitems);
    }
   }
```

The search is non case sensitive and the results are stored in a vector. This is the code for the filtering:

```
  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;


   Vector v=new Vector();
   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)
     v.addElement(list[i]);
   }
   int vectorsize=v.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) v.elementAt(i);
    }
   }
   return result;
  }
```

Multiple Selection versus Single Selection List

In this example, a Multiple Selection List (i.e. a ChoiceGroup of type BUTTON or MULTIPLE) was chosen over a Single Selection List (i.e. a ChoiceGroup of type EXCLUSIVE). There are certain advantages and disadvantages for this choice. The major advantage is that a Multiple List, can have all its items unselected. This is important in case there are no matching results. If the same application were to be implemented with a Single Selection List, at least one dummy result should be always displayed.

(C) Copyright Nokia 2014. All rights reserved.

This is because Single Selection Lists, require that exactly one item is always selected. Another advantage of using a Multiple Selection List, is the obvious case where two or more items from the list should be selected. In this case the design of this example could be easily modified by adding a Command to the Form, so that the user first selects the number of choosen items and then finalizes his or her selection by clicking on the Command. This example, as it is written here, considers the case where only one item is needed, and a selection can be made instantly by just clicking on the item, rather than adding a separate Command. The code snippet below, runs within the ItemStateChanged method, identifies the chosen item and then restores the list:

```
  if(label.equals(""))
  {
   int selectedindex=findIndex();
   //if an item is selected
   if(selectedindex!=-1)
   {
    selectedchoice=choice.getString(selectedindex);
    //shows alert
    alert=new Alert("Info",selectedchoice,null,AlertType.INFO);
    alert.setTimeout(500);
    Display.getDisplay(this).setCurrent(alert,mainform);
    //unselects the selected item
    boolean[] flags=new boolean[choice.size()];
    for(int i=0;i<flags.length;i++)
    {
     flags[i]=false;
    }
    choice.setSelectedFlags(flags);
   }
  }
```

The disadvantage of using a Multiple Selection List, in the case where only one item from the list is selected every time, is that it is not user friendly. Multiple Selection Lists look like Check Boxes and the user normally expects to be able to select multiple items, instead of just one. Part 2 and Part 3 of this series of articles describe alternative solutions to this problem. See the links at the bottom of this page for more information.

The MIDlet's code

```
import java.util.Vector;

import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.ItemStateListener;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;


public class TextShortedListMIDlet extends MIDlet implements ItemStateListener,
CommandListener  {
```


(C) Copyright Nokia 2014. All rights reserved.

```
TextField searchtext;
Form mainform;
Display display;
String[] listitems;
ChoiceGroup choice;
String selectedchoice;
Alert alert;
Command exitCmd=new Command("Exit",Command.EXIT,0);
protected void destroyApp(boolean arg0) throws MIDletStateChangeException {}
protected void pauseApp() {}
protected void startApp() throws MIDletStateChangeException
{
 mainform=new Form("");
 Display display;
 searchtext=new TextField("Search","",15,TextField.ANY);
 initList();
 choice=new ChoiceGroup("",ChoiceGroup.BUTTON,listitems,null);

 display=Display.getDisplay(this);

 mainform.append(searchtext);
 mainform.append(choice);
 mainform.addCommand(exitCmd);
 mainform.setCommandListener(this);
 mainform.setItemStateListener(this);
 display.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 empty the list
   else
    updateList(new String[0]);
  }
  //if the search string is the empty string
  //set the list back to the default
  else
  {
   updateList(listitems);
  }
 }
 //when the user selects an item
 if(label.equals(""))
 {
  int selectedindex=findIndex();
```


(C) Copyright Nokia 2014. All rights reserved.

```
  //if an item is selected
  if(selectedindex!=-1)
  {
   selectedchoice=choice.getString(selectedindex);
   //shows alert
   alert=new Alert("Info",selectedchoice,null,AlertType.INFO);
   alert.setTimeout(500);
   Display.getDisplay(this).setCurrent(alert,mainform);
   //unselects the selected item
   boolean[] flags=new boolean[choice.size()];
   for(int i=0;i<flags.length;i++)
   {
    flags[i]=false;
   }
   choice.setSelectedFlags(flags);
  }
 }
}

public void updateList(String[] newitems)
{
 choice.deleteAll();
 for(int i=0;i<newitems.length;i++)
 {
  choice.append(newitems[i], null);
 }
}
//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;


 Vector v=new Vector();
 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)
   v.addElement(list[i]);
 }
 int vectorsize=v.size();
 //if at least one item is found in the list
 //that starts with the search text
 if(vectorsize>0)
 {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    result=new String[vectorsize];
    for(int i=0;i<vectorsize;i++)
    {
     result[i]=(String) v.elementAt(i);
    }
   }
   return result;
  }
  //The original 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";
```


(C) Copyright Nokia 2014. All rights reserved.

```
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";
listitems[98]="Liberia";
listitems[99]="Libya";
listitems[100]="Liechtenstein";
```


(C) Copyright Nokia 2014. All rights reserved.

```
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";
```


(C) Copyright Nokia 2014. All rights reserved.

```
 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";
}
//finds the index of the selected item, returns -1 if 0 items found
public int findIndex()
{
 int result=-1;
 boolean[] flags=new boolean[choice.size()];
 for(int i=0;i<flags.length;i++)
 {
  flags[i]=false;
 }
 choice.getSelectedFlags(flags);
 for(int i=0;i<flags.length;i++)
 {
  if(flags[i]==true)
   result=i;
 }
 return result;
```


(C) Copyright Nokia 2014. All rights reserved.

```
  }
  public void commandAction(Command c, Displayable d) {
   if(c==exitCmd)
   {
    notifyDestroyed();
   }
  }
 }
```

Resources

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

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

See also

- How to filter a list dynamically while typing with LCDUI - Part 2 (Single Selection)
- How to filter a list dynamically while typing with LCDUI - Part 3 (CustomItem List)
- eSWT's ShortedList with FILTER option

(C) Copyright Nokia 2014. All rights reserved.