Android

Adapters, Saving Activity State

2010.04.09

Database Laboratory
Hanyang Univ.

Customizing the To-Do List ArrayAdapter

- This example extends the To-Do List project, storing each item as a *ToDoItem* object that includes the date each item was created
- You will extend *ArrayAdapter* to bind a collection of *ToDoItem* objects to the *ListView* and customize the layout used to display each List View item

1. Return to the To-Do List project. Create a new *ToDoItem* class that stores the task and its creation date
   - Override the *toString* method to return a summary of the item data

Customizing the To-Do List ArrayAdapter

```
package com.paad.todolist;

import java.text.SimpleDateFormat;
import java.util.Date;

public class ToDoItem {
    String task;
    Date created;

    public String getTask() {
        return task;
    }

    public Date getCreated() {
        return created;
    }

    public ToDoItem(String _task) {
        this(_task, new Date(java.lang.System.currentTimeMillis()));
    }

    public ToDoItem(String _task, Date _created) {
        task = _task;
        created = _created;
    }

@Override
public String toString() {
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
    String dateString = sdf.format(created);
    return "(" + dateString + ")" + task;
    }
}
```

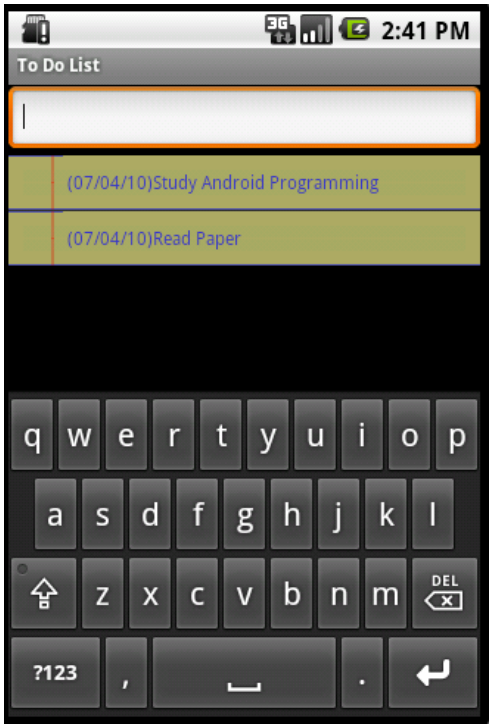
Customizing the To-Do List ArrayAdapter

2. **Open the *ToDoList* Activity, and modify the *ArrayList* and *ArrayAdapter* variable types to store *ToDoItem* objects rather than Strings**
   - You'll then need to modify the *onCreate* method to update the corresponding variable initialization
   - You'll also need to update the *onKeyListener* handler to support the *ToDoItem* objects

```
  private ArrayList<ToDoItem> todoItems;
  private ListView myListView;
  private EditText myEditText;
  private ArrayAdapter<ToDoItem> aa;
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

  // Inflate your view
    setContentView(R.layout.main);

  // Get references to UI widgets
  myListView = (ListView)findViewById(R.id.myListView);
  myEditText = (EditText)findViewById(R.id.myEditText);

  todoItems = new ArrayList<ToDoItem>();
  int resID = R.layout.to_dolist_item;
  aa = new ArrayAdapter<ToDoItem>(this, resID, todoItems);
  myListView.setAdapter(aa);

  // Assign the KeyListener to the DPad button to add new items
  myEditText.setOnKeyListener(new OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
      if (event.getAction() == KeyEvent.ACTION_DOWN)
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
        {
          ToDoItem newItem;
          newItem = new ToDoItem(myEditText.getText().toString());
          todoItems.add(0, newItem);
          myEditText.setText("");
          aa.notifyDataSetChanged();
          return true;
      }
      return false;
    }
  });
  registerForContextMenu(myListView);
}
```

Customizing the To-Do List ArrayAdapter

3. If you run the Activity, it will now display each to-do item, as shown in Figure



Customizing the To-Do List ArrayAdapter

4. Now you can create a custom layout to display each to-do item

- It will be used to show the creation date of each to-do item

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
 <TextView
    android:layout_width="wrap_content"
    android:id="@+id/rowDate"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:scrollbars="vertical"
    android:fadingEdge="vertical"
    android:textColor="@color/notepad_text"
    android:layout_alignParentRight="true">
  </TextView>
  <TextView
  android:layout_width="wrap_content"
  android:id="@+id/row"
  android:layout_height="fill_parent"
  android:scrollbars="vertical"
  android:fadingEdge="vertical"
  android:textColor="@color/notepad_text"
  android:layout_alignParentLeft="@id/rowDate">
  </TextView>
</RelativeLayout>
```

Customizing the To-Do List ArrayAdapter

5. Create a new class (*ToDoItemAdapter*) that extends an *ArrayAdapter* with a *ToDoItem*-specific variation

- Override *getView* to assign the task and date properties in the *ToDoItem* object to the Views in the layout you created in Step 4

```
package com.paad.todolist;

import java.text.SimpleDateFormat;
import android.content.Context;
import java.util.*;
import android.view.*;
import android.widget.*;

public class ToDoItemAdapter extends ArrayAdapter<ToDoItem> {

  int resource;

  public ToDoItemAdapter(Context _context, int _resource, List<ToDoItem> _items) {
    super(_context, _resource, _items);
    resource = _resource;
  }
```

Customizing the To-Do List ArrayAdapter

```
  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    LinearLayout todoView;

    ToDoItem item = getItem(position);

    String taskString = item.getTask();
    Date createdDate = item.getCreated();
    SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy");
    String dateString = sdf.format(createdDate);

    if (convertView == null) {
      todoView = new LinearLayout(getContext());
      String inflater = Context.LAYOUT_INFLATER_SERVICE;
      LayoutInflater vi = (LayoutInflater)getContext().getSystemService(inflater);
      vi.inflate(resource, todoView, true);
    } else {
      todoView = (LinearLayout) convertView;
    }

    TextView dateView = (TextView)todoView.findViewById(R.id.rowDate);
    TextView taskView = (TextView)todoView.findViewById(R.id.row);

    dateView.setText(dateString);
    taskView.setText(taskString);

    return todoView;
  }
}
```

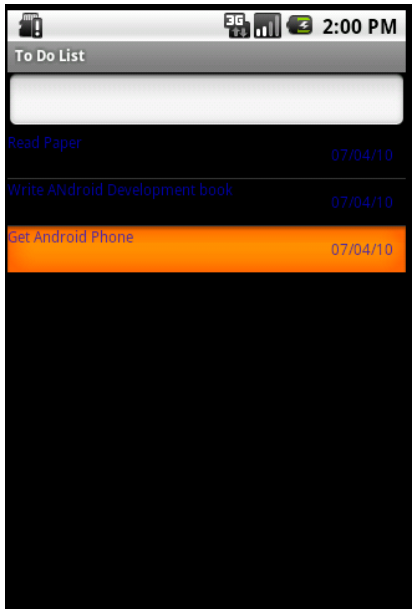
Customizing the To-Do List ArrayAdapter

6. Finally, replace the *ArrayAdapter* declaration with a *ToDoItemAdapter*

```
private ToDoItemAdapter aa;
```

```
aa = new ToDoItemAdapter(this, resID, todoItems);
```

7. Now, you run your Activity



Saving the To-Do List Activity State

- **In this example, you'll start to save the application state of the To-Do list application across sessions**
- **The instance state in the *ToDoList* Activity consists of three variables**
  - Is a new item being added?
  - What text exists in the new item entry textbox?
  - What is the currently selected item?
- **Using the Activity's default Shared Preference, you can store each of these values and update the UI when the Activity is restarted**

- ***Later in this chapter, you'll learn how to use the SQLite database to persist the to-do items as well***

Saving the To-Do List Activity State

1. **Start by adding static String variables to use as preference keys.**

```
static final private String TEXT_ENTRY_KEY = "TEXT_ENTRY_KEY";
static final private String ADDING_ITEM_KEY = "ADDING_ITEM_KEY";
static final private String SELECTED_INDEX_KEY = "SELECTED_INDEX_KEY";
```

2. **Next, override the *onPause* method. Get the Activity's private Shared Preference object, and get its Editor object**
   - Using the keys you created in Step 1, store the instance values based on whether a new item is being added and any text in the "new item" Edit Box.

```
protected void onPause(){
  super.onPause();

  // Get the activity preferences object.
  SharedPreferences uiState = getPreferences(0);
  // Get the preferences editor.
  SharedPreferences.Editor editor = uiState.edit();

  // Add the UI state preference values.
  editor.putString(TEXT_ENTRY_KEY, myEditText.getText().toString());
  editor.putBoolean(ADDING_ITEM_KEY, addingNew);

  // Commit the preferences.
  editor.commit();
}
```

Saving the To-Do List Activity State

3. Write a *restoreUIState* method that applies the instance values you recorded in Step 2 when the application restarts

- Modify the *onCreate* method to add a call to the *restoreUIState* method at the very end.

```
public void onCreate(Bundle icicle) {
// [ ... existing onCreate logic ... ]
    restoreUIState();
 }
```

```
private void restoreUIState() {
  // Get the activity preferences object.
  SharedPreferences settings = getPreferences(0);

  // Read the UI state values, specifying default values.
  String text = settings.getString(TEXT_ENTRY_KEY, "");
  Boolean adding = settings.getBoolean(ADDING_ITEM_KEY, false);

  // Restore the UI to the previous state.
  if (adding) {
    addNewItem();
    myEditText.setText(text);
  }
}
```

Saving the To-Do List Activity State

4. Record the index of the selected item using the *onSaveInstanceState / onRestore InstanceState* mechanism.

- It's then only saved and applied if the application is killed without the user's explicit instruction.

```
public void onSaveInstanceState(Bundle savedInstanceState) {
  savedInstanceState.putInt(SELECTED_INDEX_KEY, myListView.getSelectedItemPosition());

  super.onSaveInstanceState(savedInstanceState);
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
  int pos = -1;

  if (savedInstanceState != null)
    if (savedInstanceState.containsKey(SELECTED_INDEX_KEY))
      pos = savedInstanceState.getInt(SELECTED_INDEX_KEY, -1);

  myListView.setSelection(pos);
}
```

❖ **When you run the To-Do List application, you should now see the UI state persisted across sessions.**