

EMBEDDED SYSTEMS PROGRAMMING 2014-15

Application Tip: Saving State

THE PROBLEM

- How to save the state (of a UI, for instance) so that it survives even when the application is closed/killed
- The state should be restored when the application is opened again: to the user it is as if the application has always been there
- We will consider a UI with an editable text field, a checkbox, and a seekbar
- Solution: use the facilities provided by (and follow the rules dictated by) the platform's frameworks

ANDROID: ACTIVITIES AND APPLICATIONS

- UIs are managed inside **activities**
- An **application** may contain several activities
- Activities are independent by a large extent, but they share some resources associated with the application (e.g., the user ID)
- An activity may be destroyed by the OS while the application that hosts it survives. Later on, a new **instance** of the same activity may be created

ANDROID: INSTANCE STATE

- If an activity is destroyed and recreated by the OS (e.g., because the screen is rotated):
the new activity is a new instance of the old one
- If an activity is destroyed due to normal app behavior (user presses the *Back* button, `destroy()` is invoked):
“the activity instance is gone forever”
- **Instance state:** information associated with an activity instance

ANDROID: PERSISTENT STATE

- **Persistent state:** information associated with an application (e.g., user preferences)
- Persistent state must be preserved between different runs of an application. Instance state must not.

THE TIP: ANDROID, INSTANCE STATE (1/2)

- Consider the `HelloWithButton` example
- When the screen orientation changes, the `HelloWithButton` activity is destroyed, then a new instance of the activity is launched to redraw the screen according to the new orientation
- Let's modify the `HelloWithButton` example so that the instance state is preserved

THE TIP: ANDROID, INSTANCE STATE (2/2)



- The state of the `TextView` must be saved
- Platform solution: save it in the `savedInstanceState` Bundle



ANDROID CODE (1/3)

- Only one source file: `HelloWithButton.java`

```
package it.unipd.dei.es1011.saveinstancestate;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.LinearLayout;

public class HelloWithButton extends Activity
{
    /* Class variables */
    private TextView tv;
    private Button bu;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Create the TextView
        tv = new TextView(this);
        tv.setText("Press the button, please");
    }
}
```

...

ANDROID CODE (2/3)

```
...  
  
    // Restore TextView state from the savedInstanceState  
    if (savedInstanceState != null)  
    {  
        String strValue = savedInstanceState.getString("strTV");  
        if (strValue != null) tv.setText(strValue);  
    }  
  
    // Create the Button  
    bu = new Button(this);  
    bu.setText("Press me");  
  
    // Set the action to be performed when the button is pressed  
    bu.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) {  
            // Perform action on click  
            tv.setText("Good job!");  
        }  
    });  
  
    // Create the layout  
    LinearLayout mylayout = new LinearLayout(this);  
  
    // Add the UI elements to the layout  
    mylayout.addView(bu);  
    mylayout.addView(tv);  
  
    // Display the layout  
    setContentView(mylayout);  
}
```

...

ANDROID CODE (3/3)

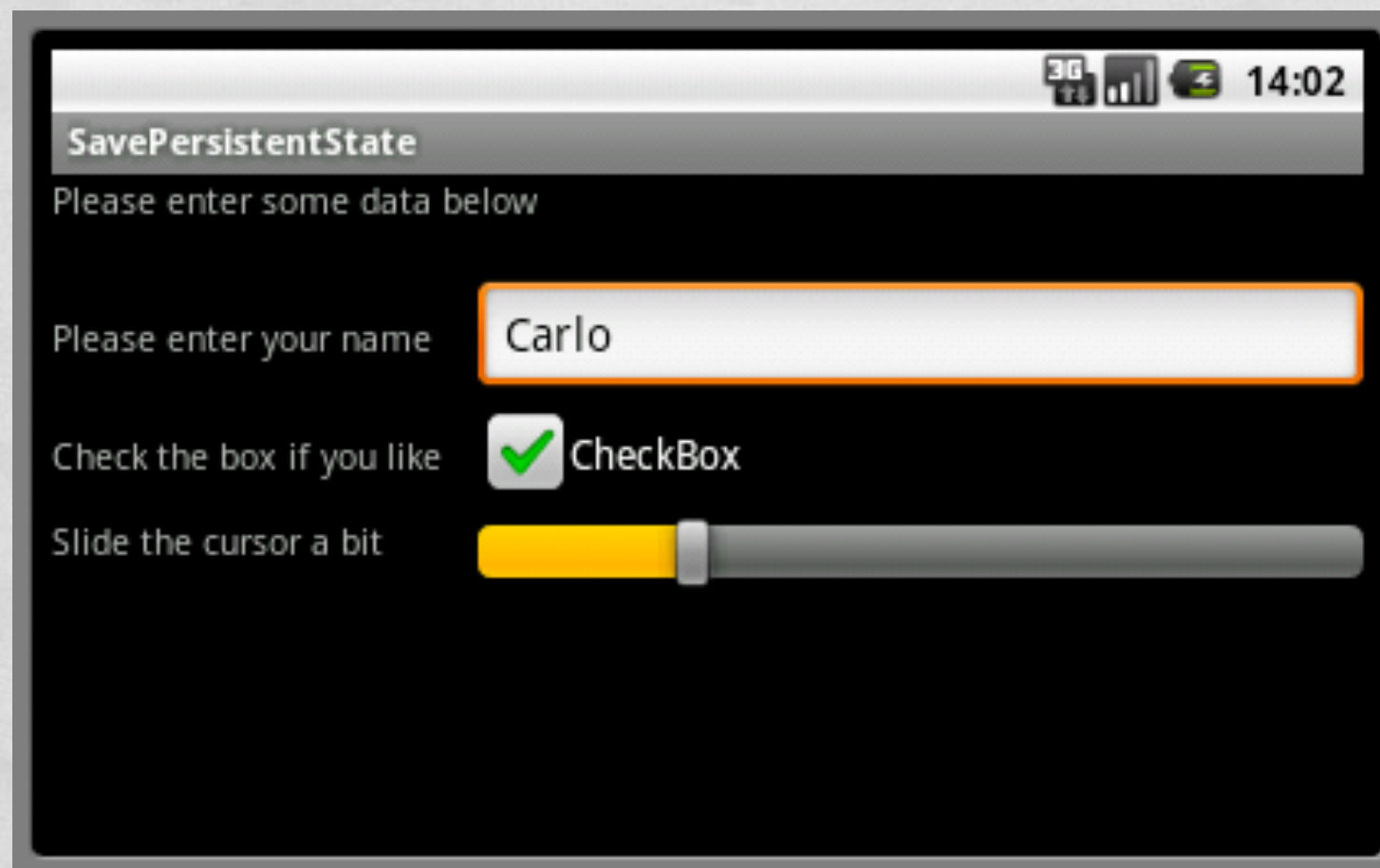
```
...  
/** Called when the system is about to pause the activity because it is  
 * resuming a previous one. This method allows you to save any  
 * dynamic INSTANCE state in your activity into the given Bundle,  
 * to be later received in onCreate(Bundle) if the activity needs  
 * to be re-created.  
 * Note: PERSISTENT state (which is different from instance state!)  
 * should be saved in the onPause() method because onSaveInstanceState()  
 * is not part of the life cycle callbacks, hence it will not be called  
 * in every situation */  
@Override  
public void onSaveInstanceState(Bundle savedInstanceState)  
{  
    // NOTE: with the implementation of this method inherited from  
    // Activity, some widgets save their state in the bundle by default.  
    // Once the user interface contains AT LEAST one non-autosaving  
    // element, you should provide a custom implementation of  
    // the method  
  
    String strTV = tv.getText().toString();  
    savedInstanceState.putString("strTV", strTV);  
  
    super.onSaveInstanceState(savedInstanceState);  
}  
}
```


THE TIP: ANDROID, PERSISTENT STATE (1/2)

- Back on track: the UI state must now survive even when the application is closed/killed (this is not the case for HelloWithButton: try force-quitting it)
- Platform solution: save data as **SharedPreferences**

THE TIP: ANDROID, PERSISTENT STATE (2/2)

- New application: the UI contains an editable text field, a checkbox, and a seekbar



ANDROID CODE (1/5)

- Source files:
 - `PersistenceActivity.java`
- Other resources:
 - `main.xml` (UI layout),
 - `strings.xml` (UI strings)

ANDROID CODE (2/5)

- strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">SavePersistentState</string>
    <string name="hello">Please enter some data below</string>
    <string name="please1">Please enter your name</string>
    <string name="please2">Check the box if you like</string>
    <string name="please3">Slide the cursor a bit</string>
</resources>
```


ANDROID CODE (3/5)

• main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:text="@string/hello" android:minHeight="32px"/>
<LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content" android:id="@+id/linearLayout1">
    <TextView android:text="@string/please1" android:id="@+id/textView1" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:minWidth="130px"></TextView>
    <EditText android:text="" android:id="@+id/editText1" android:layout_height="wrap_content"
        android:layout_width="fill_parent"></EditText>
</LinearLayout>
<LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content"
    android:id="@+id/linearLayout2">
    <TextView android:text="@string/please2" android:id="@+id/textView2" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:minWidth="130px"></TextView>
    <CheckBox android:text="CheckBox" android:id="@+id/checkbox1" android:layout_width="wrap_content"
        android:layout_height="wrap_content"></CheckBox>
</LinearLayout>
<LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content"
    android:id="@+id/linearLayout3">
    <TextView android:text="@string/please3" android:id="@+id/textView3" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:minWidth="130px"></TextView>
    <SeekBar android:layout_width="match_parent" android:layout_height="wrap_content" android:id="@+id/seekBar1"></SeekBar>
</LinearLayout>
</LinearLayout>
```


ANDROID CODE (4/5)

- PersistenceActivity.java (1/2)

```
package it.unipd.dei.es1011.savepersistentstate;

import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.CheckBox;
import android.widget.SeekBar;

public class PersistenceActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Set the view
        setContentView(R.layout.main);

        // Get persistent data stored as SharedPreferences
        SharedPreferences preferences = getPreferences(MODE_PRIVATE);
        String str_et = preferences.getString("editTextValue", null);
        boolean bln_cb = preferences.getBoolean("checkBoxValue", false);
        int int_sb = preferences.getInt("seekBarValue", 0);

        // Get references to widgets and set them according to persistent data
        EditText et = (EditText)findViewById(R.id.editText1);
        et.setText(str_et);
        CheckBox cb = (CheckBox)findViewById(R.id.checkBox1);
        cb.setChecked(bln_cb);
        SeekBar sb = (SeekBar)findViewById(R.id.seekBar1);
        sb.setProgress(int_sb);
    }
}
```


ANDROID CODE (5/5)

- PersistenceActivity.java (2/2)

```
...  
/** Called as part of the activity lifecycle when an activity is going  
 * into the background, but has not (yet) been killed.  
 * The counterpart to onResume(). */  
@Override  
protected void onPause()  
{  
    super.onPause();  
  
    // Store values between instances here  
    SharedPreferences preferences = getPreferences(MODE_PRIVATE);  
    SharedPreferences.Editor editor = preferences.edit();  
  
    // Get references to widgets and read the status  
    // of them all  
    EditText et = (EditText) findViewById(R.id.editText1);  
    String str_et = et.getText().toString();  
    CheckBox cb = (CheckBox) findViewById(R.id.checkBox1);  
    boolean bln_cb = cb.isChecked();  
    SeekBar sb = (SeekBar) findViewById(R.id.seekBar1);  
    int int_sb = sb.getProgress();  
  
    // Store status in the preferences  
    editor.putString("editTextValue", str_et);  
    editor.putBoolean("checkBoxValue", bln_cb);  
    editor.putInt("seekBarValue", int_sb);  
  
    // Commit to storage  
    editor.commit();  
}  
}
```


LAST MODIFIED: MARCH 23, 2015

COPYRIGHT HOLDER: CARLO FANTOZZI (FANTOZZI@DEI.UNIPD.IT)
LICENSE: [CREATIVE COMMONS ATTRIBUTION SHARE-ALIKE 3.0](https://creativecommons.org/licenses/by-sa/3.0/)