Mobile Development
Lecture 11: Activity State and Preferences

Mahmoud El-Gayyar
elgayyar@ci.suez.edu.eg
Elgayyar.weebly.com

Activity Instance State

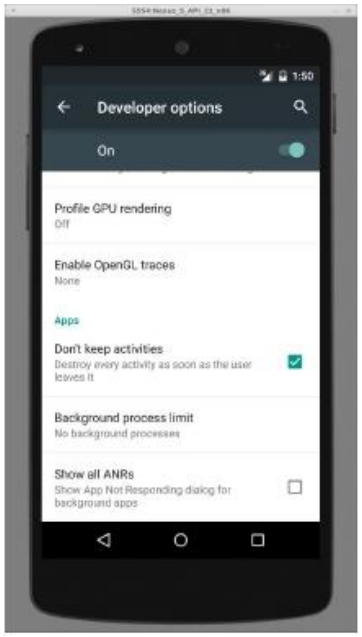
- *Instance state: Current state of an activity.*
  - Which boxes are checked
  - Any text typed into text boxes
  - Values of any private fields
- *Example: In the app at right, the*
  - instance state is that the
  - Don checkbox is checked, and the
  - Don image is showing.

Lost activity state

- *Several actions can cause your activity state to be lost:*
  - When you go from one activity to another and back, within same app
  - When you launch another app and then come back
  - When you rotate the device's orientation from portrait to landscape
  - ...

Simulating state change in AVD

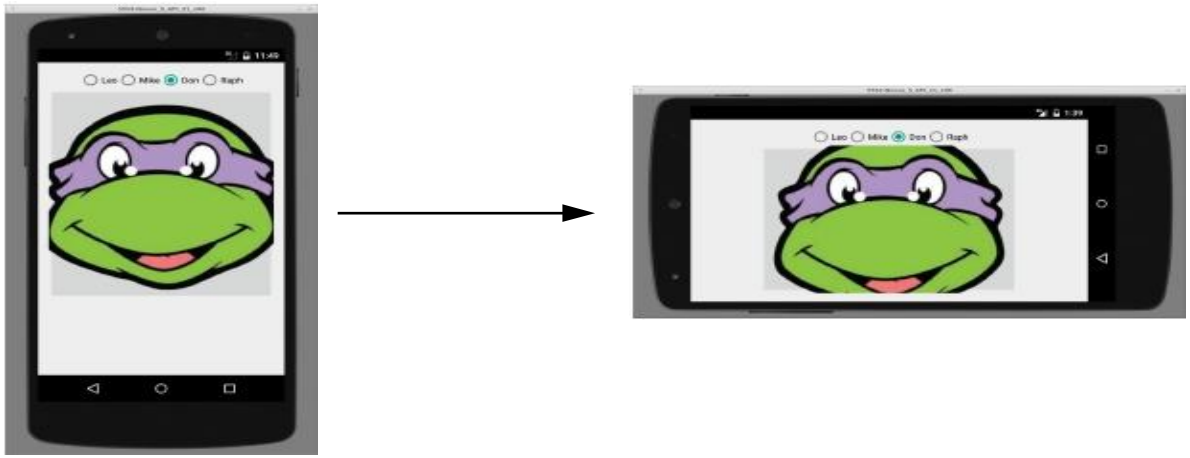
- *Testing orientation change: press Ctrl-F11 (link)*
- *Testing activity shutdown (onDestroy):*
  - Settings → Developer options → Don't keep activities
  - Developer options → Background process limit → No bg processes



Handling Rotation

- *A quick way to retain your activity's GUI state on rotation is to set the configChanges attribute of the activity in AndroidManifest.xml.*
  - This doesn't solve the other cases like loading other apps/activities.

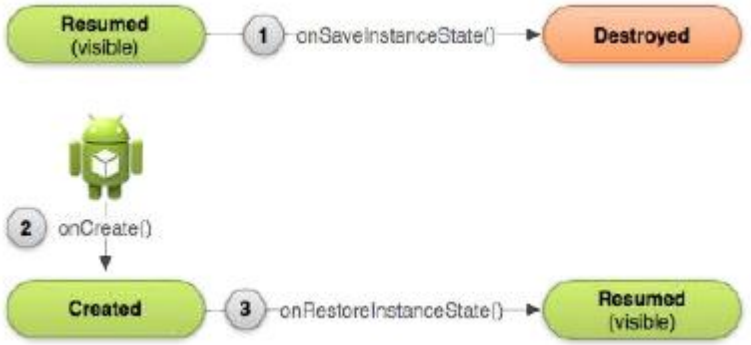
```
<activity android:name=".MainActivity"
        android:configChanges="orientation|screenSize"
        ...>
```



onSaveInstanceStace Method

- *When an activity is being destroyed, the event method onSaveInstanceState is also called.*
  - This method should save any "non-persistent" state of the app.
  - **non-persistent state:** Stays for now, but lost on shutdown/reboot.
- *Accepts a Bundle parameter storing key/value pairs.*
  - Bundle is passed back to activity if it is recreated later.

```
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);      // always call super
    outState.putInt("name", value);
    outState.putString("name", value);
    ...
}
```



onRestoreInstanceStace Method

- *When an activity is recreated later, the event method onRestoreInstanceState is called.*
  - This method can restore any "non-persistent" state of the app.
  - Bundle from onSaveInstanceState from before is passed back in.
- *Accepts a Bundle parameter storing key/value pairs.*
  - Bundle is passed back to activity if it is recreated later.

```
public void onRestoreInstanceState(Bundle inState) {
    super.onRestoreInstanceState(inState);        // always call super
    int name = inState.getInt("name");
    String name = inState.getString("name");
    ...
}
```

Saving Your Own Classes

- *By default, your own classes can't be put into a Bundle.*
- *You can make a class able to be saved by implementing the (methodless) java.io.Serializable interface.*

```
public class Date implements Serializable {
        ...
}

public class MainActivity extends Activity {
     public void onSaveInstanceState(Bundle outState) {
          super.onSaveInstanceState(outState);
          Date d = new Date(2015, 1, 25);
          outState.putSerializable("today", d);
     }
}
```

Preferences

- *SharedPreferences object can store permanent settings and data for your app.*
  - stores key/value pairs similar to a Bundle or Intent
  - pairs added to SharedPreferences persist after shutdown/reboot (unlike savedInstanceState bundles)
- *Two ways to use it:*
  - per-activity (getPreferences)
  - per-app (getSharedPreferences)

SharedPreferences Example

- *Saving preferences for the activity (in onPause, onStop):*

```
SharedPreferences prefs = getPreferences(MODE_PRIVATE);
SharedPreferences.Editor prefsEditor = prefs.edit();
prefsEditor.putInt("name", value);
prefsEditor.putString("name", value);
...
prefsEditor.apply();   // or commit();
```

- *Loading preferences later (e.g. in onCreate):*

```
SharedPreferences prefs = getPreferences(MODE_PRIVATE);
int i = prefs.getInt("name", defaultValue);
String s = prefs.getString("name", "defaultValue");
...
```

SharedPreferences Example

- *You can call getSharedPreferences and supply a file name if you want to have multiple pref. files for the same activity:*

```
~~SharedPreferences prefs = getPreferences(MODE_PRIVATE);~~
SharedPreferences prefs = getSharedPreferences(
        "filename", MODE_PRIVATE);
SharedPreferences.Editor prefsEditor = prefs.edit();
prefsEditor.putInt("name", value);
prefsEditor.putString("name", value);
...
prefsEditor.commit();
```