Deep Dive Into Android
State Restoration
TWITTER
@cyrilmottier

WEBSITE
cyrilmottier.com
capitaine train
The story of a newbie Android developer
Kevin has just developed his first Android app.
He discovers an annoying bug: Fields are cleared on rotate.
3 options
3 options

Don’t care
3 options
Don’t care | Block orientation
3 options
Don’t care | Block orientation | Use configChanges

Hint: all options are
<activity
  android:name=".HomeActivity"
  android:configChanges="orientation">
  <!-- Some sweet IntentFilters. -->
</activity>
Kevin’s satisfied
Still having issues on...
Still having issues on... language changes
<activity>
    android:name=".HomeActivity"
    android:configChanges="orientation|locale">
    <!-- Some annoying IntentFilters. -->
</activity>
Angry Kevin is ANGRY!
<activity>
  android:name=".HomeActivity"
  android:configChanges="orientation|locale|
mcc|mnc|touchscreen|keyboard|
keyboardHidden|navigation|uiMode|
screenLayout|fontScale|screenSize|
osmallestScreenSize">
  <!-- Some fuc**** IntentFilters. Arrggh! -->
</activity>
The nightmare continues...

Still having issues when moving the app to the background
God save the STATE
State restoration

key components
The container
Parcel
The container
Parcel

The content
Primitives types
Primitives arrays
Parcelable
1 parcel.writeInt(1);
2 parcel.writeLong(2L);
3 parcel.writeFloat(3F);
4 parcel.writeString("Hi!");

The content
Primitives types
Primitives arrays
Parcelable
1 parcel.writeIntArray(new int[]{1, 2, 3});
2 parcel.writeLongArray(new long[]{1L, 2L, 3L});
3 parcel.writeDoubleArray(new double[]{1, 2, 3});
4 parcel.writeStringArray(new String[]{
5     "Hi", "Droidcon", "guys!"
6 });
public final class Suggestion implements Parcelable {

    public final String id;
    public final String name;
    public final int type;

    public Suggestion(String id, String name, int type) {
        this.id = Objects.requireNonNull(id);
        this.name = Objects.requireNonNull(name);
        this.type = type;
    }
}
@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(id);
    dest.writeString(name);
    dest.writeInt(type);
}

public static final Parcelable.Creator<Suggestion> CREATOR =
        new Parcelable.Creator<Suggestion>() {
            public Suggestion createFromParcel(Parcel in) {
                return new Suggestion(in.readString(),
                                       in.readString(),
                                       in.readInt());
            }

            public Suggestion[] newArray(int size) {
                return new Suggestion[size];
            }
        };

Parcelable.Creator
The base creator interface

Parcelable.ClassLoaderCreator
A creator with the ClassLoader passed on read.

ParcelableCompat & ParcelableCompatCreatorCallbacks
Compatibility stuff
Bundle

A key-value map & type-safe Parcelable
Parcel internally uses reflection
(required to get the CREATOR instance)

...i.e. beware Proguard
Activity level
state restoration
onCreate(null)
onCreate(null)
onCreate(null)
onSaveInstanceState( )
onCreate(null)

onSaveInstanceState(非空Bundle)

onCreate(非空Bundle)
onCreate(null)
onSaveInstanceState(非空Bundle)
onCreate(非空Bundle)
onRestoreInstanceState(非空Bundle)
onCreate(null)

onSaveInstanceState(null)

onCreate(null)

onRestoreInstanceState(null)
onCreate(null)

onSaveInstanceState(●)

onCreate(●)

onRestoreInstanceState(●)

●: non-null Bundle
What to save?

Non persistent or non reconstructible info
public class SearchActivity extends Activity {

    private static final String STATE_OUTWARD = "state:outward";

    private DateComponents mOutward;

    @Override
    protected void onCreate(Bundle inState) {
        super.onCreate(inState);

        if (inState != null) {
            DateComponents components = inState.getParcelable(STATE_OUTWARD);
            if (components != null) {
                setOutward(components);
            }
        }
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putParcelable(STATE_OUTWARD, mOutward);
    }
}
onSaveInstanceState saves Window
onSaveInstanceState saves
onSaveInstanceState saves

Window | Fragments | Dialogs
Always call the **SUPER METHODS**

Android has no guards on save-related methods
android:stateNotNeeded

For restart-on-crash apps only

(i.e. launcher app)
Developer options

Don’t keep activities
Developer options

Don’t keep activities
Destroy every activity as soon as the user leaves it
View level state restoration
Android saves UI state AUTOMAGICALLY
Android saves UI state AUTOMAGICALLY (aka “It just works!™”)
Android saves UI state AUTOMAGICALLY (aka “It just works!™”) ...except in some cases
Works out-of-the-box if Views

1. Have an ID
2. Are “save” enabled
3. Come from the framework
It always begins with a call to

saveHierarchyState()
RelativeLayout
@id/container

EditText
@id/text

TextView

checkBox
@id/check_box

SparseArray<Parcelable>
RelativeLayout
  @id/container
    EditText
      @id/text
    TextView
    CheckBox
      @id/check_box

SparseArray<Parcelable>
  @id/container
  @id/text
  @id/check_box
Controlling save

```java
setSaveEnabled(boolean)
setSaveFromParentEnabled(boolean)
```
It always ends with a call to restoreHierarchyState()
RelativeLayout
  @id/container
    EditText
      @id/text
    TextView
    CheckBox
      @id/check_box

SparseArray<Parcelable>
  @id/container
  @id/text
  @id/check_box
Ensure your Views’ IDs are unique & constant
static class SavedState extends BaseSavedState {
    int checked;

    SavedState(Parcelable superState) { super(superState); }

    private SavedState(Parcel in) {
        super(in);
        checked = in.readInt();
    }

    @Override public void writeToParcel(Parcel out, int flags) {
        super.writeToParcel(out, flags);
        out.writeInt(checked);
    }

    public static final Parcelable.Creator<SavedState> CREATOR =
        new Parcelable.Creator<SavedState>() {
            public SavedState createFromParcel(Parcel in) {
                return new SavedState(in);
            }

            public SavedState[] newArray(int size) {
                return new SavedState[size];
            }
        };
}
public Parcelable onSaveInstanceState() {
    final Parcelable superState = super.onSaveInstanceState();
    SavedState ss = new SavedState(superState);
    ss.checked = isChecked() ? 1 : 0;
    return ss;
}

@Override
public void onRestoreInstanceState(Parcelable state) {
    SavedState ss = (SavedState) state;
    super.onRestoreInstanceState(ss.getSuperState());
    setChecked(ss.checked == 1);
}
Custom views with children with same IDs
static class SavedState extends BaseSavedState {
  SparseArray childrenStates;

  SavedState(Parcelable superState) { super(superState); }

  private SavedState(Parcel in, ClassLoader loader) {
    super(in);
    childrenStates = in.readSparseArray(loader);
  }

  @Override
  public void writeToParcel(Parcel out, int flags) {
    super.writeToParcel(out, flags);
    out.writeSparseArray(childrenStates);
  }

  public static final Creator<SavedState> CREATOR = ParcelableCompat.
    newCreator(new ParcelableCompatCreatorCallbacks<SavedState>() {
      @Override
      public SavedState createFromParcel(Parcel source, ClassLoader loader) {
        return new SavedState(source, loader);
      }
      @Override
      public SavedState[] newArray(int size) {
        return new SavedState[size];
      }
    });
}
@Override
public Parcelable onSaveInstanceState() {
    final Parcelable superState = super.onSaveInstanceState();
    SavedState ss = new SavedState(superState);
    ss.childrenStates = new SparseArray<Parcelable>();
    for (int i = 0; i < getChildCount(); i++) {
        getChildAt(i).saveHierarchyState(ss.childrenStates);
    }
    return ss;
}

@Override
public void onRestoreInstanceState(Parcelable state) {
    SavedState ss = (SavedState) state;
    super.onRestoreInstanceState(ss.getSuperState());
    for (int i = 0; i < getChildCount(); i++) {
        getChildAt(i).restoreHierarchyState(ss.childrenStates);
    }
}
That has solved nothing!
Still need to block save/restore dispatch
@Override
protected void dispatchSaveInstanceState(SparseArray.ParcelArray container) {
    dispatchFreezeSelfOnly(container);
}

@Override
protected void dispatchRestoreInstanceState(SparseArray.ParcelArray container) {
    dispatchThawSelfOnly(container);
}
Fragment level state restoration
Very similar to Activities state restoration lifecycle.

(Fragments are tied to Activity after all)
Fragment blocks Activity save mechanism

with framework
setSaveFromParentEnabled(false)

with support library
NoSaveStateFrameLayout
2 distinct states

Fragment + View
  common case

View only
  detach, addToBackStack, etc.
Leveraging save/restore

Can to be used to create smooth transitions between your Activities:

- Save the state $S_A$ of A
- Start B with no animations passing $S_A$
- Apply $S_A$ to B
- Transition between A and B was smooth
Summarizing in three rules
Always save the state

An Android app must survive configuration changes & low memory conditions.
Only save essential info

Only save info that is non persistent or can not be reconstructed later.
Use correct levels

Save instance states at the appropriate component level: Activity, Fragment or View.
Thank you!

@cyrilmottier

cyrilmottier.com
Resources

Dressed for Iceland • Cécile Bernard
Moelwynion, Eryri, Cymru • Marc Poppleton
Happy, Confused, Wink, Sad, Angry • Megan Sheehan
Floppy-Disk • Alex Auda Samora

Fonts

Source Sans Pro
Courier