# An adventure into Fragment life-cycles

## Staring: 👻

by Jeroen Tietema

# Meet "Spooky Boy" 👻

» just learned about fragments

» working on his first feature

# Consider the following Activity

```
public class SpookyActivity extends FragmentActivity {

  private SpookyObject spookyObject;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.my_content);

    spookyObject = new SpookyObject();

    if (savedInstanceState == null) {
      Fragment f = new SpookyFragment();
      FragmentTransaction t = getSupportFragmentManager.beginTransaction();
      t.add(R.id.fragment_container, f);
      t.commit();
    }
  }

  public SpookyObject getSpookyObject() { return spookyObject; }
}
```

# SpookyFragment

```
public class SpookyFragment extends Fragment {

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    SpookyObject spookyObject = ((SpookyActivity) getActivity()).getSpookyObject();

    spookyObject.doSomethingSpooky();
  }
}
```

# QA Meet "Squishy Face" 🐙

» he will be QA'ing our Spooky feature

» he has a lot of "arms"

» and in general struggles with phone orientation

# 💥 Crash!

```
public class SpookyFragment extends Fragment {

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    SpookyObject spookyObject = ((SpookyActivity) getActivity()).getSpookyObject();

    spookyObject.doSomethingSpooky(); // <--- NPE
  }
}
```

# FragmentTransaction

```
if (savedInstanceState == null) {
  Fragment f = new SpookyFragment();
  FragmentTransaction t = getSupportFragmentManager
                                .beginTransaction();
  t.add(R.id.fragment_container, f);
  t.commit();
}
```

# FragmentTransaction

```
final class BackStackRecord extends FragmentTransaction
        implements FragmentManager.BackStackEntry, Runnable {
  // ...
  int commitInternal(boolean allowStateLoss) {
    // ...
    mCommitted = true;
    if (mAddToBackStack) {
      mIndex = mManager.allocBackStackIndex(this);
    } else {
      mIndex = -1;
    }
    mManager.enqueueAction(this, allowStateLoss);
    return mIndex;
  }

  // ...
}
```

# FragmentTransaction

- `mManager` == FragmentManager
- `mManger.enqueueAction` will in turn call `mActivity.mHandler.post(backStackRecord)`
- The actual transaction will be executed during the next UI frame

# 😻 Happy path; all good

```
» SpookyActivity#onCreate()
» SpookyActivity#onStart()
» SpookyAcitivty#onResume()
» === render the current frame ===
» SpookyFragment#onCreate()
» SpookyFragment#onStart()
» SpookyFragment#onResume()
```

# Instance re-creation

```
public class FragmentActivity extends Activity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {

    // ...

    if (savedInstanceState != null) {
      Parcelable p = savedInstanceState.getParcelable(FRAGMENTS_TAG);
      mFragments.restoreAllState(p, nc != null ? nc.fragments : null);
    }
    mFragments.dispatchCreate(); }
  }
}
```

# 🙀 Unhappy path

```
» SpookyActivity#onCreate()
  » SpookyFragment#onCreate()
» SpookyActivity#onStart()
  » SpookyFragment#onStart()
» SpookyActivity#onResume()
  » SpookyFragment#onResume()
```

# How to fix our Spooky App? 

# Fix 1/2

## Create spookyObject earlier:

```
public class SpookyActivity extends FragmentActivity {

  private SpookyObject spookyObject;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    spookyObject = new SpookyObject(); // <-- create spooky object earlier

    super.onCreate(savedInstanceState);
    setContentView(R.layout.my_content);

    // ...
  }
}
```

# Fix 2/2

Use `onActivityCreated(Bundle)`

```
public class SpookyFragment extends Fragment {
  @Override
  public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);

    SpookyObject spookyObject = ((SpookyActivity) getActivity()).getSpookyObject();

    spookyObject.doSomethingSpooky();
  }
}
```

# Recap so far

- The life-cycles of Fragments and Activities are not always called in the same order with respect to each other.
- Make sure you test Acitivity recreation (rotation is the easiest way).
- Pro-Tip: download the Android sources in Android Studio (Find usages kind of works now a days with AS 2.0 Preview 9)

# Fragments in layouts? 👻

```
public class FragmentActivity extends Activity {
    @Override
    public View onCreateView(String name, @NonNull Context context, @NonNull AttributeSet attrs) {
        if (!"fragment".equals(name)) {
            return super.onCreateView(name, context, attrs);
        }
        final View v = mFragments.onCreateView(name, context, attrs);
        if (v == null) {
            return super.onCreateView(name, context, attrs);
        }
        return v;
    }
}
```

# Fragments in layouts

- triggered by `setContentView()`
- `FragmentActivity#onCreateView()` delegates to `FragmentManagerImpl#onCreateView()`
- `FragmentManagerImpl` instanciates the fragment class and moves it to the `Fragment.CREATED` state

# FragmentMngrImpl#moveState()

```
if (f.mFromLayout) {
  // For fragments that are part of the content view
  // layout, we need to instantiate the view immediately
  // and the inflater will take care of adding it.
  f.mView = f.performCreateView(f.getLayoutInflater(
              f.mSavedFragmentState), null, f.mSavedFragmentState);
  if (f.mView != null) {
    f.mInnerView = f.mView;
    if (Build.VERSION.SDK_INT >= 11) {
      ViewCompat.setSaveFromParentEnabled(f.mView, false);
    } else {
      f.mView = NoSaveStateFrameLayout.wrap(f.mView);
    }
    if (f.mHidden) f.mView.setVisibility(View.GONE);
    f.onViewCreated(f.mView, f.mSavedFragmentState);
  } else {
    f.mInnerView = null;
  }
}
```

# Recap before 🍺

- the life-cycles are more similar during recreation when using fragments in layouts
- `FragmentManagerImpl#moveState()` is a really intresting method to read if you use fragments

# Questions?