ROBOGUICE

Bringing Best Practices to Android
One App at a Time

ANDROID HAS MUCH TO LIKE

- Large Userbase
- Active Development
- Open Source

BUT ANDROID IS IN YOUR FACE

- It is nearly impossible to write a class that isn’t instantly recognizable as an Android class.

ANDROID BOILERPLATE = BUGS

The less code you write, the fewer bugs

HOW DOES ROBOGUICE HELP?

INJECTION

- Widely used
- Robust principles
- Now coming to other languages

INJECTION EXAMPLE

- First there were logging objects
- Then there were static factories
- Now: injection

INJECTVIEW

-
  ```
  class MyActivity extends RoboActivity {
      @InjectView(R.id.text2) TextView nameView;
      ...
  }
  ```

INJECTVIEW

- TypeSafe in that it fails fast
- Removes boilerplate from onCreate

INJECT RESOURCE, EXTRAS, PREFERENCES, ETC.

- *<string name=”hello”>Hello, there</string>*
- @InjectResource(R.string.hello) String hello;

SYSTEM SERVICES

- @Inject LocationManager locationManager;
  - *LocationManager locationManager = (LocationManager) context.getSystemService(Context.LOCATION_MANAGER)*
- *@Inject SharedPreferences prefs;*

DEMO

BEYOND THE BASICS

CUSTOM BINDINGS

- Make a Guice Module
- Set up your custom bindings
  - bind(MyInterface.class).to(MyImplementation.class)
  - bind(MyClass.class).toProvider(MyFancyFactoryForMyClass.class)
- Override bindings for testing
- Simple, but very powerful

BACKGROUND TASKS

- SafeAsyncTask is a more robust AsyncTask
- adds onException(), onFinally()

EVENTS

- void killBackgroundThread(@Observes OnDestroyEvent e) {
      thread.cancel();
  }

TESTING

- Easy to swap out bindings with mock objects
- Works with Robolectric, mockito, and other test frameworks

SUMMARY

- Production-ready and widely used
- Simplifies your code
- Makes development fun

QUESTIONS?

roboguice.org