Android Development Simplified

An Introduction to AndroidAnnotations

Jamie Ridgway
Solution Architect - HP
Twitter : @jamie_ridgway
email : jamie.ridgway@hp.com

Topics

- Annotations 101
- Intro to AndroidAnnotations
- A Quick Look at Android Annotations
- Demo
- Deeper Dive
- Questions?

Annotations 101

- a form of metadata, providing data about a program that is not part of the program itself
- can be used to generate code
- some are available at runtime
- @annotation

AndroidAnnotations

- Open Source Library started by Pierre-Yves Ricau
- Built on Java annotation mechanism
- Includes dependency injection
- Eliminates much of the “boilerplate” and “glue” code found in a typical Android app
- Performed as an additional compilation step
- …so, there is NO performance runtime

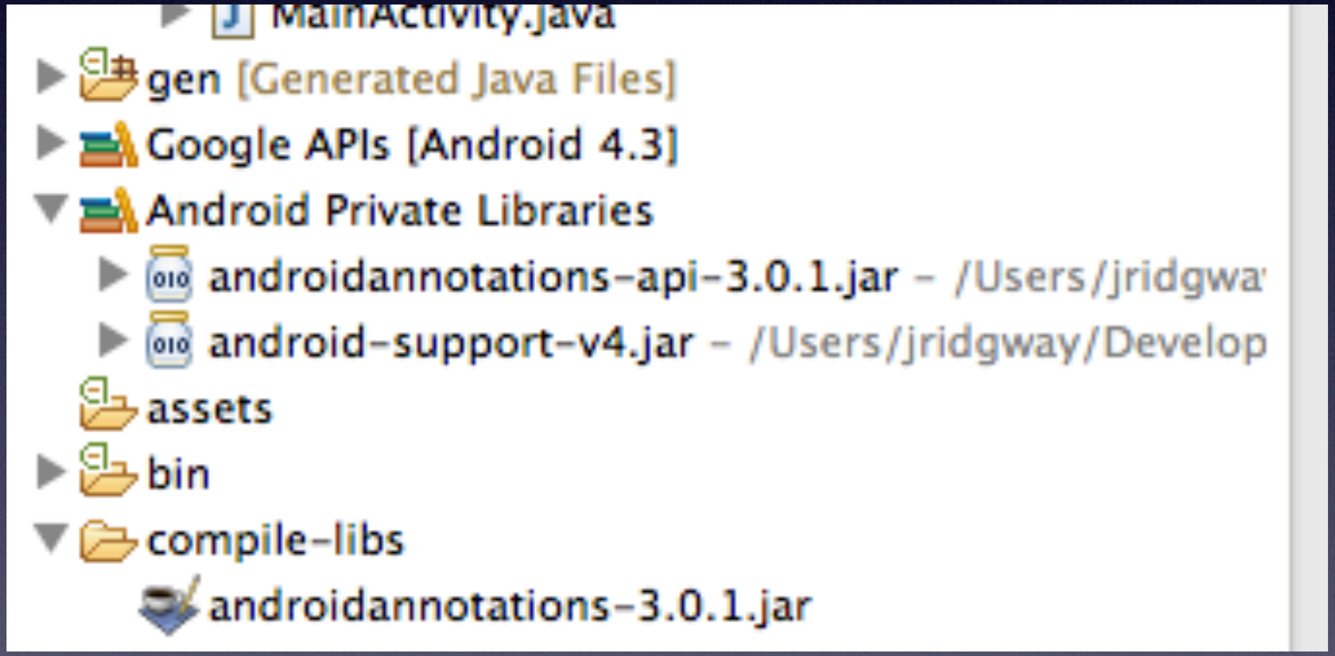
AndroidAnnotations

- Most current release is 3.0.1
- 80+ annotations
- Typically smaller, less complex code
- Enhances readability

Configuring AndroidAnnotations

- Add API jar to the libraries on the class path

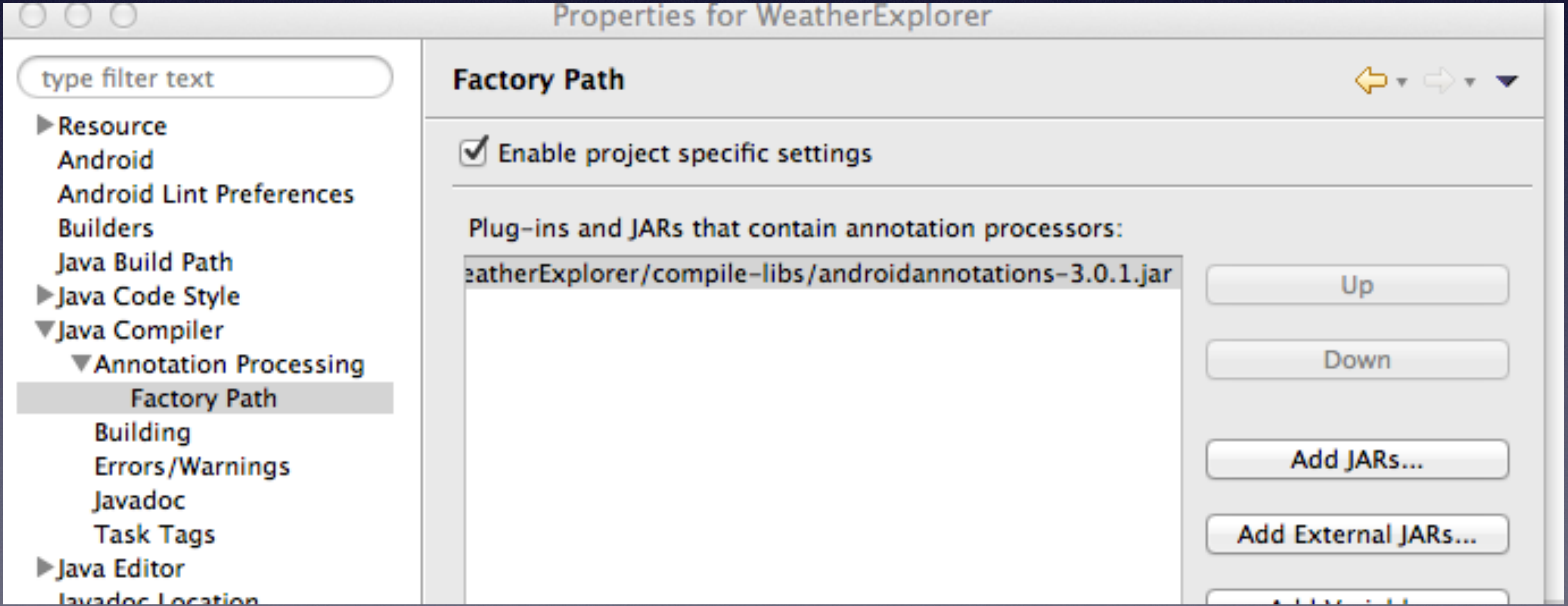
- Non API jar copied to a folder that's NOT on the class path. **This is important!**



```
    ▶ MainActivity.java
▶ gen [Generated Java Files]
▶ Google APIs [Android 4.3]
▼ Android Private Libraries
    ▶ androidannotations-api-3.0.1.jar - /Users/jridgwa
    ▶ android-support-v4.jar - /Users/jridgway/Develop
  assets
▶ bin
▼ compile-libs
    androidannotations-3.0.1.jar
```

Configuring AndroidAnnotations

- For eclipse, add non-API jar to the libraries on the Annotation Processing Factory path



Properties for WeatherExplorer

Factory Path

☑ Enable project specific settings

Plug-ins and JARs that contain annotation processors:

eatherExplorer/compile-libs/androidannotations-3.0.1.jar

A Quick Look At Some Annotations

Enhanced Components

```
@EActivity(R.layout.main)
public class MyActivity extends Activity {

}
```

```
@EActivity
public class MyListActivity extends ListActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

}
```

@EActivity

```
@EActivity(R.layout.main)
public class MainActivity extends Activity {

}
```

You MUST rename the Activity in the manifest.

```
 <activity
    android:name=“com.knoxmobilitylab.weatherexplorer.MainActivity"


  …..


<activity
    android:name=“com.knoxmobilitylab.weatherexplorer.MainActivity_"
```

String Resources

```
@StringRes(R.string.credits)
String credits;
```

```
<string name=“credits">A String of Text</string>
```

Demo

Deeper Dive

Enhanced Components

- @EActivity
- @EApplication
- @EBean
- @EFragment
- @EProvider
- @EReceiver
- @EIntentService
- @EService
- @EView
- @EViewGroup

@EFragment

```
@EFragment
public class MyFragment extends Fragment {

}
```

Must Rename Fragment in manifest.

```
<fragment
   android:id="@+id/myFragment"
   android:name="com.company.MyFragment_"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent" />
```

@EBean

```
@EBean
public class MyClass {

}
```

@EBean allows non-Android classes to use other AndroidAnnotations.

However, there are specific constructor rules.

Injection

- @AfterInject
- @AfterViews
- @App
- @Bean
- @Extra
- @FragmentArg
- @FragmentById
- @FragmentByTag
- @FromHtml
- @HttpsClient
- @NonConfigurationInstance
- @RootContext
- @SystemService
- @ViewById

@Bean

```
@EActivity
public class MyActivity extends Activity {

    /* A MyImplementation instance will be injected.
     * MyImplementation must be annotated with @EBean
       and implement MyInterface.
     */
    @Bean(MyImplementation.class)
    MyInterface myInterface;
```

@ViewById

```
@ViewById
TextView textViewTemp;

@ViewById(R.id.textViewLocation)
TextView textViewLocationValue;
```

@AfterViews

```
 @ViewById
TextView myTextView;

@AfterViews
void updateTextWithDate() {
    myTextView.setText("Date: " + new Date());
}
```

- Called after all views are created and view bindings are complete.
- Can have multiple @AfterViews.
- Do Not use views in OnCreate()

Event Binding

- @TextChange
- @AfterTextChange
- @BeforeTextChange
- @FocusChange
- @CheckedChange
- @Touch
- @Click
- @LongClick
- @ItemClick
- @ItemLongClick
- @ItemSelect
- @OptionsItem
- @SeekBarProgressChange
- @SeekBarTouchStart
- @SeekBarTouchStop

@Click

```
@Click(R.id.myButton)
void myButtonWasClicked() {
    [...]
}
@Click
void anotherButton() {
    [...]
}
@Click
void yetAnotherButton(View clickedView) {
    [...]
```

Threading

The Android UI toolkit is not thread-safe. So, you must not manipulate your UI from a worker thread—you must do all manipulation to your user interface from the UI thread. Thus, there are simply two rules to Android's single thread model:

1. Do not block the UI thread
2. Do not access the Android UI toolkit from outside the UI thread

Source : http://developer.android.com/guide/components/processes-and-threads.html

Threading

- @Background
- @UiThread

@Background

```
@Background
void someBackgroundWork(String aParam, long anotherParam) {
    [...]
}
@Background(id="cancellable_task")
void someCancellableBackground(String aParam,…) {
    [...]
}
@Background(delay=2000)
void doInBackgroundAfterTwoSeconds() {
}
```

@Background indicates the method is to be run in a thread other than the UIThread.

@UiThread

```
@UiThread
void doInUiThread(String aParam, long anotherParam) {
    [...]
}

@UiThread(delay=2000)
void doInUiThreadAfterTwoSeconds() {
}
```

Ensures the method WILL be run on the UIThread.

Together with @Background mean no more ASyncTask!

REST API

- @Rest
- @RestService
- @Get
- @Post
- @Put
- @Delete
- @Options
- @Head
- @Accept

@Rest, @Get, @Post, @Delete

```
@Rest(rootUrl = "http://company.com/ajax/services", converters =
{ MappingJacksonHttpMessageConverter.class })
public interface MyRestClient {

  @Get("/events/{year}/{location}")
  EventList getEventsByYearAndLocation(int year, String location);

  @Post("/events/{id}")
  void addEventById(Event event, long id);

  @Put("/events")
  void updateEvent(Event event);

  @Delete("/events/{id}")
  Event deleteEventById(long id);

}
```

The REST client is built on Spring REST Template*.

*Some assembly is required.

Resource Injection

- @StringRes
- @AnimationRes
- @ColorRes
- @DimensionPixelOffsetRes
- @DimensionPixelSizeRes
- @DimensionRes
- @BooleanRes
- @ColorStateListRes
- @DrawableRes
- @IntArrayRes
- @IntegerRes
- @LayoutRes
- @MovieRes
- @StringArrayRes
- @TextArrayRes
- @TextRes
- @HtmlRes

Typesafe SharedPreferences

- @DefaultBoolean
- @DefaultFloat
- @DefaultInt
- @DefaultLong
- @DefaultString
- @DefaultRes
- @Pref
- @SharedPref

Miscellaneous

- @InstanceState
- @WindowFeature
- @Fullscreen
- @NoTitle
- @CustomTitle
- @OptionsMenu
- @OrmLiteDao
- @RoboGuice
- @Trace
- @Transactional
- @OnActivityResult
- @HierarchyViewerSupport
- @ServiceAction

Questions?

Resources

- Code & Slides are available on GitHub
  https://github.com/ridgwayj/codepalousa2014.git

- AndroidAnnotations site
  https://github.com/excilys/androidannotations