

# Android: annotations to the rescue

---

Romain Rochegude

2016.03.30

# Introduction

---

# Introduction

- Huge gain of productivity
  - Don't (re)write boilerplate code
  - Code generation
- Code quality improvements
  - Widely tested and documented third-party libraries
  - Less code to write, less bugs to solve
- What about performances?
  - Annotations processing compile time vs. runtime

# Android compliance

---

# Android compliance

- android-apt plugin for Android Studio
  - Allows developers to configure a compile time annotation processor as a dependency in the build.gradle file
  - Runs annotation processing
  - Example:

```
dependencies {  
    compile 'a.group:annotation:x.x.x'  
    apt 'a.group:processor:x.x.x'  
}
```

# Views

---

# Butter Knife

- No `findViewById` anymore, elegant binding mechanism
- Still want to use the `findViewById`? Just try the autocast alternative
- Simple way to bind resources
- Bind anything: activities, fragments, views, view holders
- The power of view lists (actions, setters)

- Event bindings:
  - OnClick/OnLongClick, OnItemSelected, OnCheckedChanged, OnEditorAction, OnFocusChange, OnItemLongClick, OnPageChange, OnTextChanged, OnTouch



- Under the hood
  - Same package binding class generation
  - A single entry point: the ButterKnife class, that resolves the concrete binder

```

@FragmentWithArgs
public class FragmentRepoDetail extends Fragment {

    @Bind(R.id.FragmentRepoDetail_TextView_Description)
    TextView mTextViewDescription;
    @Bind(R.id.FragmentRepoDetail_TextView_Url)
    TextView mTextViewUrl;
    @Bind(R.id.FragmentRepoDetail_TextView_Empty)
    TextView mTextViewEmpty;
    @Bind(R.id.FragmentRepoDetail_TextView_Error)
    TextView mTextViewError;
    @Bind(R.id.FragmentRepoDetail_ProgressBar_Loading)
    ProgressBar mProgressBarLoading;
    @Bind(R.id.FragmentRepoDetail_ContentView)
    LinearLayout mContentView;

    @Override
    public void onViewCreated(final View poView, final Bundle poSavedInstanceState) {
        super.onViewCreated(poView, poSavedInstanceState);

        ButterKnife.bind(this, poView);
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        ButterKnife.unbind(this);
    }
}

```

**Figure 1:** FragmentRepoDetail\_Butterknife.java

```

public class FragmentRepoDetail$$ViewBinder<T extends fr.guddy.androidstarteralt.mvp.repoDetail.FragmentRepoDetail> implements ViewBinder<T> {
    @Override public void bind(final Finder finder, final T target, Object source) {
        View view;
        view = finder.findRequiredView(source, 2131558574, "field 'mTextViewDescription'");
        target.mTextViewDescription = finder.castView(view, 2131558574, "field 'mTextViewDescription'");
        view = finder.findRequiredView(source, 2131558575, "field 'mTextViewUrl'");
        target.mTextViewUrl = finder.castView(view, 2131558575, "field 'mTextViewUrl'");
        view = finder.findRequiredView(source, 2131558570, "field 'mTextViewEmpty'");
        target.mTextViewEmpty = finder.castView(view, 2131558570, "field 'mTextViewEmpty'");
        view = finder.findRequiredView(source, 2131558571, "field 'mTextViewError'");
        target.mTextViewError = finder.castView(view, 2131558571, "field 'mTextViewError'");
        view = finder.findRequiredView(source, 2131558572, "field 'mProgressBarLoading'");
        target.mProgressBarLoading = finder.castView(view, 2131558572, "field 'mProgressBarLoading'");
        view = finder.findRequiredView(source, 2131558573, "field 'mContentView'");
        target.mContentView = finder.castView(view, 2131558573, "field 'mContentView'");
    }

    @Override public void unbind(T target) {
        target.mTextViewDescription = null;
        target.mTextViewUrl = null;
        target.mTextViewEmpty = null;
        target.mTextViewError = null;
        target.mProgressBarLoading = null;
        target.mContentView = null;
    }
}

```

Figure 2: FragmentRepoDetail\$\$ViewBinder.java

# Navigation

---

# IntentBuilder and FragmentArgs

- Boilerplate code to declare a new screen
- Annotations to declare an Activity/Fragment
- Annotations to declare (optional) parameter(s) to pass
- Class generation following the Builder pattern
- Method to inject parameter(s) in the target class

```

@IntentBuilder
public class ActivityRepoDetail extends Activity {

    @Extra
    Long mItemId;

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

        ActivityRepoDetailIntentBuilder.inject(getIntent(), this);
        // ...
    }
}

```

**Figure 3:** ActivityRepoDetail.java

```

public final class ActivityRepoDetailIntentBuilder {
    private final Long mItemId;

    public ActivityRepoDetailIntentBuilder(Long mItemId) {
        this.mItemId = mItemId;
    }

    public Intent build(Context context) {
        Intent intent = new Intent(context, ActivityRepoDetail.class);
        intent.putExtra("mItemId", mItemId);
        return intent;
    }

    public static void inject(Intent intent, ActivityRepoDetail activity) {
        Bundle extras = intent.getExtras();
        if (extras.containsKey("mItemId")) {
            activity.mItemId = (Long) extras.get("mItemId");
        } else {
            activity.mItemId = null;
        }
    }

    public static Long getMItemId(Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras.containsKey("mItemId")) {
            return (Long) extras.get("mItemId");
        } else {
            return null;
        }
    }
}

```

**Figure 4:** ActivityRepoDetailIntentBuilder.java

```
@FragmentWithArgs
public class FragmentRepoDetail extends Fragment {

    @Arg
    Long mItemId;

    public FragmentRepoDetail() {
    }

    @Override
    public void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        FragmentArgs.inject(this);
    }
}
```

**Figure 5:** FragmentRepoDetail\_FragmentArgs.java



```

public final class FragmentRepoDetailBuilder {
    private final Bundle mArguments = new Bundle();

    public FragmentRepoDetailBuilder(@NonNull Long itemId) {
        mArguments.putLong("itemId", itemId);
    }

    @NonNull
    public static FragmentRepoDetail newFragmentRepoDetail(@NonNull Long itemId) {
        return new FragmentRepoDetailBuilder(itemId).build();
    }

    public static final void injectArguments(@NonNull FragmentRepoDetail fragment) {
        Bundle args = fragment.getArguments();
        if (args == null) {
            throw new IllegalStateException("No arguments set. Have you setup this Fragment with the corresponding FragmentArgs Builder? ");
        }
        if (!args.containsKey("itemId")) {
            throw new IllegalStateException("required argument itemId is not set");
        }
        fragment.mItemId = args.getLong("itemId");
    }

    @NonNull
    public FragmentRepoDetail build() {
        FragmentRepoDetail fragment = new FragmentRepoDetail();
        fragment.setArguments(mArguments);
        return fragment;
    }

    @NonNull
    public <F extends FragmentRepoDetail> F build(@NonNull F fragment) {
        fragment.setArguments(mArguments);
        return fragment;
    }
}

```

**Figure 6:** FragmentRepoDetailBuilder.java

# Interactions

---

# AutoValue: Cursor Extension

- Abstract class with the `@AutoValue` to define POJO
- Simple `@ColumnName` to define the binding between column names and class fields

@AutoValue

```
public abstract class Person {  
    @ColumnName("name") abstract String name();  
  
    public static Person create(Cursor cursor) {  
        return AutoValue_Person.  
            createFromCursor(cursor);  
    }  
  
    abstract ContentValues toContentValues();  
}
```

```

abstract class $AutoValue_Person extends Person {
    private final String name;
    private final String surname;
    private final int age;

    $AutoValue_Person(
        String name,
        String surname,
        int age) {
        if (name == null) {
            throw new NullPointerException("Null name");
        }
        this.name = name;
        if (surname == null) {
            throw new NullPointerException("Null surname");
        }
        this.surname = surname;
        this.age = age;
    }

    @ColumnName(value = "name")
    @Override
    String name() {
        return name;
    }

    @ColumnName(value = "surname")
    @Override
    String surname() {
        return surname;
    }

    @ColumnName(value = "age")
    @Override
    int age() {
        return age;
    }

    @Override
    public String toString() {
        return "Person{"
            + "name=" + name + ", "
            + "surname=" + surname + ", "
            + "age=" + age
            + "}";
    }

    @Override
    public boolean equals(Object o) {
        if (o == this) {
            return true;
        }
        if (o instanceof Person) {
            Person that = (Person) o;
            return this.name.equals(that.name())
                && this.surname.equals(that.surname())
                && this.age == that.age();
        }
        return false;
    }

    @Override
    public int hashCode() {
        int h = 1;
        h ^= 1000003;
        h ^= this.name.hashCode();
        h ^= 1000003;
        h ^= this.surname.hashCode();
        h ^= 1000003;
        h ^= this.age;
        return h;
    }
}

```

Figure 7: \$AutoValue\_Person.java

```

final class AutoValue_Person extends $AutoValue_Person {
    AutoValue_Person(String name, String surname, int age) {
        super(name, surname, age);
    }

    static AutoValue_Person createFromCursor(Cursor cursor) {
        String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));
        String surname = cursor.getString(cursor.getColumnIndexOrThrow("surname"));
        int age = cursor.getInt(cursor.getColumnIndexOrThrow("age"));
        return new AutoValue_Person(name, surname, age);
    }

    public ContentValues toContentValues() {
        ContentValues values = new ContentValues(3);
        values.put("name", name());
        values.put("surname", surname());
        values.put("age", age());
        return values;
    }
}

```

Figure 8: AutoValue\_Person.java

# Others

---

# Others

- ORM: requery
- JSON parsing: LoganSquare
- Bus: EventBus
- Saving and restoring instance state: Icepick
- Easily deal with the result from an activity started for result: onActivityResult
- and so on:  
<http://android-arsenal.com/tag/166>



# Write custom annotation processors

---

# Concepts

- Provide a robust annotation API
- Implement the algorithm to search for your annotations and deal with it

# Generate Java source files: JavaPoet

- Powerful and complete API to describe
  - static imports,
  - classes, interfaces, enums, anonymous inner classes,
  - fields, parameters, variables,
  - methods, constructors,
  - annotations, javadoc
- Specific wildcards to format the output code
- Test generated files with Google's *Compile Testing* and *Truth*

```
MethodSpec main = MethodSpec
    .methodBuilder("main")
    .addModifiers(Modifier.PUBLIC,
                  Modifier.STATIC)
    .returns(void.class)
    .addParameter(String[].class, "args")
    .addStatement("$T.out.println($S)",
                  System.class,
                  "Hello, JavaPoet!")
    .build();
```

to

```
public static void main(String[] args) {  
    System.out.println("Hello, JavaPoet!");  
}
```

# Google AutoService

- Simple `@AutoService` annotation for your `javax.annotation.processing.Processor` subclass
- Automatic generation of the `META-INF/services/javax.annotation.processing.Processor` in the output classes folder
- Automatic inclusion of the annotated processor

**A must-read article: ANNOTATION  
PROCESSING 101 by Hannes Dorfmann**

# Conclusion

---



# Conclusion

- Performance: machine and/or human
- Readable
- Maintainable