

# *Annotation Processing With Android I*

Andrew Grosner

[@agrosner/github.com/agrosner](https://github.com/agrosner)

Android Developer

***Raizlabs***

# *Breakdown*

- History of Annotation Processing
- The meaning of boilerplate code
- What is **ANNOTATION PROCESSING?** and How does it work?
- High-level Concepts
- Existing Android Libraries

# *History of Annotation Processing*

# *Annotations*

@Override

@Bind

@Table(databaseName = MyDatabase.NAME)

# *A Bit Of History*

- Java 5 -> 9/30/2004
  - Annotations
- ~6 months later
  - **apt, doclet** were standalone tools
- Java 6 -> 10/11/2006
  - JSR 269 integrated the tool into **javac**
- “Mirror API”

# *History of A.P. In Android*

- Past ~ 1-2 years
- Butterknife, Dagger2, Schematic, Android Annotations, DBFlow, Icepick, etc
- More and more libraries
- Exciting!

*BOILERPLATE*

*BOILERPLATE*

*BOILERPLATE*

*BOILERPLATE*

# Boilerplate

```
public void handleResponse(String responseString) {  
  
    User user = new User();  
    JSONObject json = new JSONObject(responseString);  
    user.firstName = json.getString("firstName");  
    user.lastName = json.getString("lastName");  
    user.token = json.getString("token");  
    // etc..  
}
```



# *Boilerplate*

- “...sections of code that have to be included in many places with little or no alteration. It is often used...[in] languages that are considered verbose...the programmer must write a lot of code to do minimal jobs.” - Wikipedia
- Repetitive, Repetitive, Repetitive, Repetitive, Repetitive
- Error prone
- Messy


*What is Annotation Processing?*

# *What is Annotation Processing?*

- Pre-compile time annotation evaluation used to generate code, files, or other useful pieces of information

```
@Builder
public class Player {
    String name;
    boolean isActive;
}

public class PlayerBuilder {
    public PlayerBuilder name(String name) {...}
    public PlayerBuilder isActive(boolean isActive) {...}
    public Player build(){...};
}
```

A horizontal arrow points from the Player class code on the left to the PlayerBuilder class code on the right, indicating the transformation performed by annotation processing.

*How It Actually Works*

# *How it Actually Works*

- **ServiceLoader** discovers **Processor** instances in the build path
- **ProcessingEnvironment** is initialized
- **Processor** classes are initialized with the **ProcessingEnvironment**
- Annotation Processing begins, processors write to the **File**
- Processing executes in rounds, until last round completed.

# *How it Actually Works*

- Runs in a separate JVM :)
- Compiled Sources + Dependencies are not available :(
- But since it runs isolated from your app, use whatever dependencies you want!
  - AKA Guava

*“Ill just use reflection”*

## *Cons*

SLOW (~2x)

Bugs @ runtime

Access-level violation

Tracking down errors

## **Pros**

Less difficult

Concise code

In same project



*“You are writing code that writes code that  
must then compile correctly”*

## *Pros*

FAST

No rt. performance hit

Code is fully debuggable

Will fail on compile ~95%

## Cons

Separate API

Separate Gradle Plugin

Tricky to write

Must be accessible


# *High-level Concepts*

# Processor

```
public class MyProcessor extends AbstractProcessor {  
  
    Set<String> getSupportedAnnotationTypes();  
  
    SourceVersion getSupportedSourceVersion();  
  
    void init(ProcessingEnvironment env);  
  
    boolean process(Set<? extends TypeElement> elements,  
        RoundEnvironment roundEnv);  
}
```

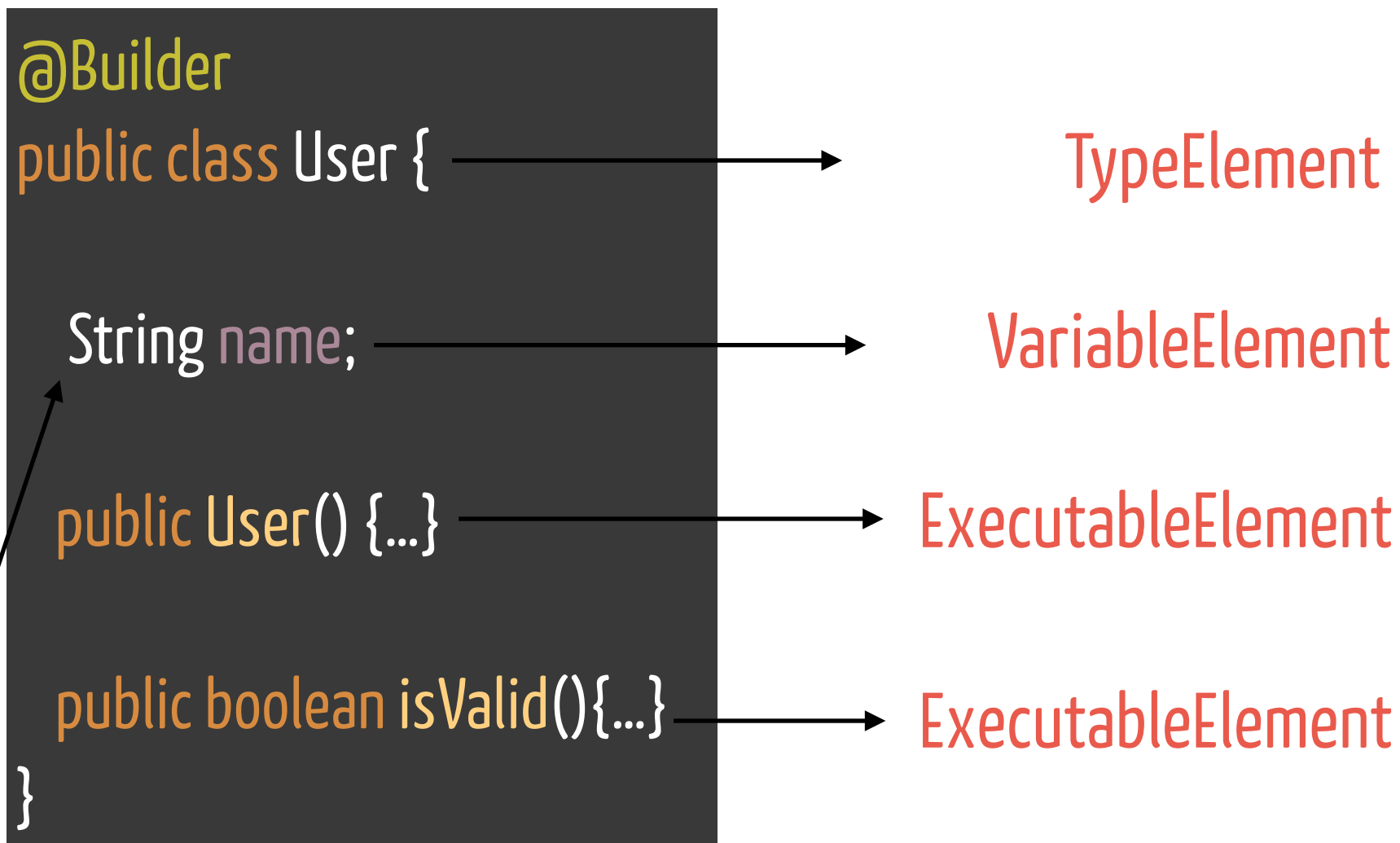
# Processor

```
public class MyProcessor extends AbstractProcessor {  
  
    @Override  
    public boolean process(Set<? extends TypeElement> elements,  
        RoundEnvironment roundEnv) {  
        Set<? extends Element> elements =  
            roundEnv.getElementsAnnotatedWith(SomeAnnotation.class);  
  
        for (Element element: elements) {  
            // do something here  
        }  
    }  
}
```



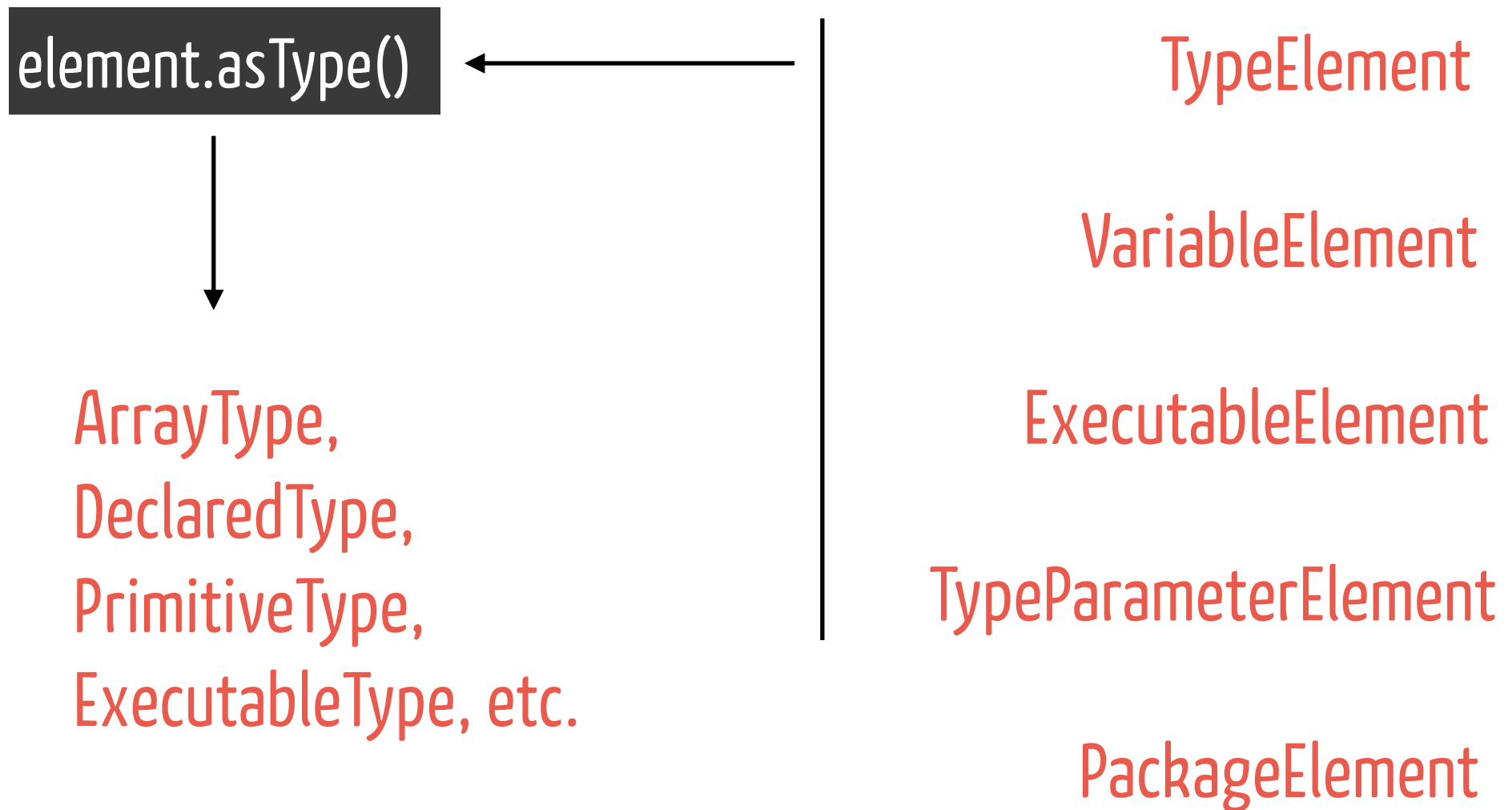
What's an  
Element?

# Elements



Where can I get what type it is?

# TypeMirror



*Existing Libraries*



# *ButterKnife*

- View Injection Library
  - `TextView view = (TextView) findViewById(R.id.myView)`
- Eliminates anonymous inner classes for listeners
  - `view.setOnClickListener(this)`
- Jake Wharton (<https://github.com/JakeWharton/butterknife>)

# ButterKnife

```
public HomeFragment extends Fragment {  
  
    TextView name;  
    TextView email;  
  
    String warningMessage;  
  
    @Override  
    public void onCreateView(View view, Bundle  
savedInstanceState) {  
        ...  
        name = (TextView) view.findViewById(R.id.name);  
        email = (TextView) view.findViewById(R.id.email);  
  
        warningMessage = view.getContext()  
            .getString(R.string.warning_message);  
    }  
}
```

```
public HomeFragment extends Fragment {  
  
    @Bind(R.id.name) TextView name;  
    @Bind(R.id.email) TextView email;  
  
    @BindString(R.string.warning_message)  
    String warningMessage;  
  
    @Override  
    public void onCreateView(View view, Bundle  
savedInstanceState) {  
        ...  
        ButterKnife.bind(this);  
    }  
}
```

# DBFlow

- Database SQLite ORM library
- Generates all of the database interaction, setup, and more
  - Maps classes to tables using **@Table**
- Many features
  - Model-Caching, Triggers, Views, Indexes, Multi-Database support, and more
- SQLite-query wrapping
- Andrew Grosner, Raizlabs (<https://github.com/Raizlabs/DBFlow>)

# DBFlow

```
@Database(name = MLBDatabase.NAME, version = MLBDatabase.VERSION)
public class MLBDatabase {

    public static final String NAME = "MLBDatabase";

    public static final String VERSION = 1;
}
```

# DBFlow

```
@Table(databaseName = MLBDatabase.NAME)
public class Player extends BaseModel {

    @Column
    @PrimaryKey(autoincrement = true)
    long id;

    @Column
    String name;

    @Column
    Date birthdate;

    @Column
    @ForeignKey(references =
        @ForeignKeyReference(columnName = "team_id",
            columnType = long.class,
            foreignColumnName = "id"))
    Team team;
}
```

# *Android Annotations*

- Goal is to speed up android app development -> **@EActivity**
- Dependency injection -> **@ViewById**
- Simplified threading model -> **@Background, @UiThread**
- Event Binding -> **@Click**
- REST client
- Excilys (<https://github.com/excilys/androidannotations>)

# Android Annotations

```
@EActivity(R.layout.translate) // Sets content view to R.layout.translate
public class TranslateActivity extends Activity {

    @ViewById // Injects R.id.textInput
    EditText textInput;

    @ViewById(R.id.myTextView) // Injects R.id.myTextView
    TextView result;

    @Click // When R.id.doTranslate button is clicked
    void doTranslate() {
        translateInBackground(textInput.getText().toString());
    }
}
```

# Android Annotations

```
@EActivity(R.layout.translate) // Sets content view to R.layout.translate
public class TranslateActivity extends Activity {
```

```
    @Background // Executed in a background thread
```

```
    void translateInBackground(String textToTranslate) {
```

```
        String translatedText = callGoogleTranslate(textToTranslate);
        showResult(translatedText);
```

```
    }
```

```
    @UiThread // Executed in the ui thread
```

```
    void showResult(String translatedText) {
```

```
        result.setText(translatedText);
        result.startAnimation(fadeIn);
```

```
    }
```

```
}
```



# *Parser*

- JSON ORM for serialization/deserialization
- Custom conversions support
- Merging data from different sources
- Andrew Grosner, Raizlabs (<https://github.com/raizlabs/Parser>)

# Parser

```
[
{
  name: "New York Yankees",
  coach: "Joe Girardi",
  players: [
    {
      name: "Mark Teixeira",
      position: "First Base"
    }
  ]
}
]
```

```
@Translatable
public class Team {

    @Key
    String name;

    @Key
    String coach;

    @Key(name = "players")
    List<Player> playersList;
}
```

```
@Translatable
public class Player {

    @Key
    String name;

    @Key
    String position;
}
```

*Questions?*

# Feedback

Please take a moment to fill out the class feedback form via the app. Paper feedback forms are also available in the back of the room.

[eventmobi.com/adcboston](http://eventmobi.com/adcboston)



***Raizlabs***



*Thank You.*

IF YOU LIKED THIS, FOLLOW ME @AGROSNER  
STAR ME ON [GITHUB.COM/AGROSNER](https://github.com/agrosner)

