ToiletPaper #84

Android Data Binding

Author: Tobias Schröpf / Senior Software Architect / Business Division Automotive World

✕ Problem

In many Android apps, the UI code calls findViewById to find views from the layout of a screen. In a next step, these views are downscaled into the needed type (e.g. TextView) and finally, views are called to manipulate them (e.g. setText).
This (unnecessary) "glue code", which binds UI to the models, is usually located in the largely complex activities and fragments.

✓ Solution

With the help of Google's Android Data Binding Library, this boilerplate code can be eliminated. Model classes (e.g. POJOs) may be referenced directly from the XML layout, to set e.g. Drawables and StringResources directly in the views. Using ObservableFields or BaseObservables, the layout can even **bind** to the properties of the models and apply changes automatically.

➔ Example

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Wrap your Layout into <layout></layout> tags --> <layout
[...]>

    <!-- define variables and imports you want to use -->
    <data>
        <variable name="model" type="de.schroepf.databinding.model.User" />
    </data>

    <LinearLayout [...]>

        <!-- Surround any data binding expression with @{...} -->
        <ImageView [...] app:imageResource='@{model.photo}' />
        <TextView [...] android:text='@{model.firstName + " " + model.lastName}' />
    </LinearLayout>
</layout>
```

```java
// The fields of this model class can be accessed in the layout.xml file
// Extend BaseObservable or use ObservableFields to allow the layout to respond to changes.
public class User {
    public final String firstName;
    public final String lastName;
    @DrawableRes
    public final int photo;

    public User(String firstName, String lastName, @DrawableRes int photo) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.photo = photo;
    }
}
```

```java
public class MainActivity extends Activity {
    // represents the state we want to display in this activity
    private User myModel = new User("Coffee", "Toffee", R.drawable.coffee_toffee);

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

        // Make sure to use DataBindingUtil to setContentView/inflate your view and retrieve the binding
        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);

        // use the binding to access the layout's view hierarchy (typesafe!) and set variables
        binding.setModel(myModel);
    }
}
```

✚ Further Aspects

- With the help of generated binding classes, views in the layouts can be accessed programmatic, type-safe (no downscaling!) and without findViewById
- "Method References" and "Listener Bindings" also allow binding of UI events to Java code directly in the XML layout
- Google documentation: https://developer.android.com/topic/libraries/data-binding/index.html