View Binding

Jigisha Patel

View Binding

- *View binding* is a feature that allows you to more easily write code that interacts with views.
- Once view binding is enabled in a module, it generates a *binding class* for each XML layout file present in that module.
- An instance of a binding class contains direct references to all views that have an ID in the corresponding layout.
- In most cases, view binding replaces findViewById.

Setting up View Binding

- View binding is enabled on a module-by-module basis.
- To enable view binding in a module, set the viewBinding build option to true in the module-level build.gradle file,

```
android {
        ...
    viewBinding {
        enabled true
    }
}
```

Using View Binding

- If view binding is enabled for a module, a **binding class** is generated for each XML layout file that the module contains.

- Each binding class contains **references to the root view** and all views that have an **ID**.

- The name of the binding class is generated by converting the name of the XML file to Pascal case and adding the word "**Binding**" to the end.

```
<LinearLayout ... >
    <TextView android:id="@+id/tv_name" />
    <ImageView android:cropToPadding="true" />
    <Button android:id="@+id/btn_play"
        android:background="@drawable/rounded_button" />
</LinearLayout>
```

- Given a layout file called activity_main.xml, the generated binding class is called **ActivityMainBinding**.
- This class has two fields:
  - a TextView called tv_name and
  - a Button called btn_play.
- The ImageView in the layout has no ID, so there is no reference to it in the binding class.

- Every binding class also includes a **getRoot()** method, providing a direct **reference** for the root view of the corresponding layout file.

- In this example, the getRoot() method in the ActivityMainBinding class returns the LinearLayout root view.

Using View Binding in Activity

- To set up an instance of the binding class for use with an activity, perform the following steps in the activity's **onCreate()** method:
  1. Call the static **inflate()** method included in the generated binding class. This creates an instance of the binding class for the activity to use.
  2. Get a reference to the root view by either calling the **getRoot()** method or using Kotlin property syntax.
  3. Pass the root view to **setContentView()** to make it the active view on the screen.

findViewById vs View Binding

- View binding has important advantages over using findViewById:

- **Null safety:**
  - Since view binding creates direct references to views, there's no risk of a null pointer exception due to an invalid view ID.
  - Additionally, when a view is only present in some configurations of a layout, the field containing its reference in the binding class is marked with @Nullable.
- **Type safety:**
  - The fields in each binding class have types matching the views they reference in the XML file.
  - This means that there's no risk of a class cast exception.
- These differences mean that incompatibilities between your layout and your code will result in your build failing at compile time rather than at runtime.

References

- https://developer.android.com/topic/libraries/view-binding