Data Binding

Mobile technology - Exercise 4

Radek Vala

Data Binding

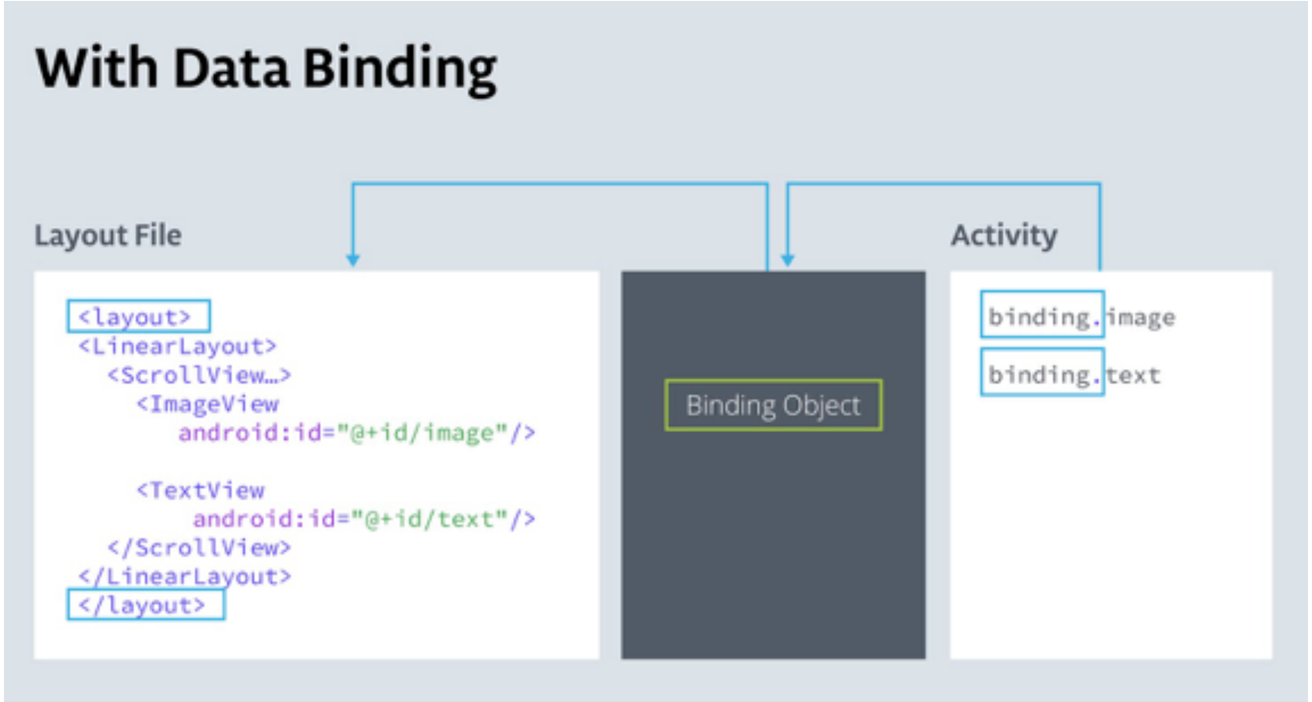
Intro

- Part of Android Jetpack
  - **Jetpack** is a **suite of libraries** to help developers **follow best practices**, reduce boilerplate code, and write code that works consistently across Android versions and devices so that developers can focus on the code they care about.
- The purpose is to **get rid** of **findViewById()** (it slows down the app in case of complex view hierarchies)
- Data Binding can eliminate use of **findViewById()**
- Using Data Binding you can access **data directly from view**

Data Binding

How it works

- Data Binding will create an object containing reference to each view
- The object is called Binding - whole app can use it



Data Binding

Benefits

- Shorter code, easier to read and maintain
- Data and views are separated
- Android OS traverses the view hierarchy only once, during app startup (not when users is interacting)
- There is type safety for accessing views

Data Binding

Enable data binding

- Enable in Gradle file
- build.gradle (Module: app)
  - section android before closing brace } add:

```
buildFeatures {

    dataBinding true

}
```

- Then File > Sync Project with Gradle Files

Data Binding

DEMO 1.1

- Create new Project
  - BasicActivity
  - Kotlin Support

Data Binding

DEMO 1.2

- Enable Data Binding in gradle file

Data Binding

DEMO 1.3

- Change layout files to be usable with data binding:
  - open layout/fragment_first.xml
  - wrap current root viewgroup with layout tag
  - move all namespace (xmlns) attributes from wrapped tag to root layout tag
  - Resulting part of source on right side:

```
<?xml version="1.0" encoding="utf-8"?>

<layout

    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools">



    <androidx.constraintlayout.widget.ConstraintLayout

        android:layout_width="match_parent"

        android:layout_height="match_parent"

        tools:context=".FirstFragment">
```

Data Binding
DEMO 1.4

- Create a binding object in Fragment (FirstFragment.kt)
- Declare binding variable in the FragmentsClass:

```
private var _binding: FragmentFirstBinding? = null  // nullable
private val binding get() =_binding as FragmentFirstBinding // getter of non-nullable
```

- In fragment's onCreateView() method create the binding object, get and return view:

```
_binding = DataBindingUtil.inflate(inflater, R.layout.fragment_first, container,
false)

val view = binding.root

return view
```

Data Binding
DEMO 1.5

- Use Data Binding in Fragment
- Now you can write binding.VIEW_OBJ_ID to get the View Object
- In FirstFragment.kt onViewCreated() method, get rid of findViewById():
- In fragment’s onCreateView() method create the binding object, get and return view:

```
binding.buttonFirst.setOnClickListener {

    findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)

}
```

Data Binding

DEMO 1.6

- Using Data Binding, change the textView text on FirstFragment
- In FirstFragment.kt - onViewCreated() method:

```
binding.textviewFirst.text = getString(R.string.app_name)
```

Data Binding
DEMO 1.7

- Change layout file to be usable with data binding:
  - open layout/activity_first.xml
  - wrap current root viewgroup with layout tag
  - move all namespace (xmlns) attributes from wrapped tag to root layout tag
  - Resulting part of source on right side:

```
<?xml version="1.0" encoding="utf-8"?>

<layout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools">



    <androidx.coordinatorlayout.widget.CoordinatorLayout

        android:layout_width="match_parent"

        android:layout_height="match_parent"

        tools:context=".MainActivity">
```

Data Binding
DEMO 1.8

- Create Data Binding object in Activity
- In MainActivity.kt, declare the variable in MainActivity class:

```
private lateinit var binding: ActivityMainBinding
```

- in onCreate, remove setContentView() method, create binding object instance, instead:

```
//setContentView(R.layout.activity_main)

binding = DataBindingUtil.setContentView(this, R.layout.activity_main)
```

Data Binding

DEMO 1.9

```
<data>
    <variable
        name=“person"
        type=“com.example.android.data.Person" />
</data>
```

- Bind View to data
- Create a data class for your data
- Add <data> block inside the <layout> block
- Define a <variable> with a name, and a type that is the data class.
- In MainActivity, create a variable with an instance of the data class. For example:
  private val person: Person = Person(“John" ,”Doe”)
- In the binding object, set the variable to the variable you just created: binding.person = person
  n the XML, set the content of the view to the variable that you defined in the <data> block. Use dot notation to access the data inside the data class.
  android:text=“@={person.name}”

Data Binding
DEMO 1.10

- Revalidate existing views with changed data

When data changes in the data class, you can revalidate them using apply method:

```
binding.apply {

    person?.name = binding.editText.text.toString()

    invalidateAll()

}
```