Android Studio: MVVM basics


manifests
AndroidManifest.xml
java
com.example.user.myapplication
MainActivity
com.example.user.myapplication (androidTest)
com.example.user.myapplication (test)
generatedJava
android
com
com.example.user.myapplication
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: MyApplication3)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
1
2
android {
compileSdkVersion 28
defaultConfig {
applicationId "com.example.user.myapplication"
minSdkVersion 22
targetSdkVersion 28
versionCode 1
versionName "1.0"
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
dataBinding {
enabled = true
}
}
dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation 'com.android.support.constraint:constraint-layout:1.1.3'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}

activity_main.xml | app | MyApplication3 | MainActivity.java

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3
4  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.an
6      xmlns:tools="http://schemas.
7      android:layout_width="match_
8      android:layout_height="match_parent"
9      tools:context=".MainActivity">
10
11     <TextView
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="Hello World!"
15         app:layout_constraintBottom_toBottomOf="parent"
16         app:layout_constraintLeft_toLeftOf="parent"
17         app:layout_constraintRight_toRightOf="parent"
18         app:layout_constraintTop_toTopOf="parent" />
19
20 </android.support.constraint.ConstraintLayout>
```

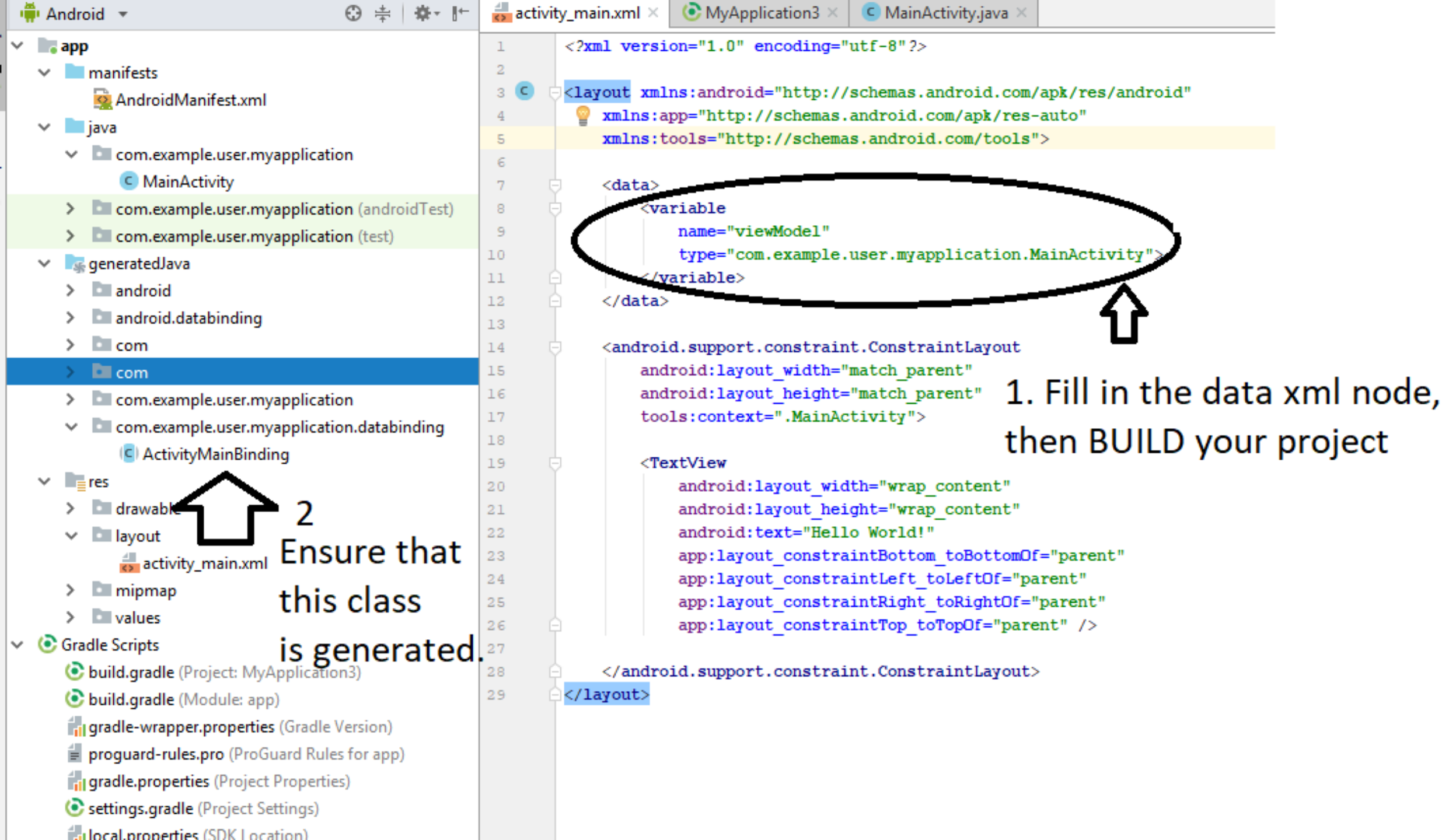
Convert to data binding layout
Override Resource in Other Configuration...
Annotate class 'ConstraintLayout' as @Deprecated

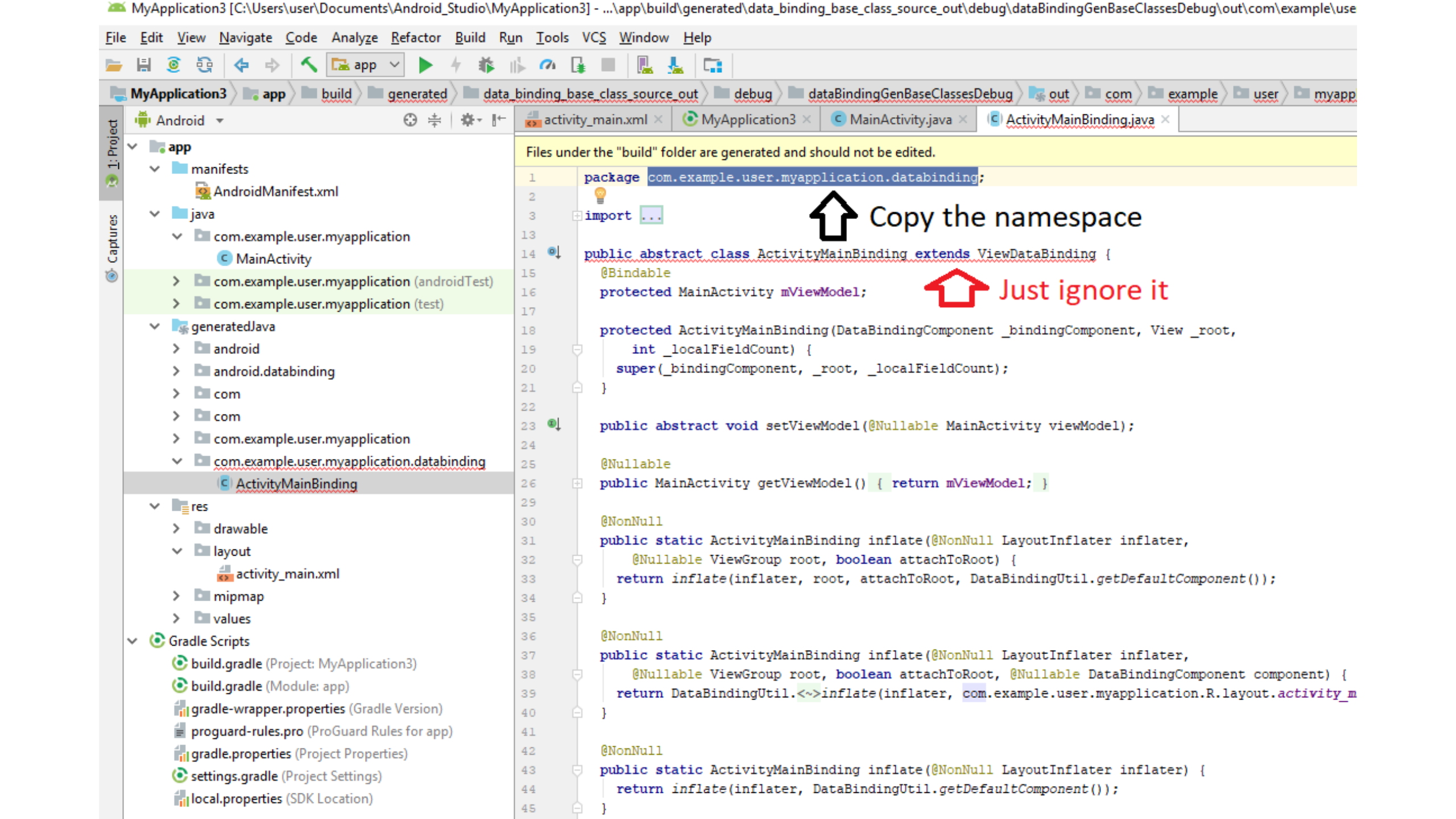
Alt+Enter then Enter

Results in...

activity_main.xml | app | MyApplication3 | MainActivity.java

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3
4  <layout xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      xmlns:tools="http://schemas.android.com/tools">
7
8      <data>
9
10     </data>
11
12     <android.support.constraint.ConstraintLayout
13         android:layout_width="match_parent"
14         android:layout_height="match_parent"
15         tools:context=".MainActivity">
16
17         <TextView
18             android:layout_width="wrap_content"
19             android:layout_height="wrap_content"
20             android:text="Hello World!"
21             app:layout_constraintBottom_toBottomOf="parent"
22             app:layout_constraintLeft_toLeftOf="parent"
23             app:layout_constraintRight_toRightOf="parent"
24             app:layout_constraintTop_toTopOf="parent" />
25
26     </android.support.constraint.ConstraintLayout>
27 </layout>
```


Android
app
manifests
AndroidManifest.xml
java
com.example.user.myapplication
MainActivity
com.example.user.myapplication (androidTest)
com.example.user.myapplication (test)
generatedJava
android
android.databinding
com
com
com.example.user.myapplication
com.example.user.myapplication.databinding
ActivityMainBinding
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: MyApplication3)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MyApplication3
MainActivity.java
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <layout xmlns:android="http://schemas.android.com/apk/res/android"
4 xmlns:app="http://schemas.android.com/apk/res-auto"
5 xmlns:tools="http://schemas.android.com/tools">
6
7 <data>
8 <variable
9 name="viewModel"
10 type="com.example.user.myapplication.MainActivity">
11 </variable>
12 </data>
13
14 <android.support.constraint.ConstraintLayout
15 android:layout_width="match_parent"
16 android:layout_height="match_parent"
17 tools:context=".MainActivity">
18
19 <TextView
20 android:layout_width="wrap_content"
21 android:layout_height="wrap_content"
22 android:text="Hello World!"
23 app:layout_constraintBottom_toBottomOf="parent"
24 app:layout_constraintLeft_toLeftOf="parent"
25 app:layout_constraintRight_toRightOf="parent"
26 app:layout_constraintTop_toTopOf="parent" />
27
28 </android.support.constraint.ConstraintLayout>
29 </layout>
1. Fill in the data xml node, then BUILD your project
2
Ensure that this class is generated.

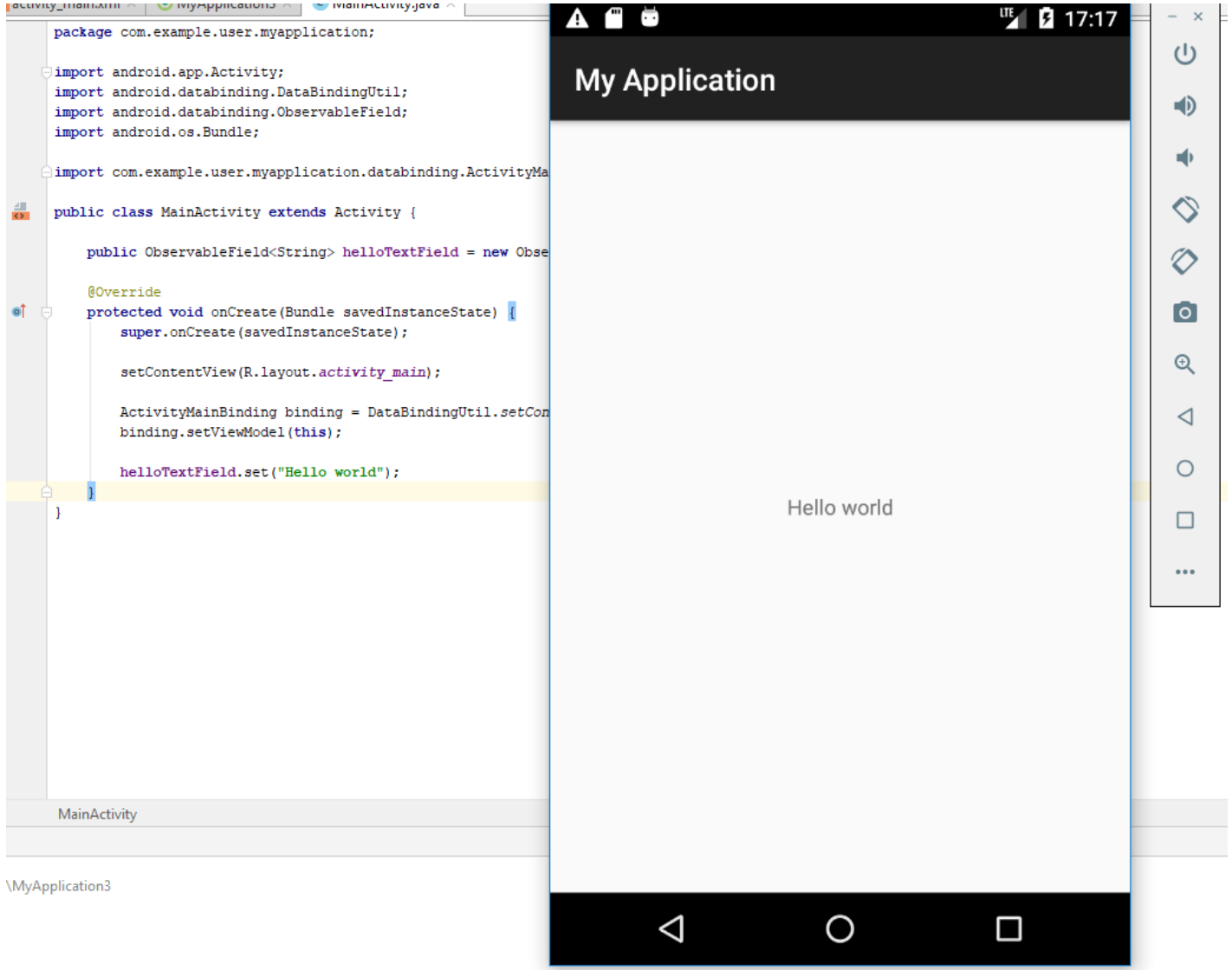

MyApplication3 [C:\Users\user\Documents\Android_Studio\MyApplication3] - ...\app\build\generated\data_binding_base_class_source_out\debug\dataBindingGenBaseClassesDebug\out\com\example\use
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app
MyApplication3 app build generated data_binding_base_class_source_out debug dataBindingGenBaseClassesDebug out com example user myapp
Android
app
manifests
AndroidManifest.xml
java
com.example.user.myapplication
MainActivity
com.example.user.myapplication (androidTest)
com.example.user.myapplication (test)
generatedJava
android
android.databinding
com
com
com.example.user.myapplication
com.example.user.myapplication.databinding
ActivityMainBinding
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: MyApplication3)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MyApplication3
MainActivity.java
ActivityMainBinding.java
Files under the "build" folder are generated and should not be edited.
package com.example.user.myapplication.databinding;
import ...
Copy the namespace
public abstract class ActivityMainBinding extends ViewDataBinding {
@Bindable
protected MainActivity mViewModel;
Just ignore it
protected ActivityMainBinding(DataBindingComponent _bindingComponent, View _root,
int _localFieldCount) {
super(_bindingComponent, _root, _localFieldCount);
}
public abstract void setViewModel(@Nullable MainActivity viewModel);
@Nullable
public MainActivity getViewModel() { return mViewModel; }
@NonNull
public static ActivityMainBinding inflate(@NonNull LayoutInflater inflater,
@Nullable ViewGroup root, boolean attachToRoot) {
return inflate(inflater, root, attachToRoot, DataBindingUtil.getDefaultComponent());
}
@NonNull
public static ActivityMainBinding inflate(@NonNull LayoutInflater inflater,
@Nullable ViewGroup root, boolean attachToRoot, @Nullable DataBindingComponent component) {
return DataBindingUtil.<~>inflate(inflater, com.example.user.myapplication.R.layout.activity_m
}
@NonNull
public static ActivityMainBinding inflate(@NonNull LayoutInflater inflater) {
return inflate(inflater, DataBindingUtil.getDefaultComponent());
}


activity_main.xml
MyApplication3
MainActivity.java
package com.example.user.myapplication;
import android.app.Activity;
import android.databinding.DataBindingUtil;
import android.databinding.ObservableField;
import android.os.Bundle;
import com.example.user.myapplication.databinding.ActivityMainBinding;
public class MainActivity extends Activity {
public ObservableField<String> helloTextField = new ObservableField<>();
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
ActivityMainBinding binding = DataBindingUtil.setContentView(activity: this, R.layout.activity_main);
binding.setViewModel(this);
helloTextField.set("Hello world");
}
}
1. Paste
2. Auto-complete with the binding class
Expose an observable field
Replace the line with this
Assign a value to your exposed field.

activity_main.xml | MyApplication3 | MainActivity.java

```xml
<?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">

    <data>
        <variable
            name="viewModel"
            type="com.example.user.myapplication.MainActivity">
        </variable>
    </data>

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{viewModel.helloTextField}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

    </android.support.constraint.ConstraintLayout>
</layout>
```

Bind the 'text' property of your TextView to your observable field.


package com.example.user.myapplication;
import android.app.Activity;
import android.databinding.DataBindingUtil;
import android.databinding.ObservableField;
import android.os.Bundle;
import com.example.user.myapplication.databinding.ActivityMa
public class MainActivity extends Activity {
public ObservableField<String> helloTextField = new Obse
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
ActivityMainBinding binding = DataBindingUtil.setCon
binding.setViewModel(this);
helloTextField.set("Hello world");
}
}
MainActivity
\MyApplication3
17:17
My Application
Hello world

activity_main.xml | MyApplication3 | MainActivity.java

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

    <data>
        <variable
            name="viewModel"
            type="com.example.user.myapplication.MainActivity">
        </variable>
    </data>

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{viewModel.helloTextField}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="@{viewModel.helloTextField}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.615"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.329" />

    </android.support.constraint.ConstraintLayout>
</layout>
```

1. Let's add a text box

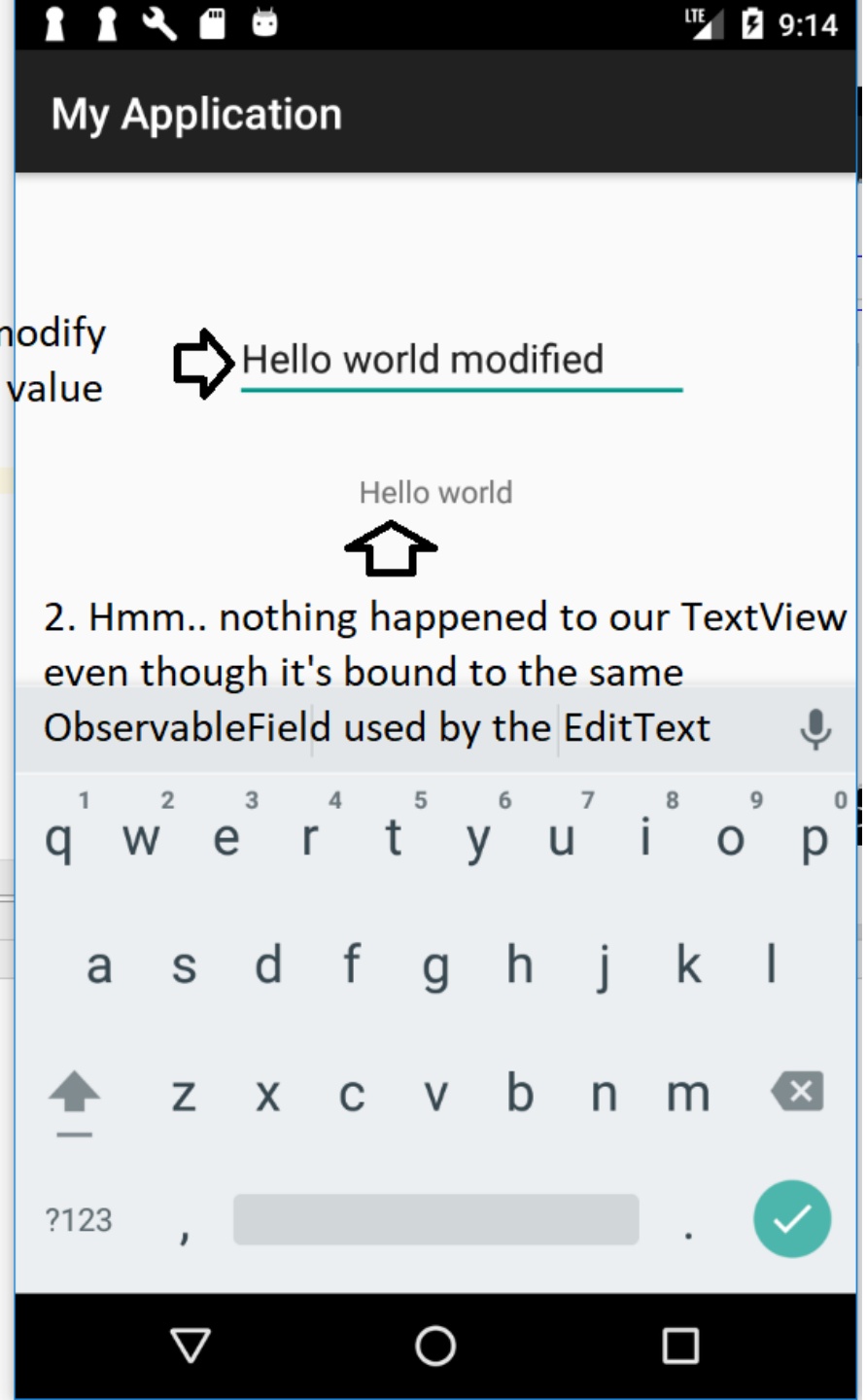
2. and bind its property 'text' to our observable field.

```
<android.support.constraint.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@{viewModel.helloTextField}"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.501"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.276" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="@{viewModel.helloTextField}"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.568"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.133" />


</android.support.constraint.ConstraintLayout>
```

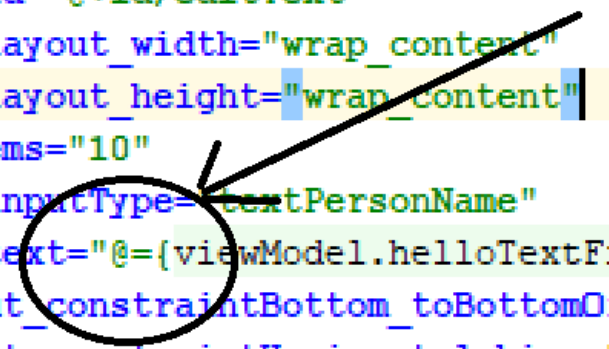
layout › android.support.constraint.ConstraintLayout › EditText

Text

1. modify the value

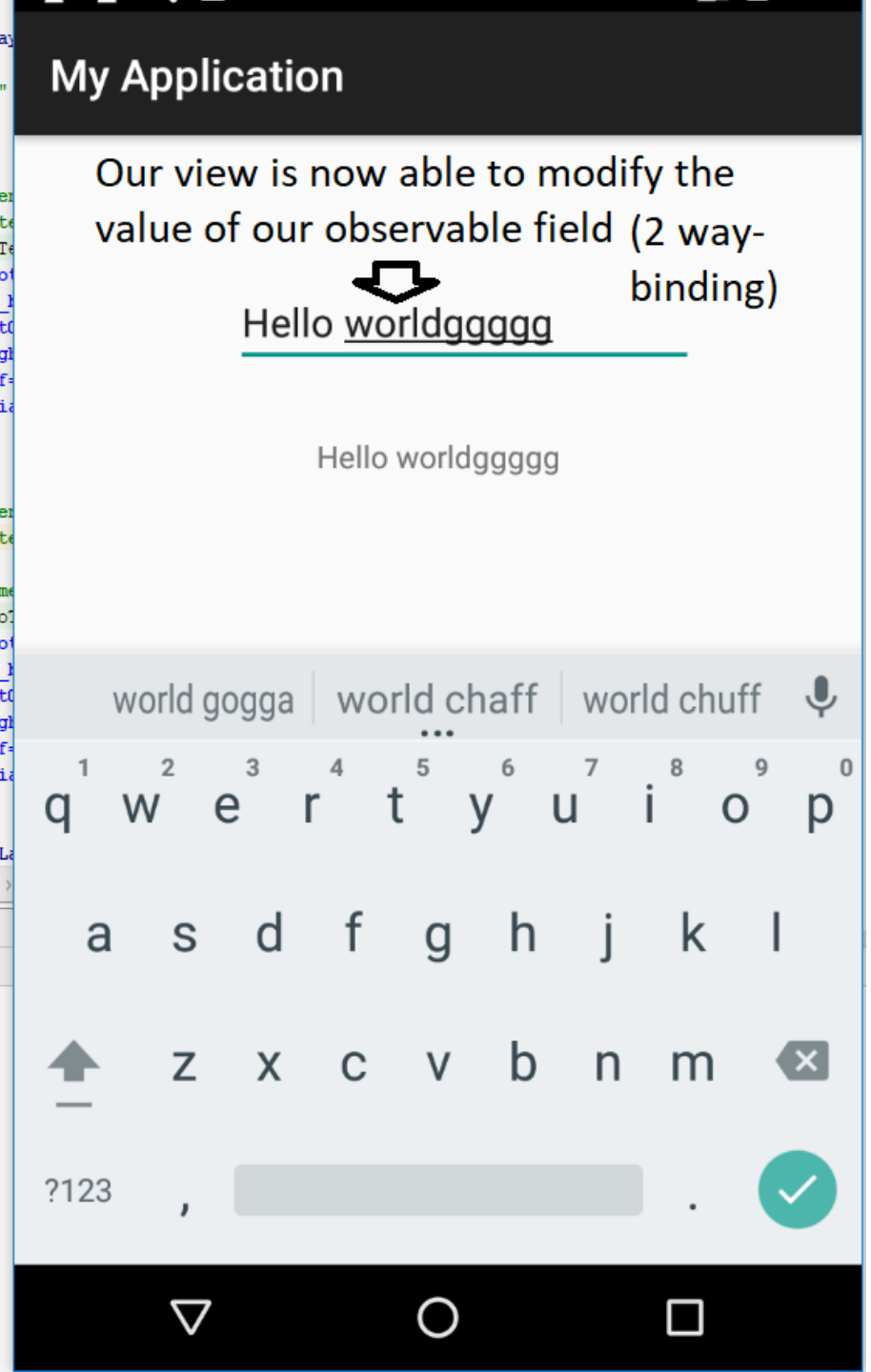


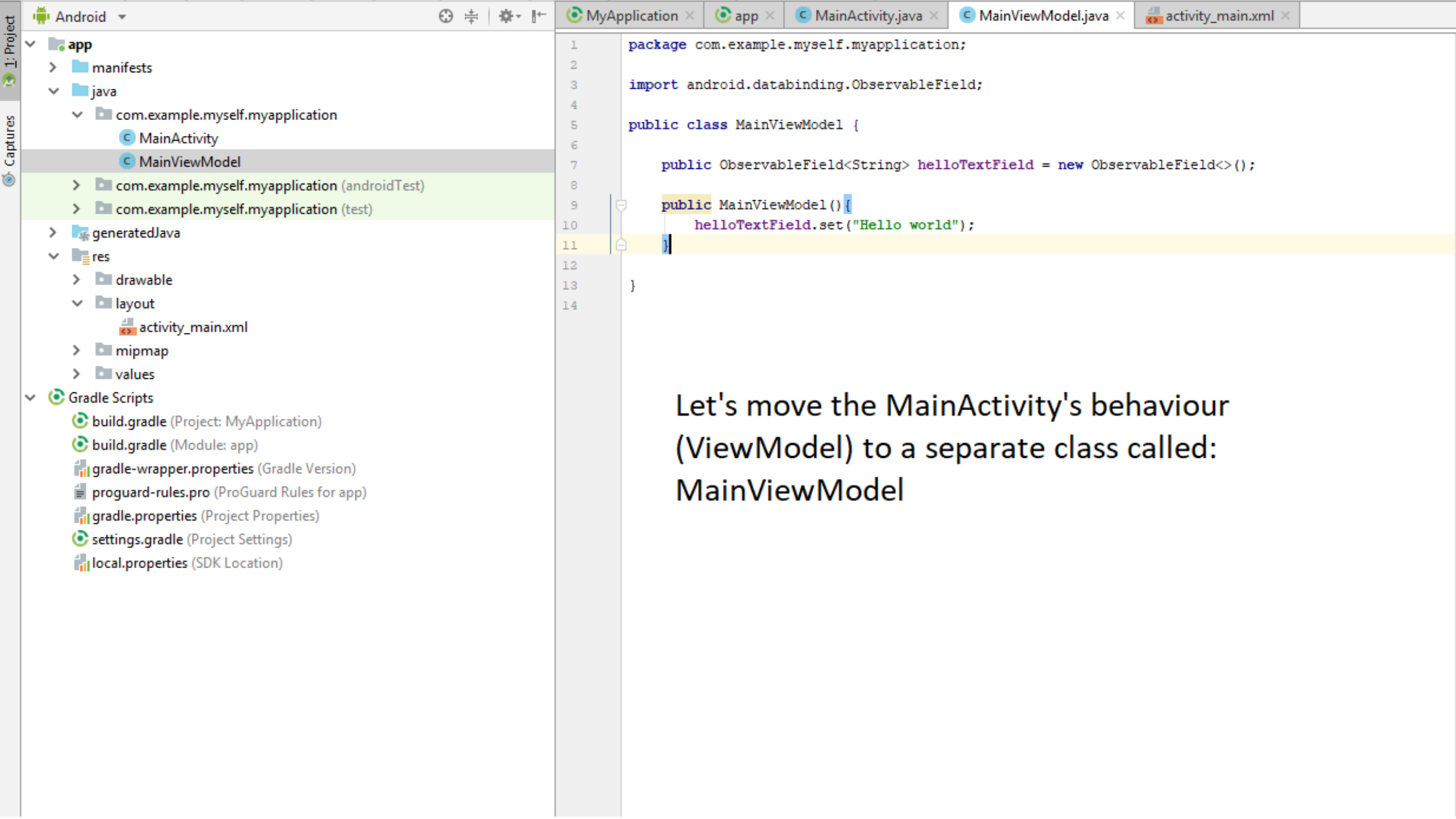
```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@{viewModel.helloTextField}"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.501"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.276" />

<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName"
    android:text="@={viewModel.helloTextField}"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.568"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.133" />
```



Let's switch to a two-way binding with '@=' instead of '@'

```
ndroid.support.constraint.ConstraintLay
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <TextView
      android:layout_width="wrap_conter
      android:layout_height="wrap_conte
      android:text="@{viewModel.helloTe
      app:layout_constraintBottom_toBot
      app:layout_constraintHorizontal_b
      app:layout_constraintLeft_toLeftO
      app:layout_constraintRight_toRigh
      app:layout_constraintTop_toTopOf=
      app:layout_constraintVertical_bia

  <EditText
      android:id="@+id/editText"
      android:layout_width="wrap_conter
      android:layout_height="wrap_conte
      android:ems="10"
      android:inputType="textPersonName
      android:text="@={viewModel.helloT
      app:layout_constraintBottom_toBot
      app:layout_constraintHorizontal_b
      app:layout_constraintLeft_toLeftO
      app:layout_constraintRight_toRigh
      app:layout_constraintTop_toTopOf=
      app:layout_constraintVertical_bia

'android.support.constraint.ConstraintLa
  android.support.constraint.ConstraintLayout
```




Android
app
manifests
java
com.example.myself.myapplication
MainActivity
MainViewModel
com.example.myself.myapplication (androidTest)
com.example.myself.myapplication (test)
generatedJava
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: MyApplication)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
MyApplication
app
MainActivity.java
MainViewModel.java
activity_main.xml
1 package com.example.myself.myapplication;
2
3 import android.databinding.ObservableField;
4
5 public class MainViewModel {
6
7 public ObservableField<String> helloTextField = new ObservableField<>();
8
9 public MainViewModel(){
10 helloTextField.set("Hello world");
11 }
12
13 }
14
Let's move the MainActivity's behaviour (ViewModel) to a separate class called: MainViewModel

MyApplication | app | MainActivity.java | MainViewModel.java | activity_main.xml

```
<?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">

    <data>

        <variable name="viewModel"
            type="com.example.myself.myapplication.MainViewModel"/>

    </data>

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{viewModel.helloTextField}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.501"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.276" />

        <EditText
            android:id="@+id/editText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="@={viewModel.helloTextField}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.568"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.133" />


    </android.support.constraint.ConstraintLayout>
</layout>
```

Change the binding context from MainActivity to MainViewModel

myself > myapplication > MainActivity

MyApplication | app | MainActivity.java | MainViewModel.java | activity_main.xml

```java
package com.example.myself.myapplication;

import android.app.Activity;
import android.databinding.DataBindingUtil;
import android.databinding.ObservableField;
import android.os.Bundle;
import  com.example.myself.myapplication.databinding.ActivityMainBinding;

public class MainActivity extends Activity {

    public ObservableField<String> helloTextField = new ObservableField<>();

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

        ActivityMainBinding binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_main);
        binding.setViewModel(new MainViewModel());

        helloTextField.set("Hello world");
    }
}
```

Change the instance of the binding context from MainActivty (this) to MainViewModel (new MainViewModel)

MyApplication | app | MainActivity.java | MainViewModel.java | activity_main.xml

```java
package com.example.myself.myapplication;

import android.app.Activity;
import android.databinding.DataBindingUtil;
import android.os.Bundle;
import  com.example.myself.myapplication.databinding.ActivityMainBinding;

public class MainActivity extends Activity {

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

        ActivityMainBinding binding = DataBindingUtil.setContentView( activity: t
        binding.setViewModel(new MainViewModel());
    }
}
```

Build and deploy: Ensure that everything's OK.

You have now achieved a perfect MVVM app architecture: Your ViewModel is completely independant of your View (and its code-behind).