

# Android Data Binding

# Was ist Data Binding ?

---

Als **Datenbindung** (engl. Data Binding) bezeichnet man die automatische Weitergabe von Daten zwischen **Objekten**. Typischerweise werden Daten aus einem Datenobjekt an ein **Steuerelement** der Benutzeroberfläche weitergegeben. Aber auch zwischen **Steuerelementen** ist **Datenbindung** in einigen Frameworks möglich.

## Beispiel

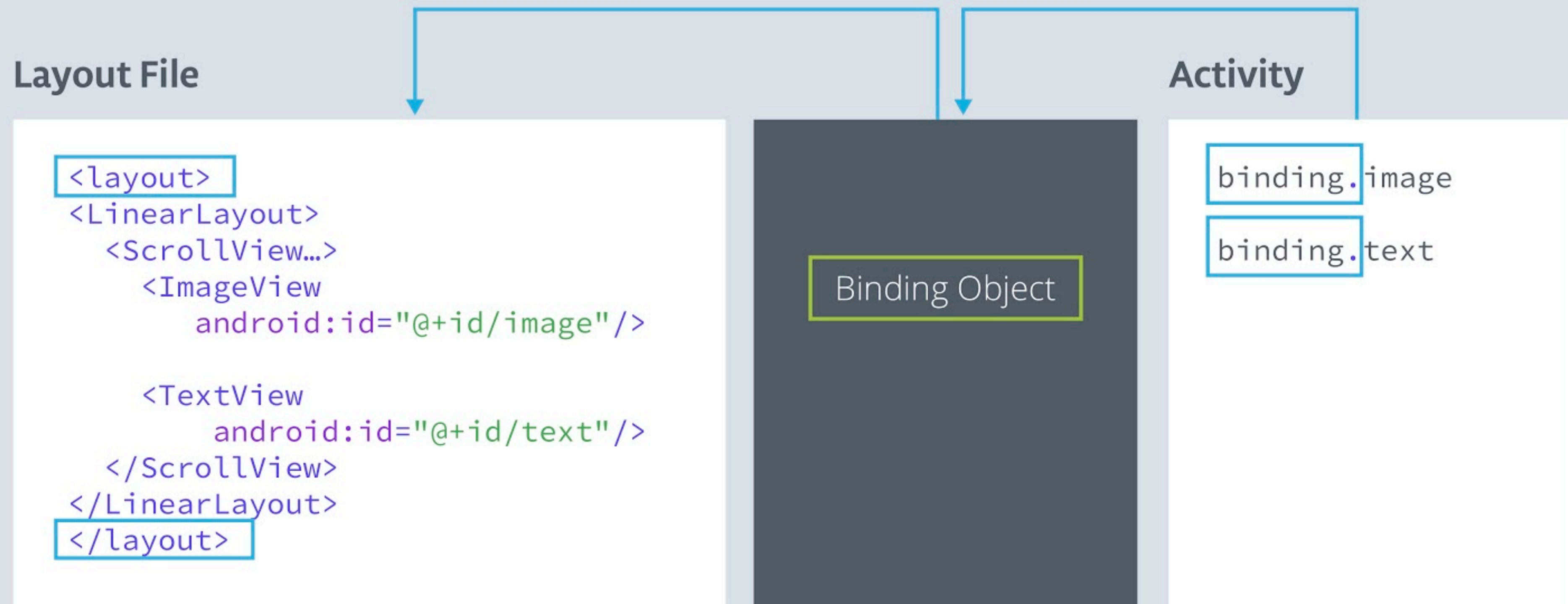
---

Wenn man eine Liste von Kunden auf dem Bildschirm in einer Tabelle ausgeben will, würde man klassischerweise eine Schleife über die Kundenliste ausführen (z.B. foreach) und dann für jedes Kundenelement eine Tabellenzeile mit den einzelnen Spalten erzeugen. Bei der **Datenbindung** entfällt die Schleife und die Erstellung der einzelnen Spalten. Vielmehr schreibt der Entwickler so etwas wie:

**Steuerelement**.DataSource = KundenListe

Das **Steuerelement** führt dann die Schleife selbst aus. In dem **Steuerelement** ist konfiguriert, welche Spalten es darstellen soll. Alternativ können einige **Steuerelemente** alle Eigenschaften des Kunden ausgeben, sofern es keine explizite Definition gibt.

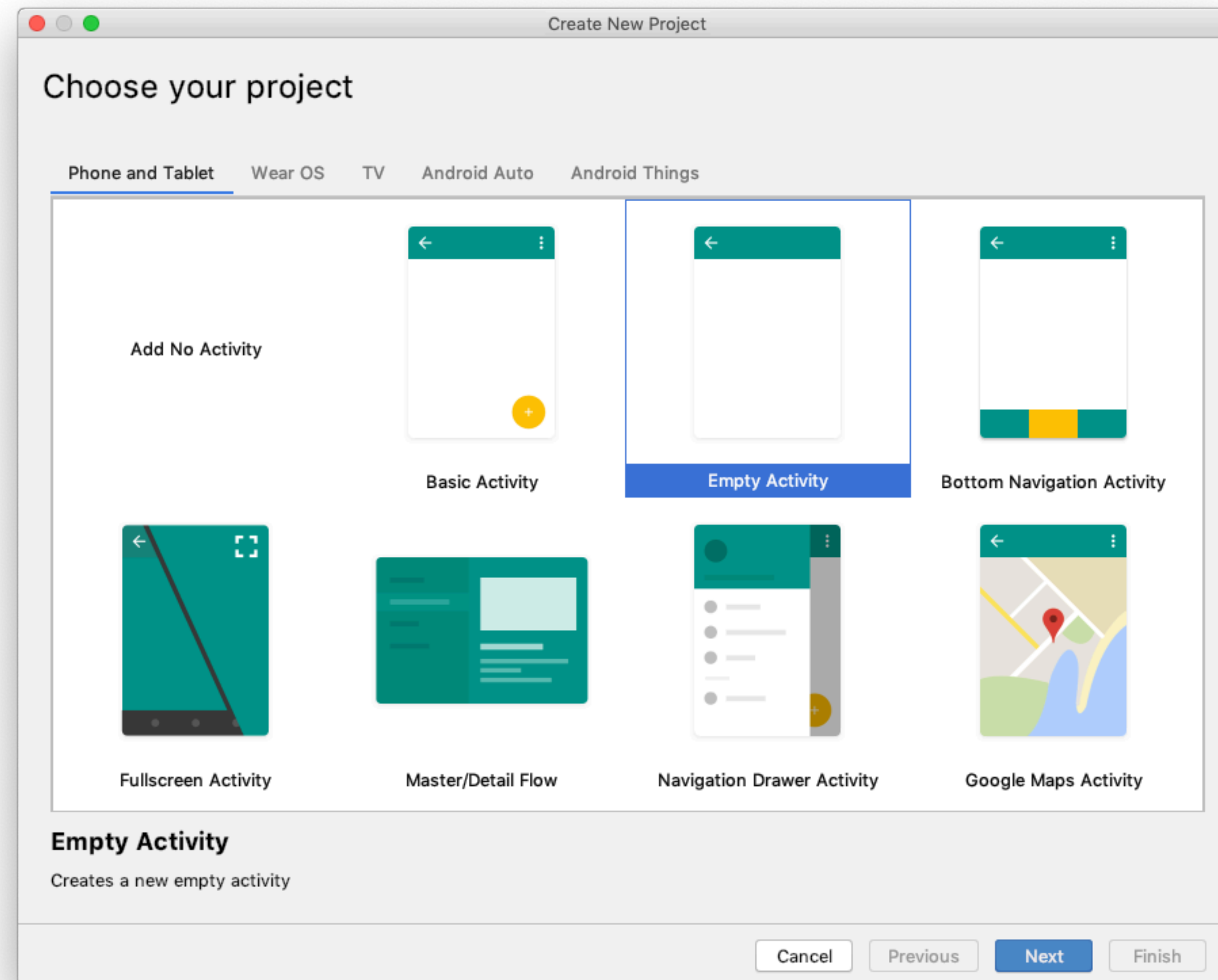
# With Data Binding



# DataBinding konfigurieren

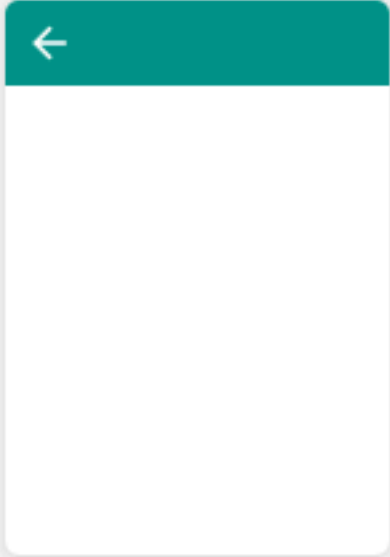


# Neues Projekt erstellen



Create New Project

### Configure your project



Empty Activity

Creates a new empty activity

Name  
Paging

Package name  
at.htl.paging

Save location  
/Users/stuetz/work/jetpack.packt/projects/Paging

Language  
Kotlin

Minimum API level  
API 23: Android 6.0 (Marshmallow)

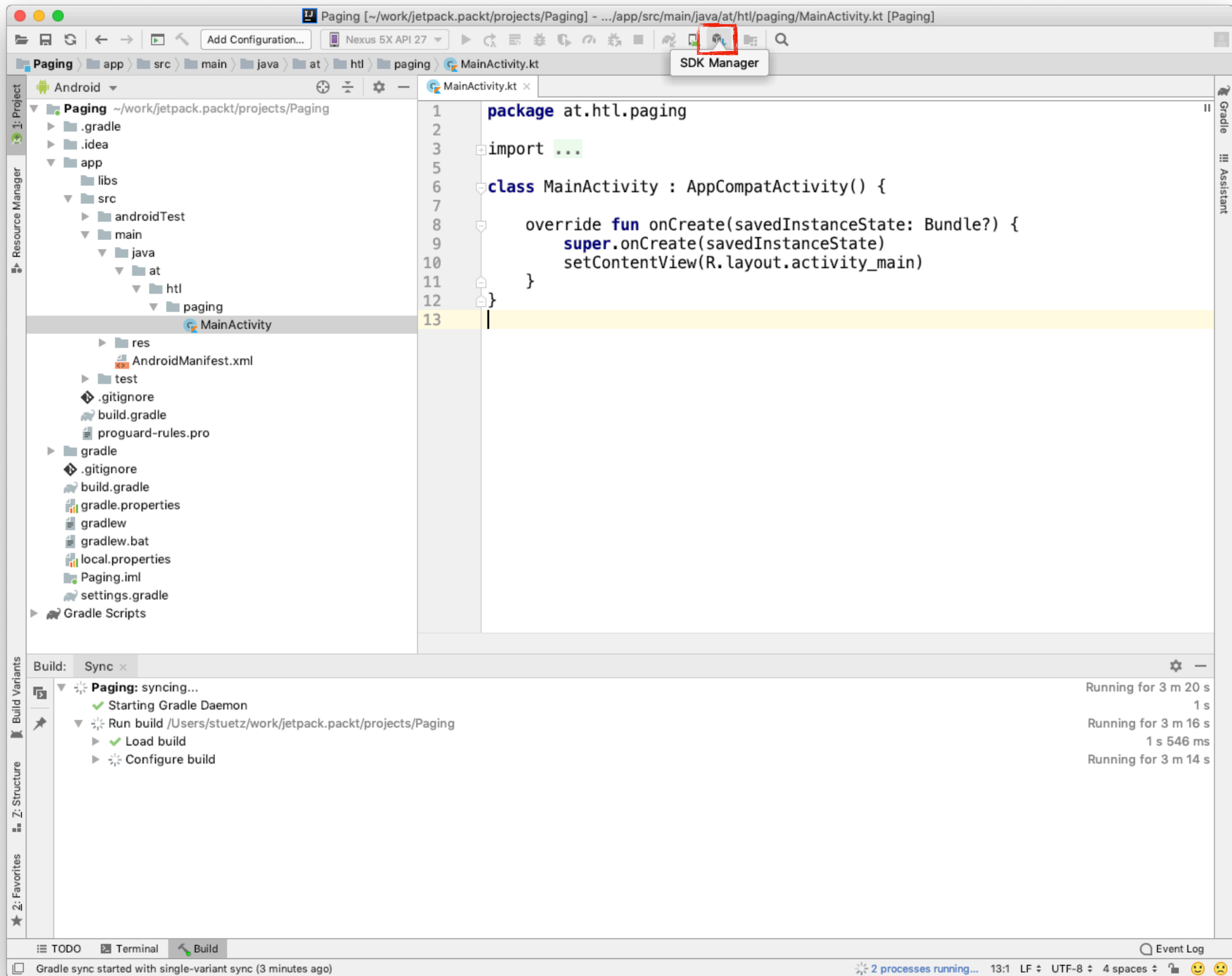
**i** Your app will run on approximately **62.6%** of devices.  
[Help me choose](#)

☐ This project will support instant apps

☒ Use AndroidX artifacts

Cancel Previous Next Finish

Verwenden Sie statt „Paging“  
besser „DataBinding“





Paging [~/work/jetpack.packt/projects/Paging] - .../app/src/main/java/at/htl/paging/MainActivity.kt [Paging]

Add Configuration... Nexus 5X API 27

Preferences for New Projects

Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /opt/android-sdk-macosx Edit

SDK Platforms SDK Tools SDK Update Sites

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

	Name	Version	Status
<input checked="" type="checkbox"/>	Android SDK Build-Tools		Installed
<input type="checkbox"/>	GPU Debugging tools		Not Installed
<input type="checkbox"/>	LLDB		Not Installed
<input type="checkbox"/>	CMake		Not Installed
<input checked="" type="checkbox"/>	Android Auto API Simulators	1.0.0	Installed
<input type="checkbox"/>	Android Auto Desktop Head Unit emulator	1.1	Not installed
<input type="checkbox"/>	Android Emulator	28.1.6	Not installed
<input checked="" type="checkbox"/>	Android SDK Platform-Tools	28.0.1	Installed
<input checked="" type="checkbox"/>	Android SDK Tools	26.1.1	Installed
<input checked="" type="checkbox"/>	Android Support Library	23.2.1	Installed
<input checked="" type="checkbox"/>	Documentation for Android SDK	1	Installed
<input checked="" type="checkbox"/>	Google Play APK Expansion Library, rev 3	3.0.0	Installed
<input type="checkbox"/>	Google Play APK Expansion library	1	Not installed
<input checked="" type="checkbox"/>	Google Play Billing Library, rev 5	5.0.0	Installed
<input type="checkbox"/>	Google Play Instant Development SDK	1.6.0	Not installed
<input type="checkbox"/>	Google Play Licensing Library	1	Not installed
<input checked="" type="checkbox"/>	Google Play Licensing Library, rev 2	2.0.0	Installed
<input checked="" type="checkbox"/>	Google Play services	49	Installed
<input checked="" type="checkbox"/>	Google Play services for Froyo, rev 12	12.0.0	Installed
<input checked="" type="checkbox"/>	Google Web Driver, rev 2	2.0.0	Installed
<input checked="" type="checkbox"/>	Intel x86 Emulator Accelerator (HAXM installer)	7.3.2	Installed
<input type="checkbox"/>	NDK	19.1.5304403	Not installed
<input checked="" type="checkbox"/>	Support Repository		
<input checked="" type="checkbox"/>	ConstraintLayout for Android		Installed
<input checked="" type="checkbox"/>	Solver for ConstraintLayout		Installed
<input checked="" type="checkbox"/>	Android Support Repository	47.0.0	Installed
<input checked="" type="checkbox"/>	Google Repository	58	Installed

☒ Hide Obsolete Packages ☐ Show Package Details

Cancel Apply OK

kontrollieren, ob Support Repository installiert ist

Build: Sync

Build Variants

2: Favorites

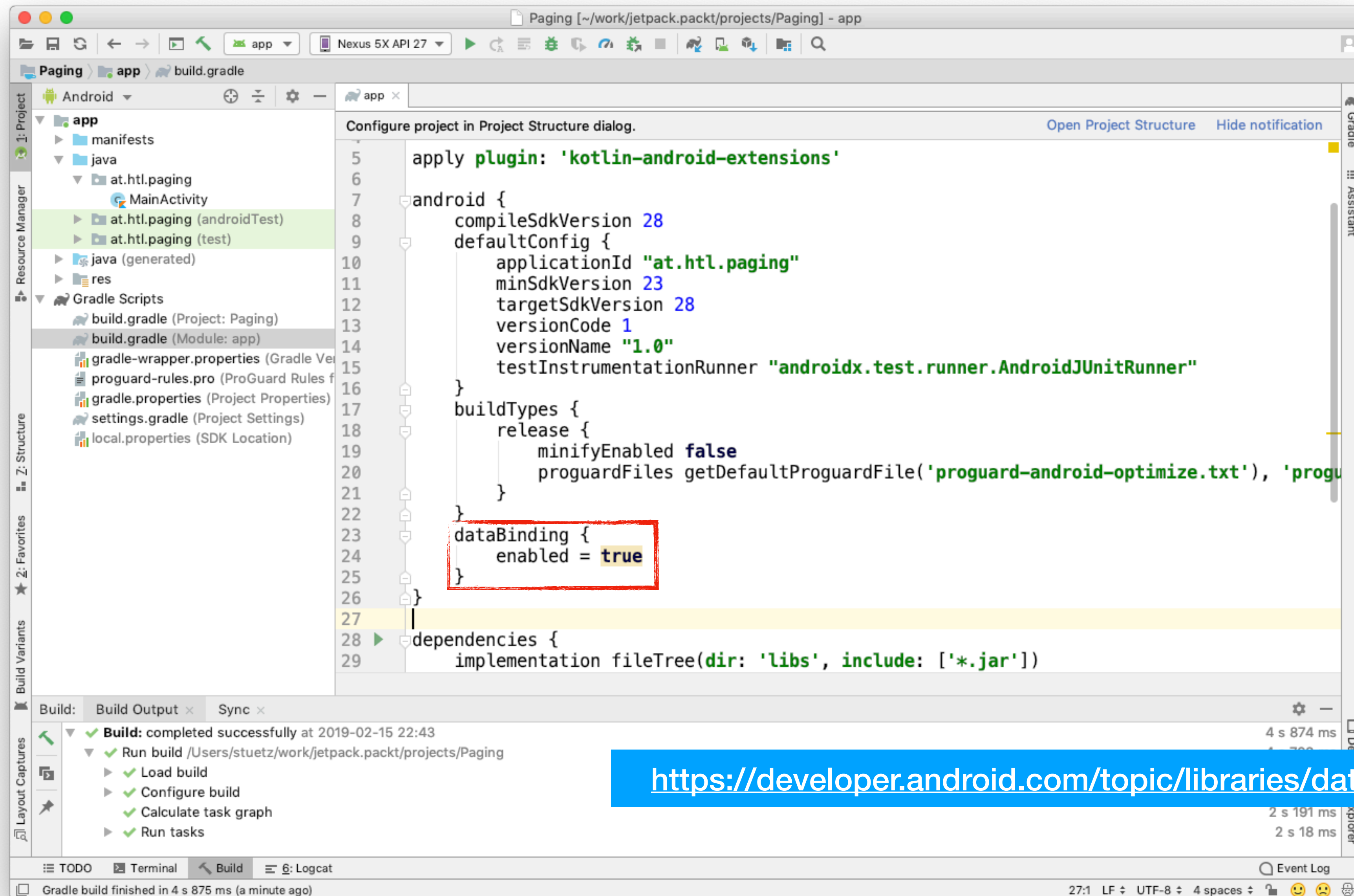
7: Structure

Event Log

Gradle sync started with single-variant sync (4 minutes ago)

2 processes running... 13:1 LF UTF-8 4 spaces

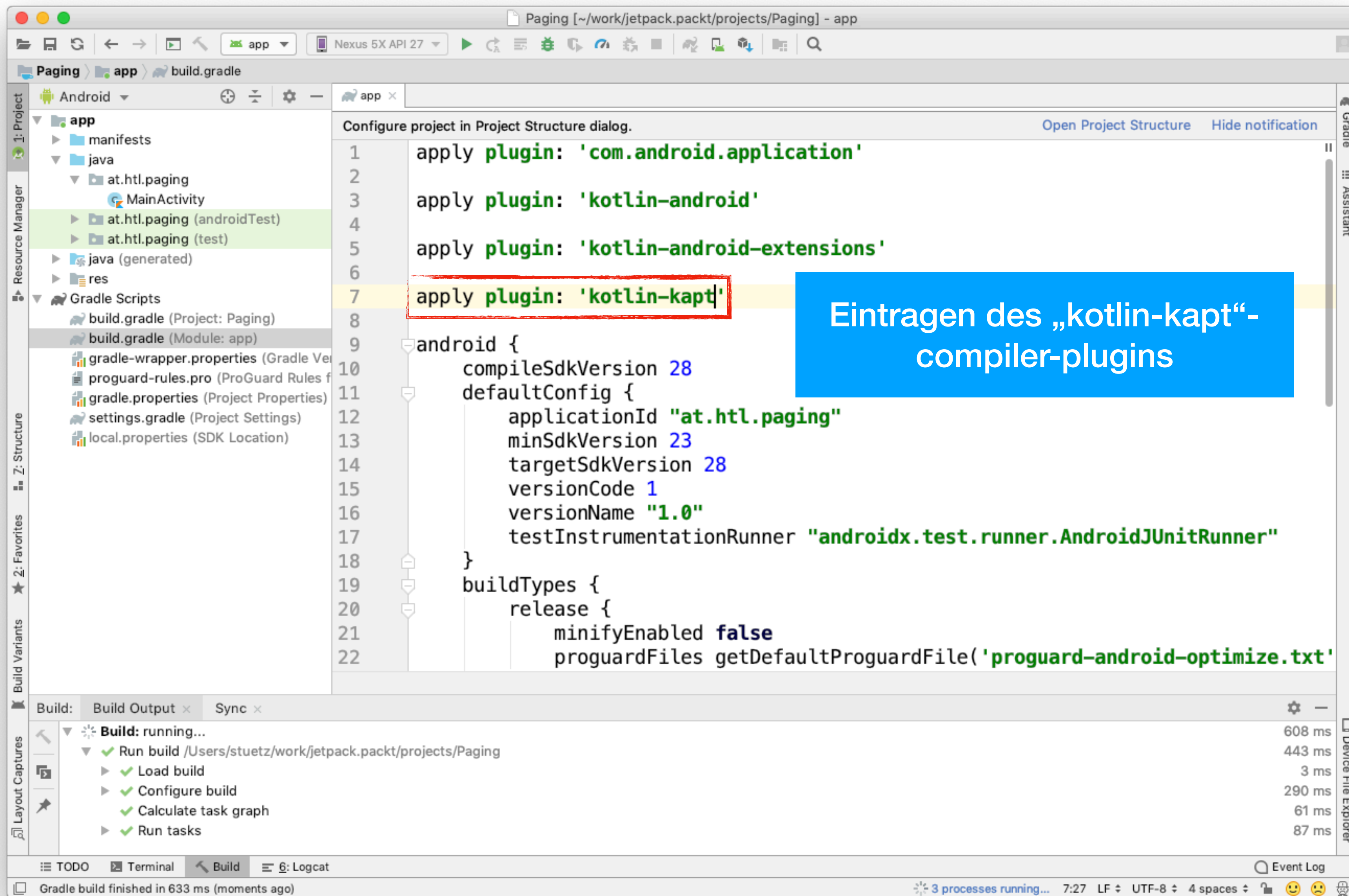
# dataBinding in build.xml eintragen



<https://developer.android.com/topic/libraries/data-binding/start>

# Layouts and Binding Expressions





<https://kotlinlang.org/docs/reference/kapt.html>

1 Markieren des Bereichs

2 Surround With via `\t\t` (Ctrl+Alt+T for Win/Linux)

3 option auswählen

4 als Tag „layout“ eintragen

Durch Verwendung des `<layout>`-Tags wird DataBinding aktiviert.

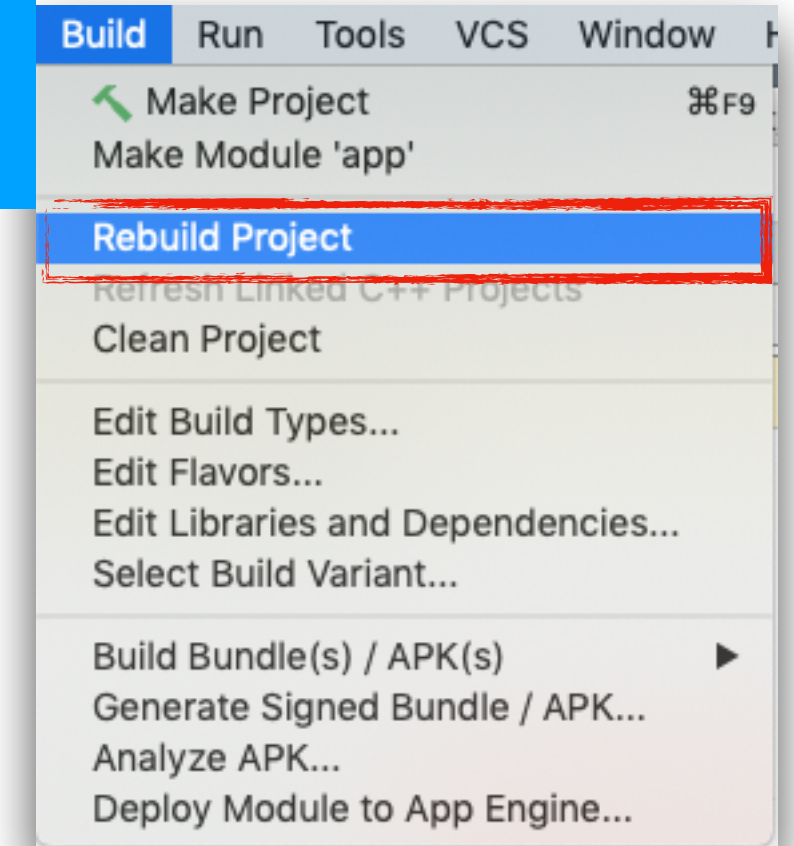
Es wird automatisch eine Data-Binding-Klasse für jedes Layout generiert.

Die Name der Klasse hängt vom Namen des Layout-Files ab. In unserem Fall wird die DataBinding-Klasse „ActivityMainBinding“

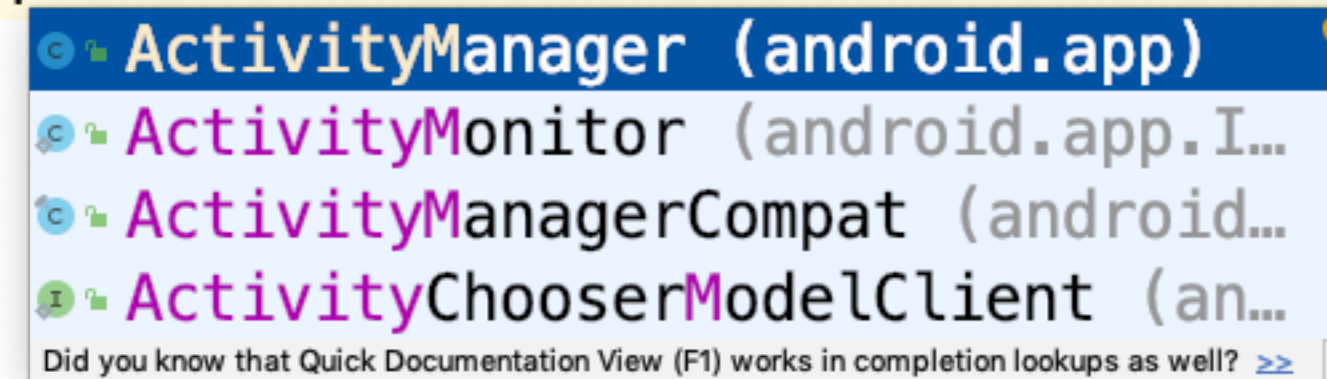
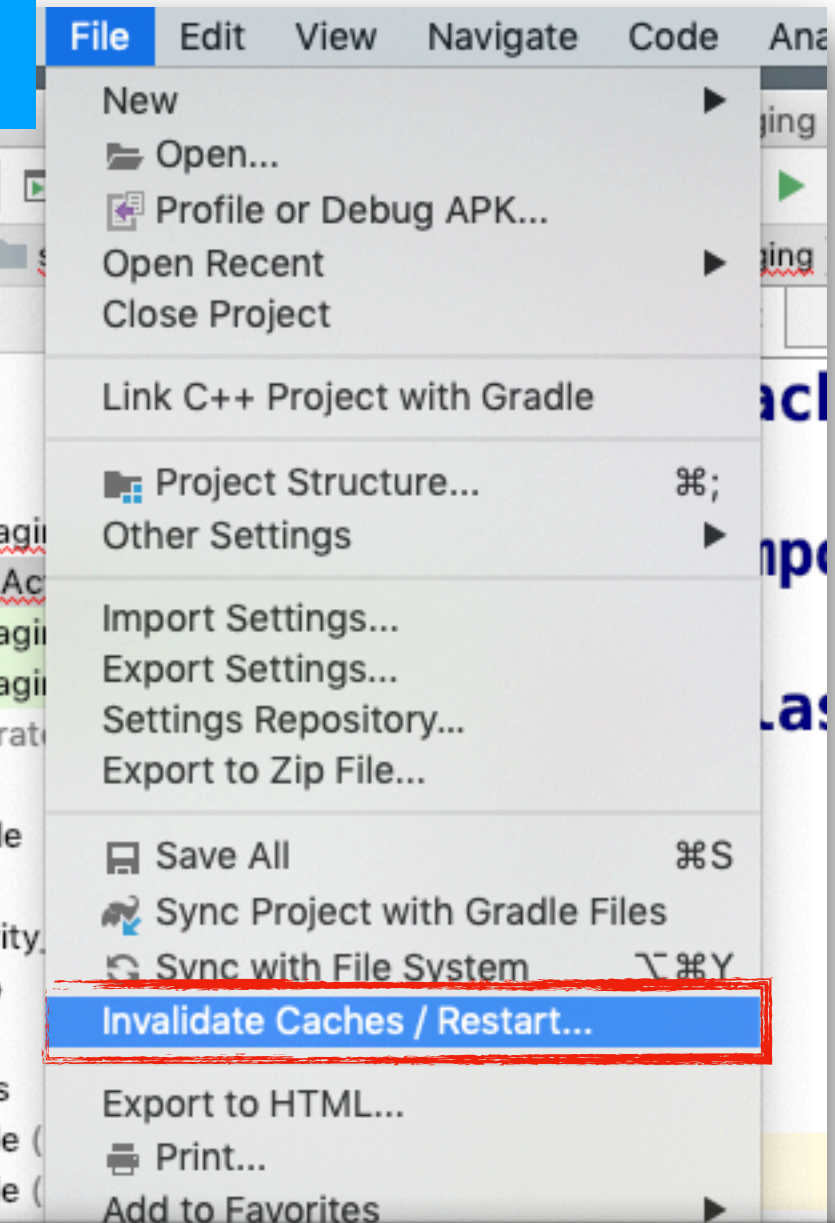


Falls das AutoComplete noch nicht funktioniert, kann man folgendes machen

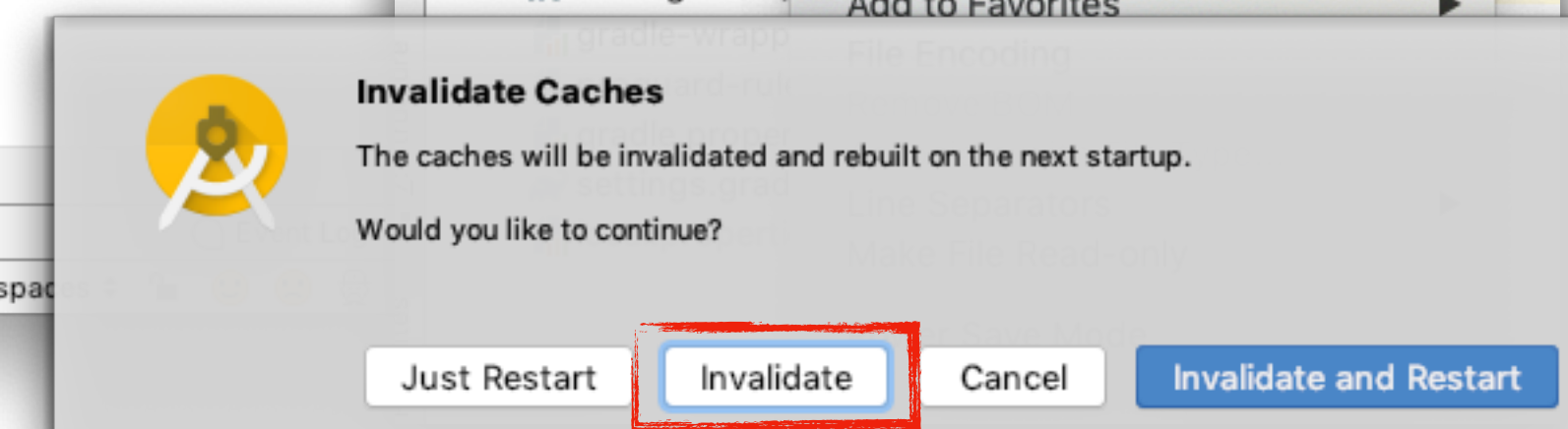
Maßnahme 1: Rebuild



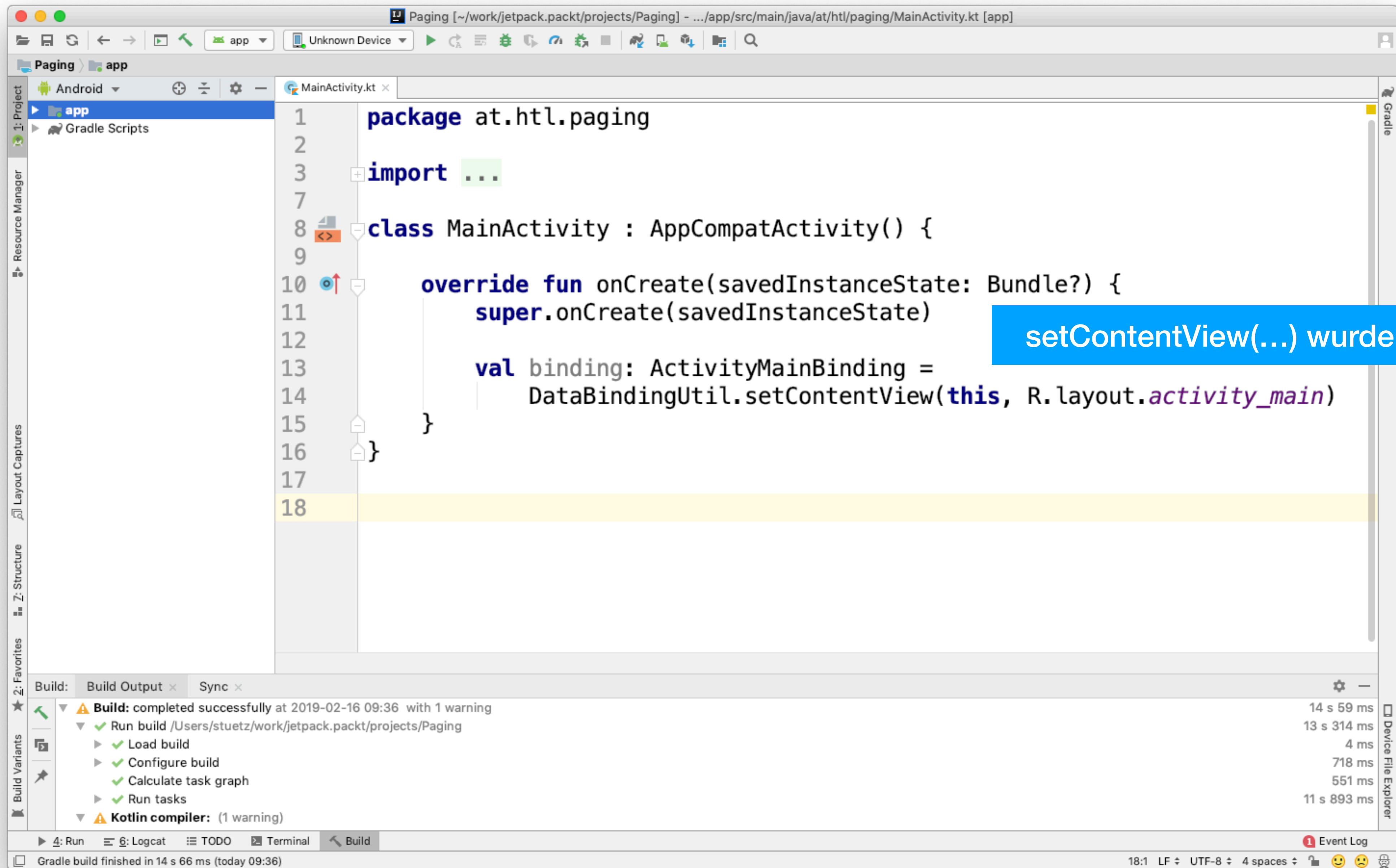
Maßnahme 2: Invalidate Caches



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







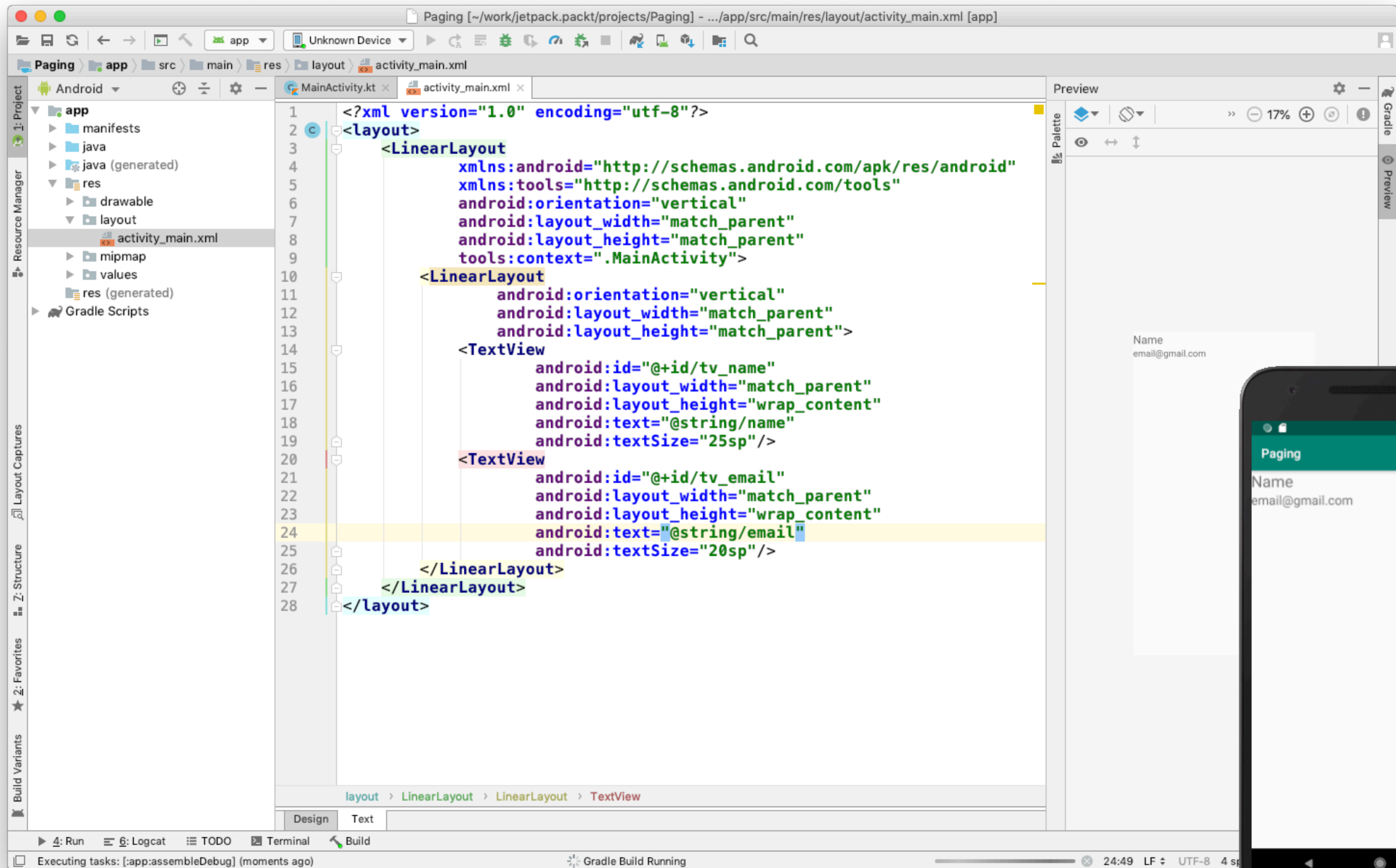
# Erstellen von String Ressourcen

The screenshot illustrates the steps to create string resources in Android Studio:

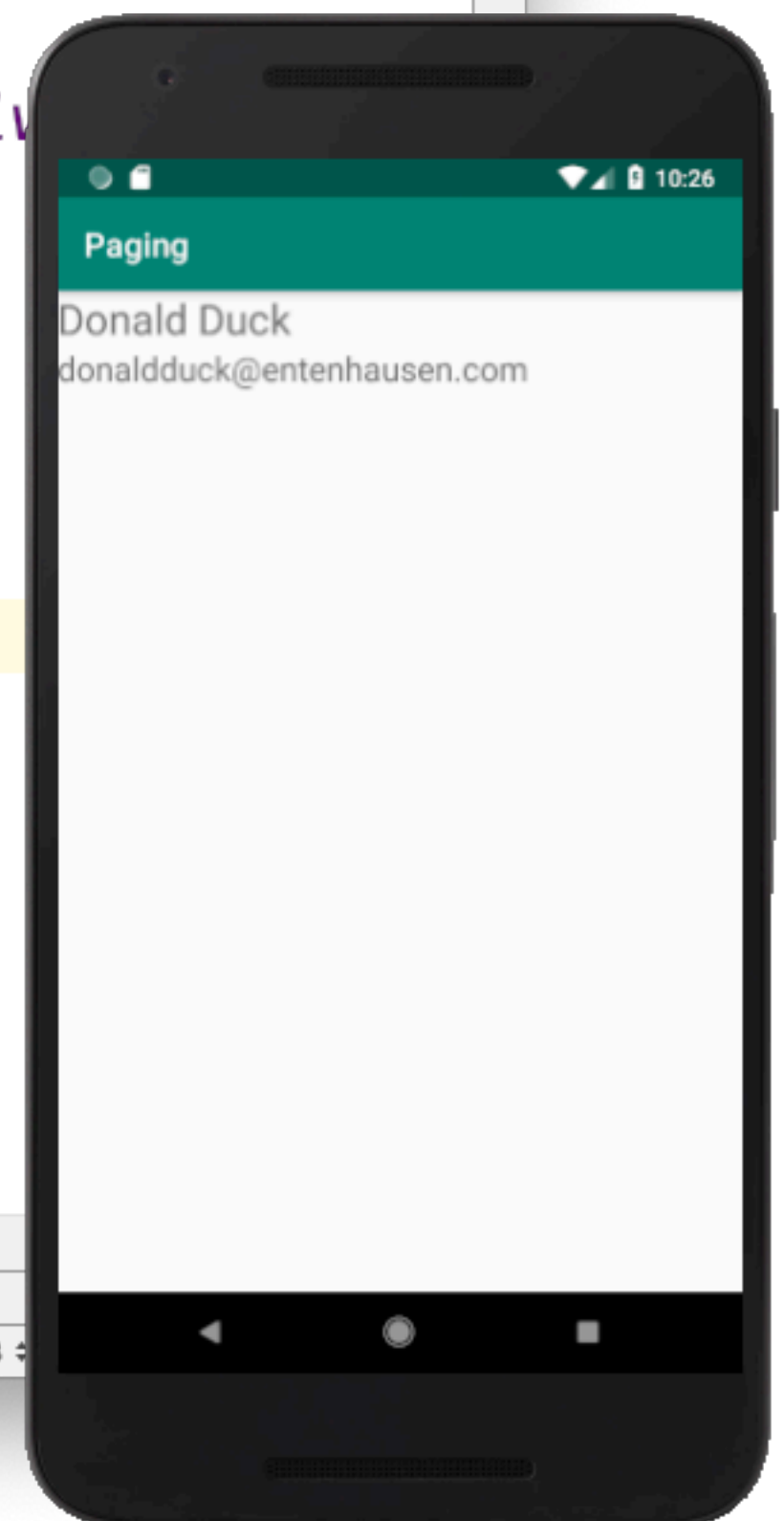
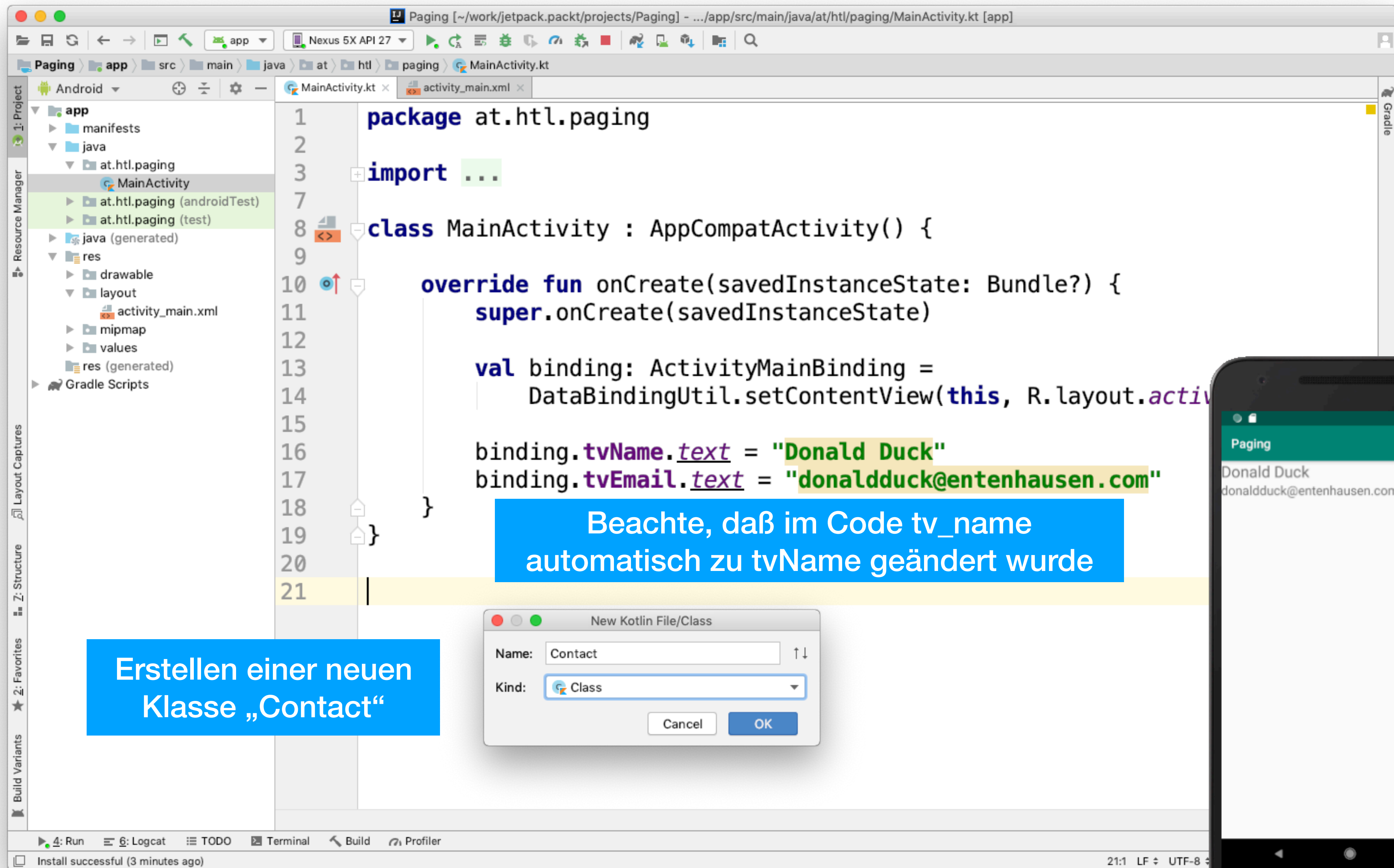
- 1**: Select the `text` attribute in the `Ab tv_email` widget's Attributes panel.
- 2**: Click `Add new resource` and then `New string Value...` in the Resources panel.
- 3**: Fill out the `New String Value Resource` dialog:
  - Resource name: `email`
  - Resource value: `email@gmail.com`
  - Source set: `main`
  - File name: `strings.xml`
  - Check `values` under "Create the resource in directories:"
- 4**: Click `OK` to confirm.

The final state shows the `Ab tv_email` widget with the `text` attribute set to `@string/email` and the `Ab tv_name` widget with the `text` attribute set to `@string/name`.

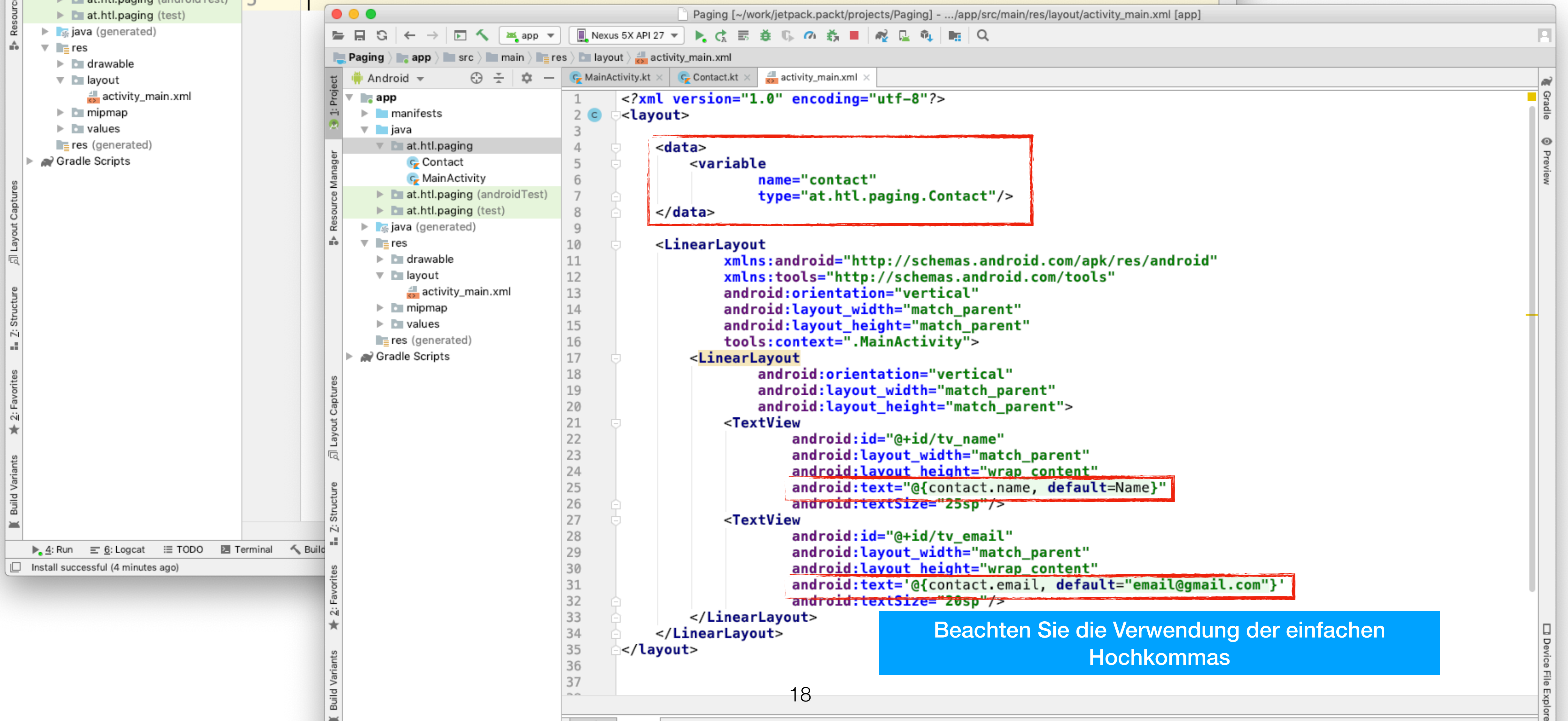
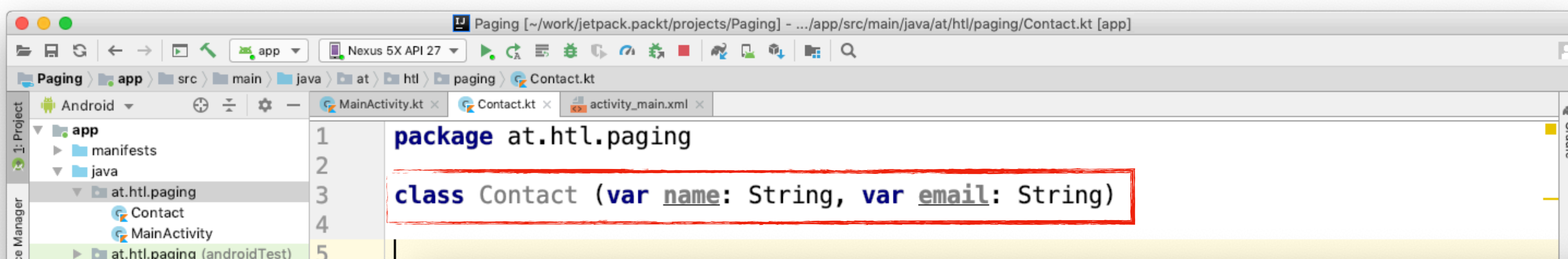






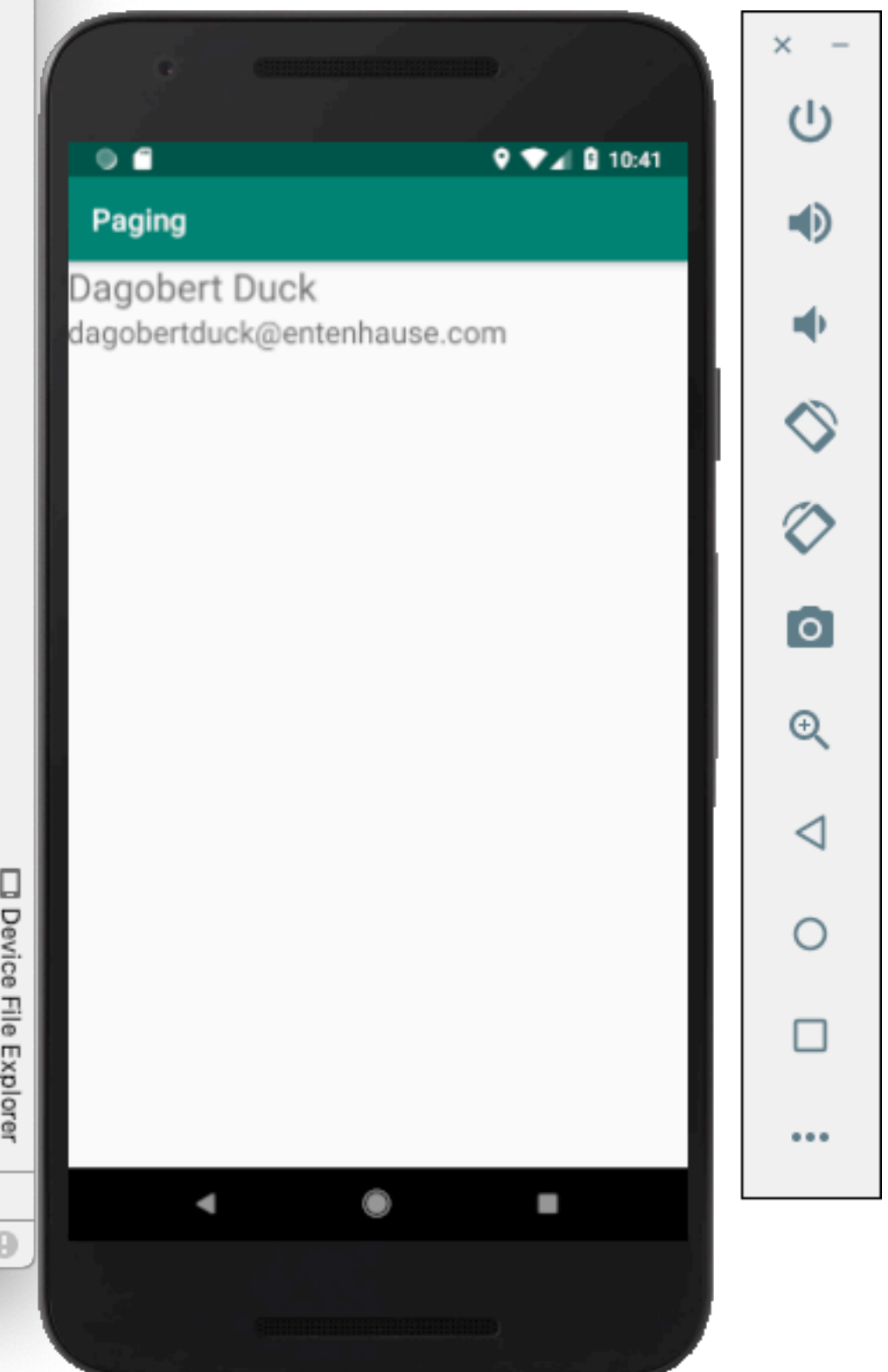
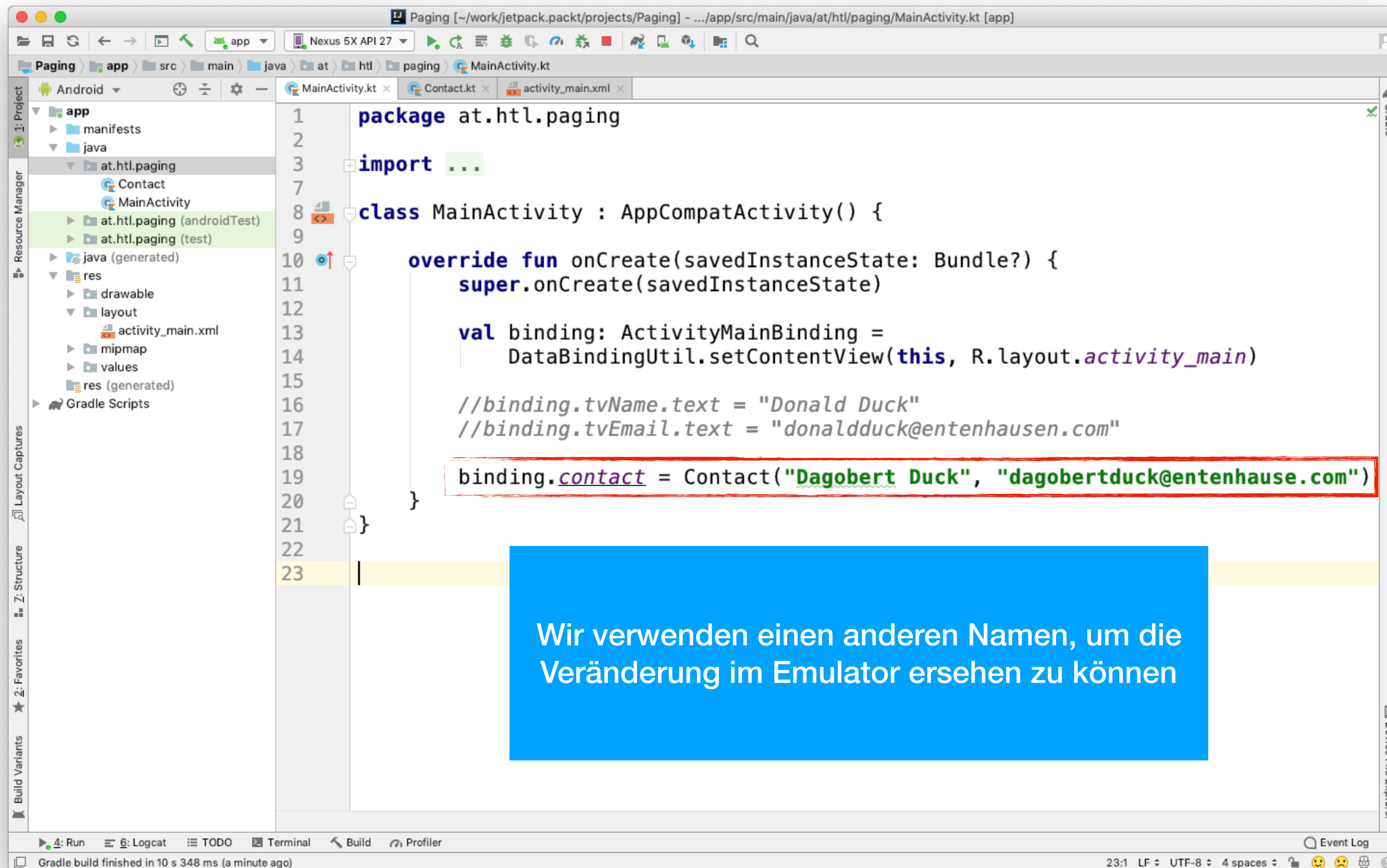






Beachten Sie die Verwendung der einfachen Hochkommas





New Kotlin File/Class

Name:

Kind:

Paging [~/work/jetpack.packt/projects/Paging] - .../app/src/main/java/at/htl/paging/EventHandler.kt [app]

Nexus 5X API 27

app

src

main

java

at

htl

paging

EventHandler.kt

1 package at.htl.paging

2

3 import android.content.Context

4 import android.widget.Toast

5

6 open class EventHandler(context: Context) {

7

8 val myContext = context

9

10 fun onClick() {

11 Toast.makeText(myContext, "Hello", Toast.LENGTH\_SHORT).show()

12 }

13 }

14

15

16

app

manifests

java

at.htl.paging

Contact

EventHandler

MainActivity

at.htl.paging (androidTest)

at.htl.paging (test)

java (generated)

res

drawable

layout

activity\_main.xml

mipmap

values

res (generated)

Gradle Scripts

Build Variants

2: Favorites

Z: Structure

Layout Captures

Device File Explorer

4: Run

6: Logcat

TODO

Terminal

Build

Profiler

Event Log

Install successful (12 minutes ago)

16:1 LF UTF-8 4 spaces



```

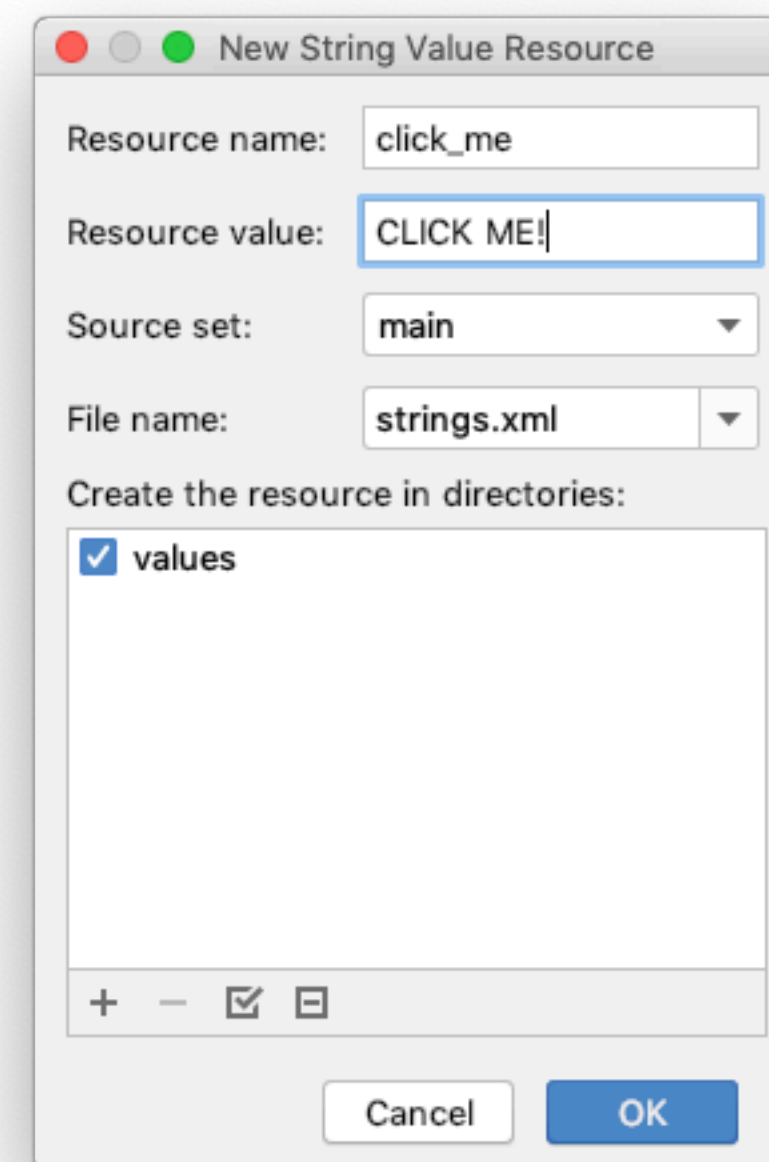
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="contact"
            type="at.htl.paging.Contact"/>
        <variable
            name="handler"
            type="at.htl.paging.EventHandler"/>
    </data>

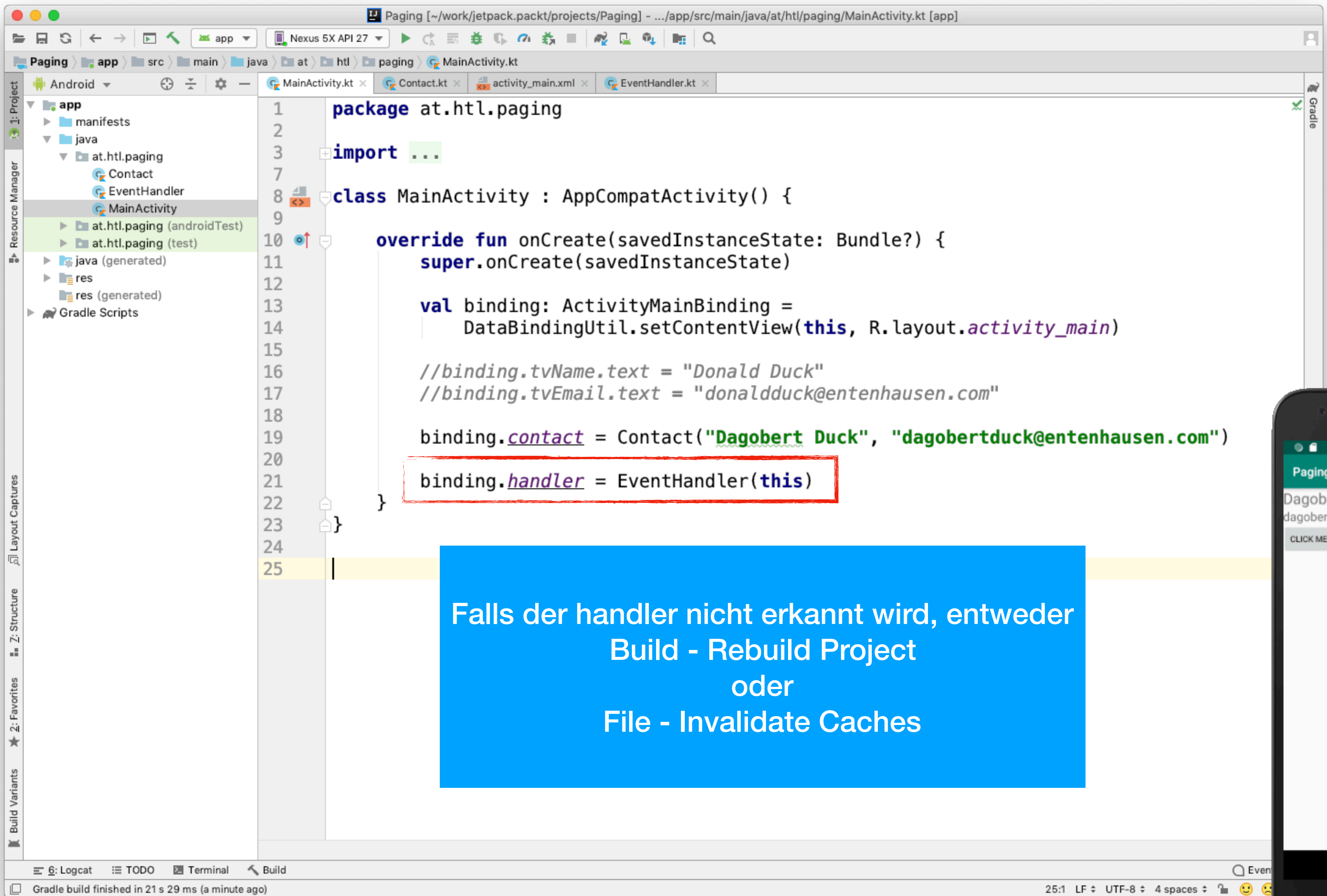
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <TextView
                android:id="@+id/tv_name"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@{contact.name, default=Name}"
                android:textSize="25sp"/>
            <TextView
                android:id="@+id/tv_email"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@{contact.email, default="email@gmail.com"}"
                android:textSize="20sp"/>
            <Button
                android:id="@+id/btn_click"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="CLICK ME!"
                android:onClick="@{() -> handler.onButtonClick()}" />
        </LinearLayout>
    </LinearLayout>
</layout>

```

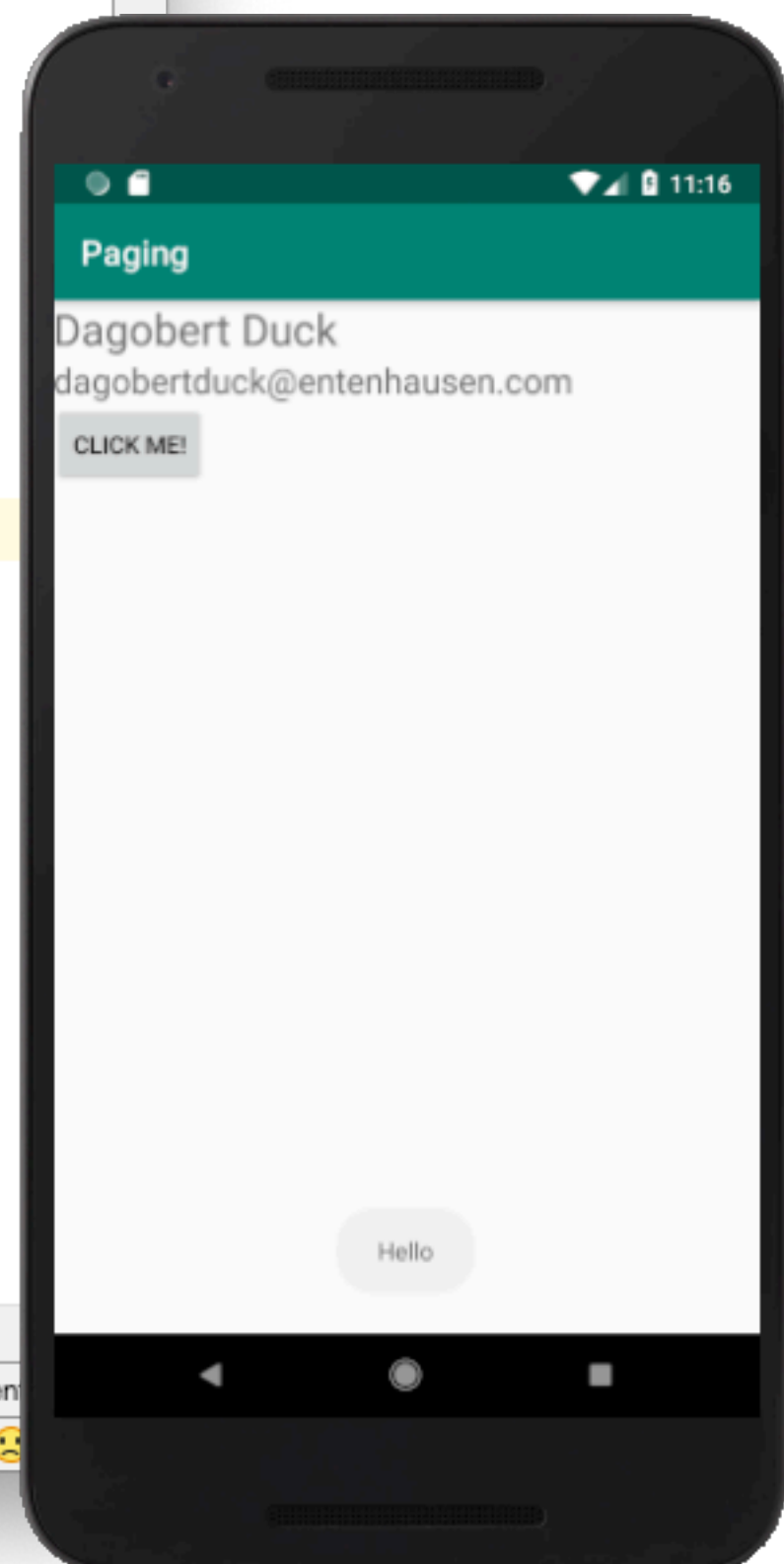
Der EventHandler wird als Variable eingetragen



Ebenso die onClick-Methode



Falls der handler nicht erkannt wird, entweder  
Build - Rebuild Project  
oder  
File - Invalidate Caches





# Zusammenfassung

- In gradle.build dataBinding-enabled und kotlin-kapt-plugin eintragen
- <layout> - Tag in layout-Resource eintragen
- Datenklassen und EventHandler-Klassen erstellen
- Datenklassen und EventHandler als <variable> in layout-Resource eintragen
- In der layout-Resource den Zugriff auf die Variablen mit #{...} eintragen
- In der zugehörigen Activity werden nun die Variablen (Datenobjekte, Handler, ...) mit Werten initialisiert

# Observable Data Objects

1: Project

Android

app

manifests

java

at.htl.paging

Contact

EventHandler

MainActivity

at.htl.paging (androidTest)

at.htl.paging (test)

java (generated)

res

drawable

layout

mipmap

values

res (generated)

Gradle Scripts

Build Variants

2: Favorites

Z: Structure

Layout Captures

Paging [~/work/jetpack.packt/projects/Paging] - .../app/src/main/java/at/htl/paging/Contact.kt [app]

Nexus 5X API 27

app

src

main

java

at

htl

paging

Contact.kt

MainActivity.kt

Contact.kt

activity\_main.xml

EventHandler.kt

```
1 package at.htl.paging
2
3 import androidx.databinding.BaseObservable
4 import androidx.databinding.Bindable
5
6 class Contact(var _name: String, var _email: String) : BaseObservable() {
7
8     @get:Bindable
9     var name: String = _name
10     set(value) {
11         field = value
12         notifyPropertyChanged(BR.name)
13     }
14
15     @get:Bindable
16     var email: String = _email
17     set(value) {
18         field = value
19         notifyPropertyChanged(BR.email)
20     }
21 }
22
23
```

Beachte: Die Parameterbezeichnungen wurden geändert

BR ist eine generierte Klasse, die alle Ressourcen beinhaltet, die für das DataBinding benötigt werden

@get  
Vgl. <https://developer.android.com/topic/libraries/data-binding/observability>

6: Logcat

TODO

Terminal

Build

Profiler

4: Run

Event Log

Gradle build finished in 7 s 895 ms (a minute ago)

23:1

LF

UTF-8

4 spaces



```

<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="contact"
            type="at.htl.paging.Contact"/>
        <variable
            name="handler"
            type="at.htl.paging.EventHandler"/>
    </data>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <TextView
                android:id="@+id/tv_name"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@{contact.name, default=Name}"
                android:textSize="25sp"/>
            <TextView
                android:id="@+id/tv_email"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@{contact.email, default="email@gmail.com"}"
                android:textSize="20sp"/>
            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/et_name"
                android:inputType="text"
                android:text="@={contact.name, default=Name}"/>
            <Button
                android:id="@+id/btn_click"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="CLICK ME!"
                android:onClick="@{() -> handler.onButtonClick()}" />
        </LinearLayout>
    </LinearLayout>
</layout>

```

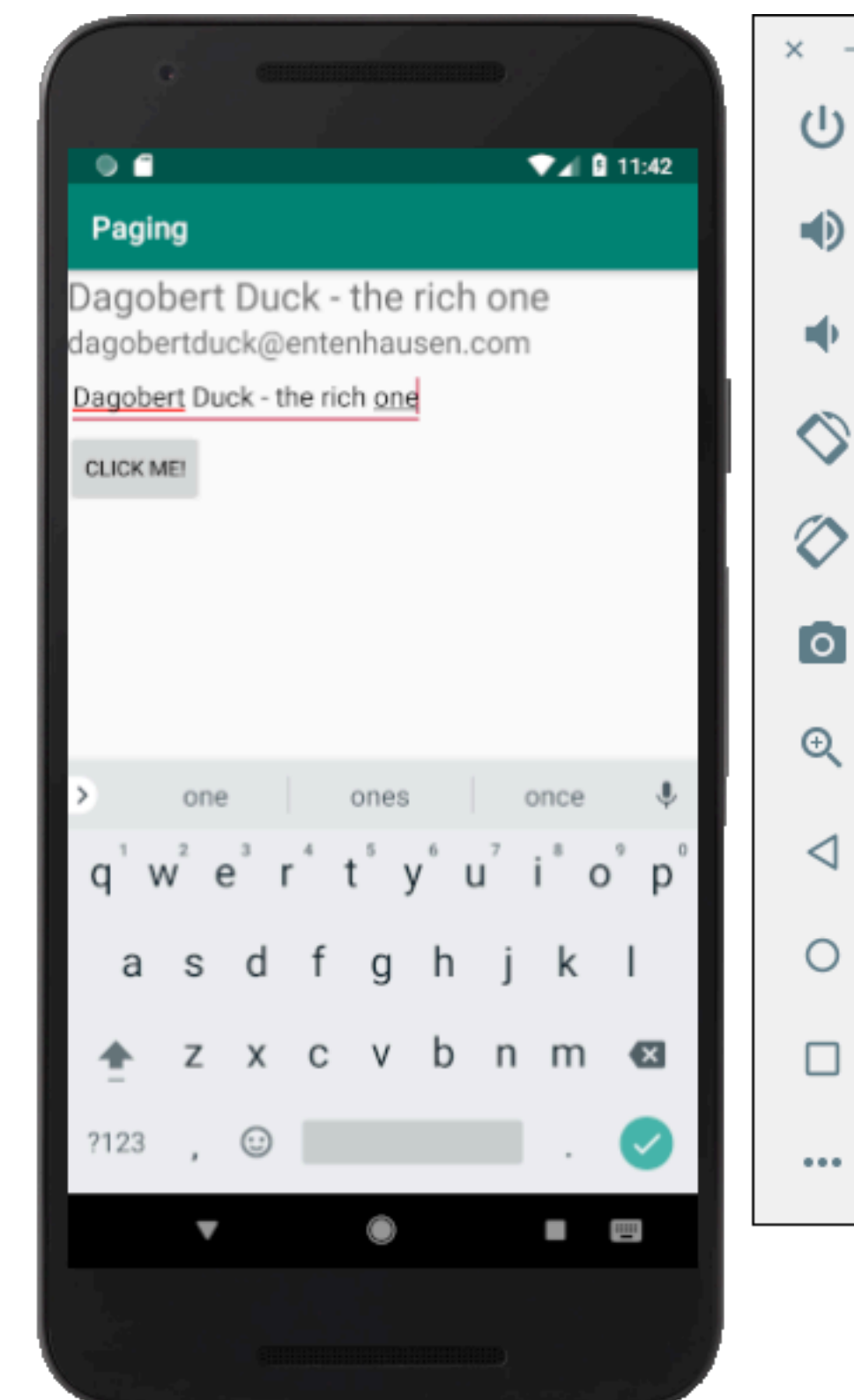
## <EditText

```

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/et_name"
android:inputType="text"
android:text="@={contact.name, default=Name}"/>

```

Das =-Zeichen bedeutet, daß der eingegebene Text akzeptiert und im Datenobjekt gespeichert wird



Ändert man nun den Text im „EditText“, so wird der Text auch in der TextView geändert

# Zusammenfassung Observable Data

- Man leitet die Entity- Klasse (Datenklasse) von BaseObservable ab
- In den Getter- und Setter-Methoden wird die Methode notifyPropertyChanged() aufgerufen
-

# EventHandler mit Parameter



# EventHandler mit Parameter

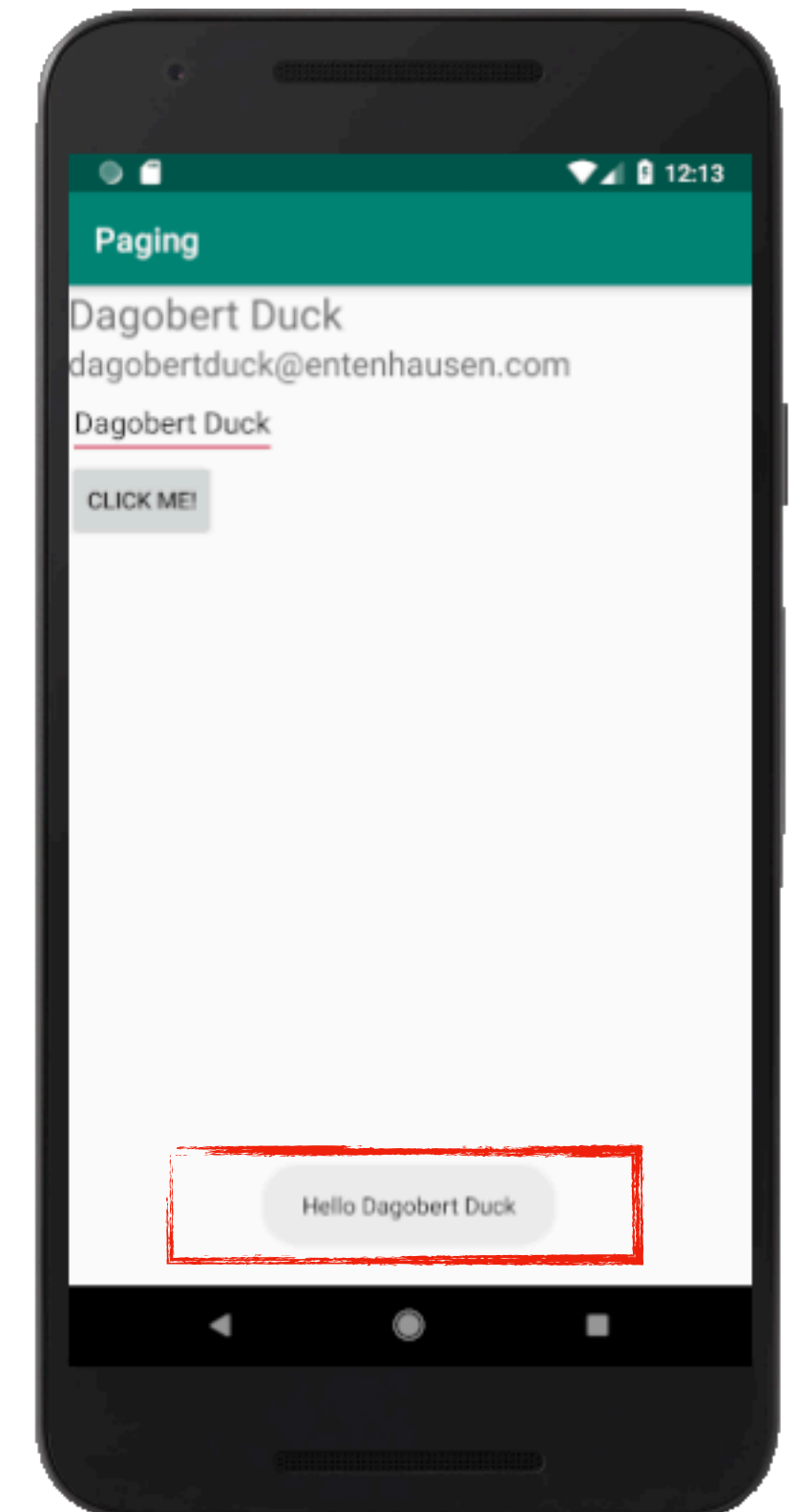
activity\_main.xml

```
<Button
    android:id="@+id/btn_click"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CLICK ME!"
    android:onClick="@{() -> handler.onButtonClick(contact.name) }"/>
```

EventHandler.kt

```
open class EventHandler(context: Context) {
    val myContext = context

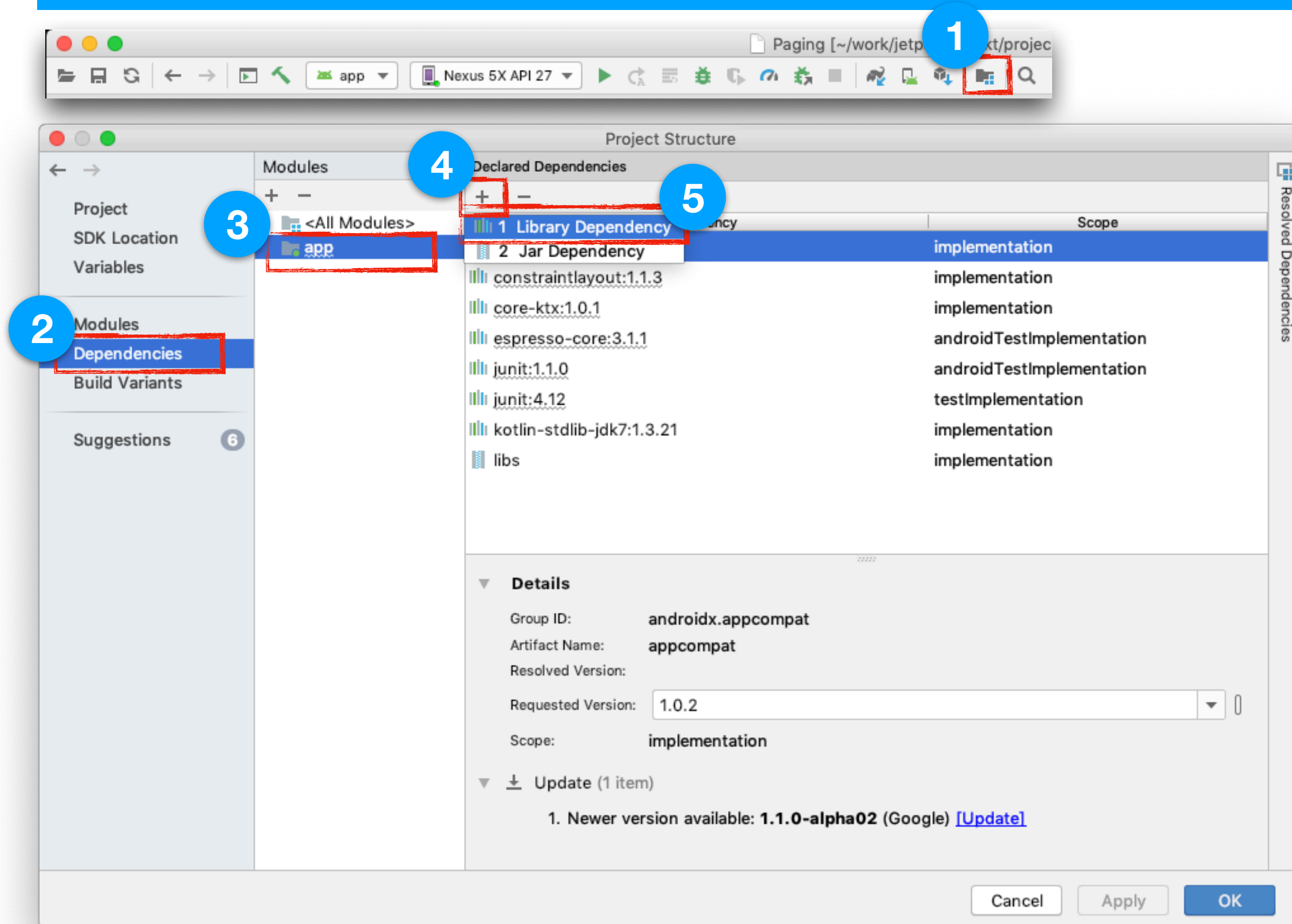
    fun onButtonClick(name: String) {
        Toast.makeText(myContext, "Hello $name", Toast.LENGTH_SHORT).show()
    }
}
```



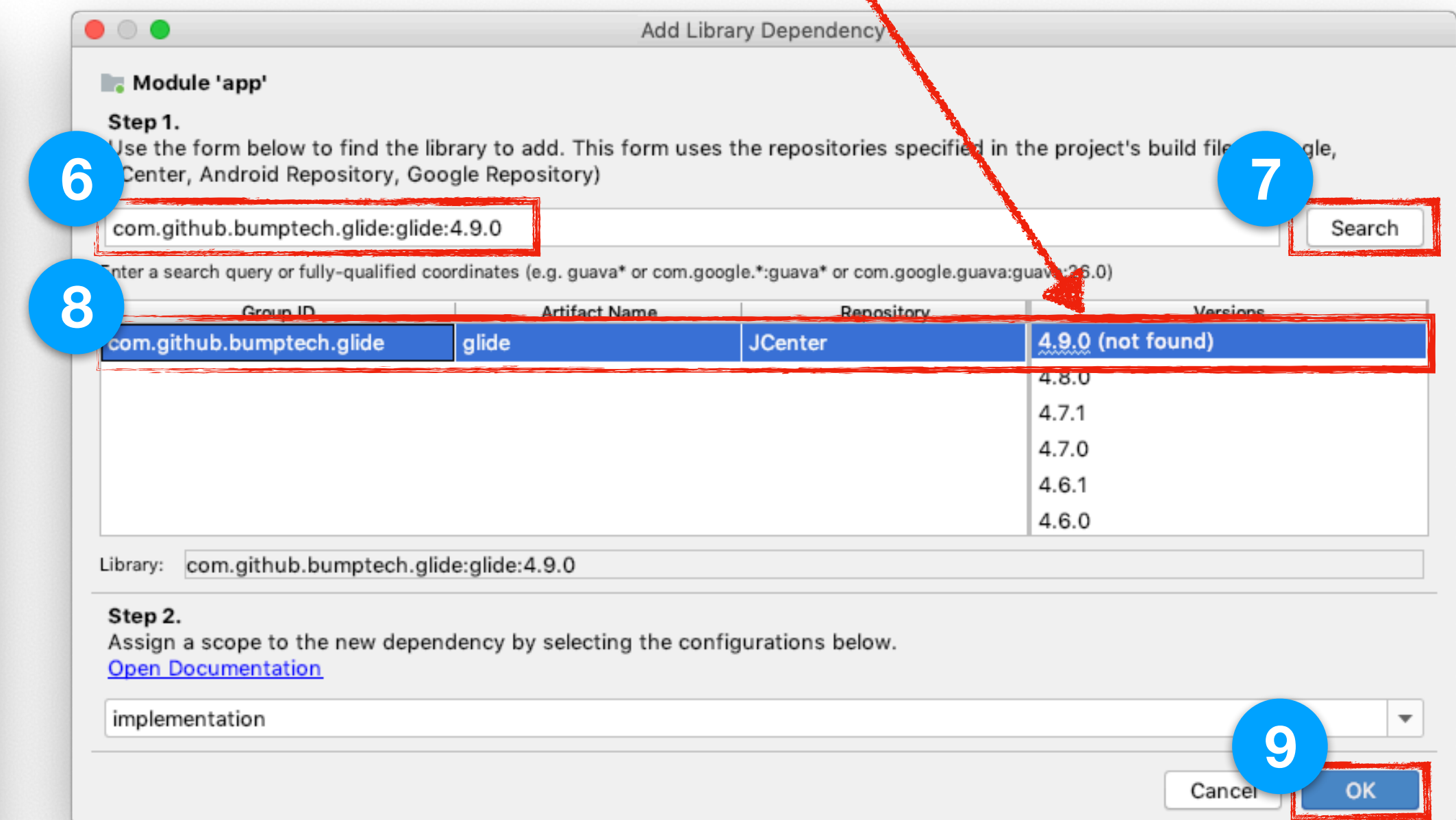
# Loading images from an URL with Glide

# Voraussetzungen

Man kann die Bibliotheken (Libraries) entweder in gradle.build eintragen, oder in der IDE

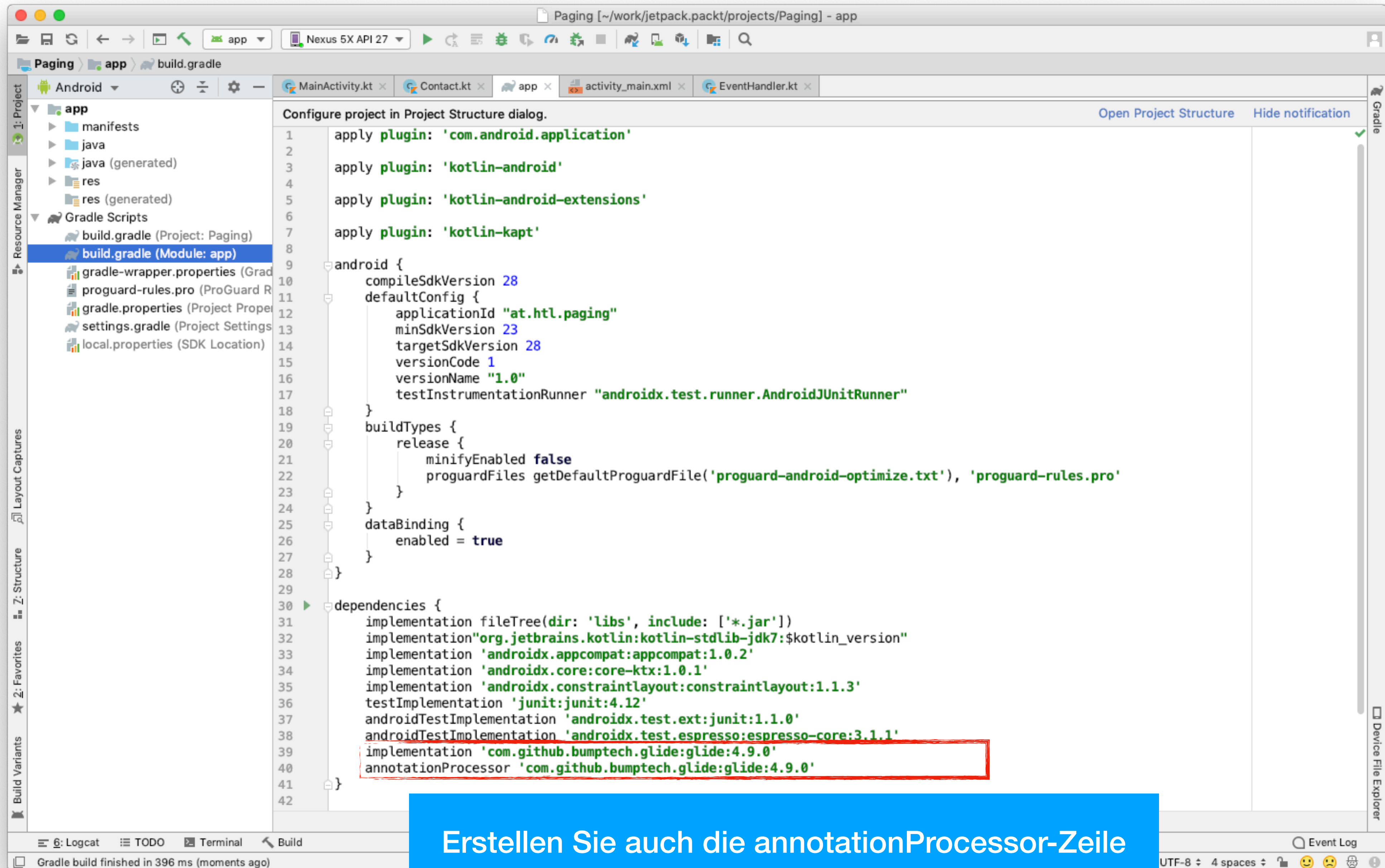


Die Library in der Version 4.9.0 wird nicht gefunden, da sie noch nicht im lokalen maven-repo enthalten ist

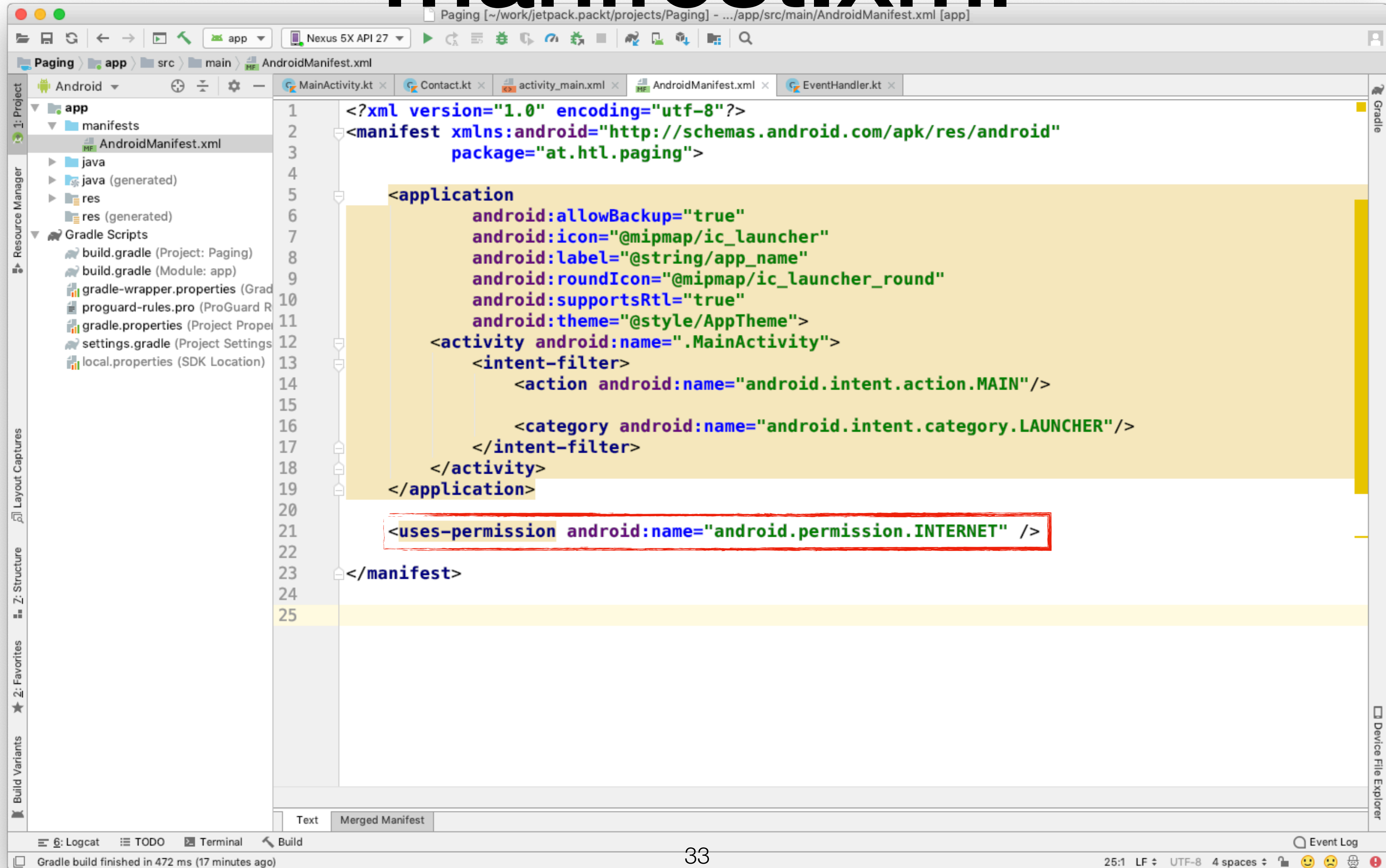


<https://github.com/bumptech/glide>





# manifest.xml





# activity\_main.xml

Die LinearLayouts werden verschachtelt. Dies ist keine optimale Lösung. Besser: ConstraintLayout

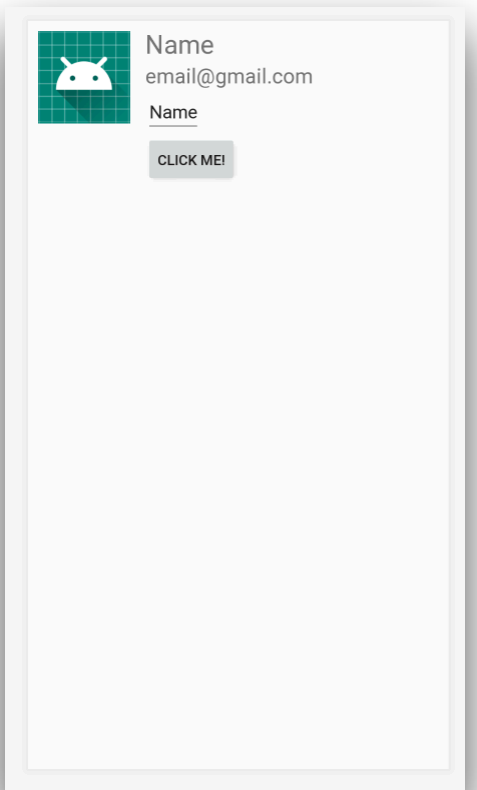
```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
```

```
    <data>
        <variable
            name="contact"
            type="at.htl.paging.Contact"/>
        <variable
            name="handler"
            type="at.htl.paging.EventHandler"/>
        <variable
            name="imageUrl"
            type="String"/>
    </data>
```

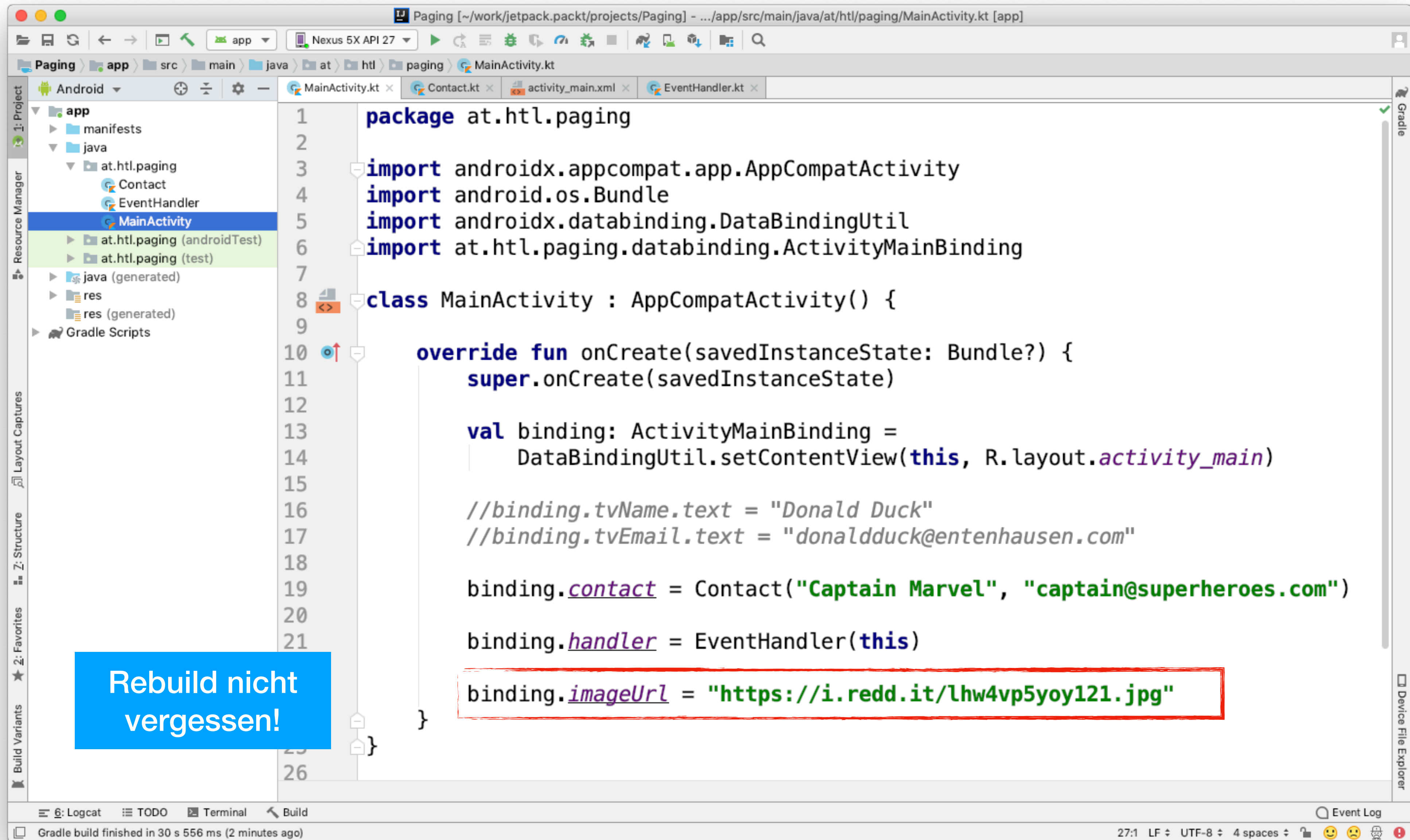
```
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
```

```
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp">
            <ImageView
                android:id="@+id/iv_profile_image"
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:padding="5dp"
                android:src="@mipmap/ic_launcher"/>
        </LinearLayout>
```

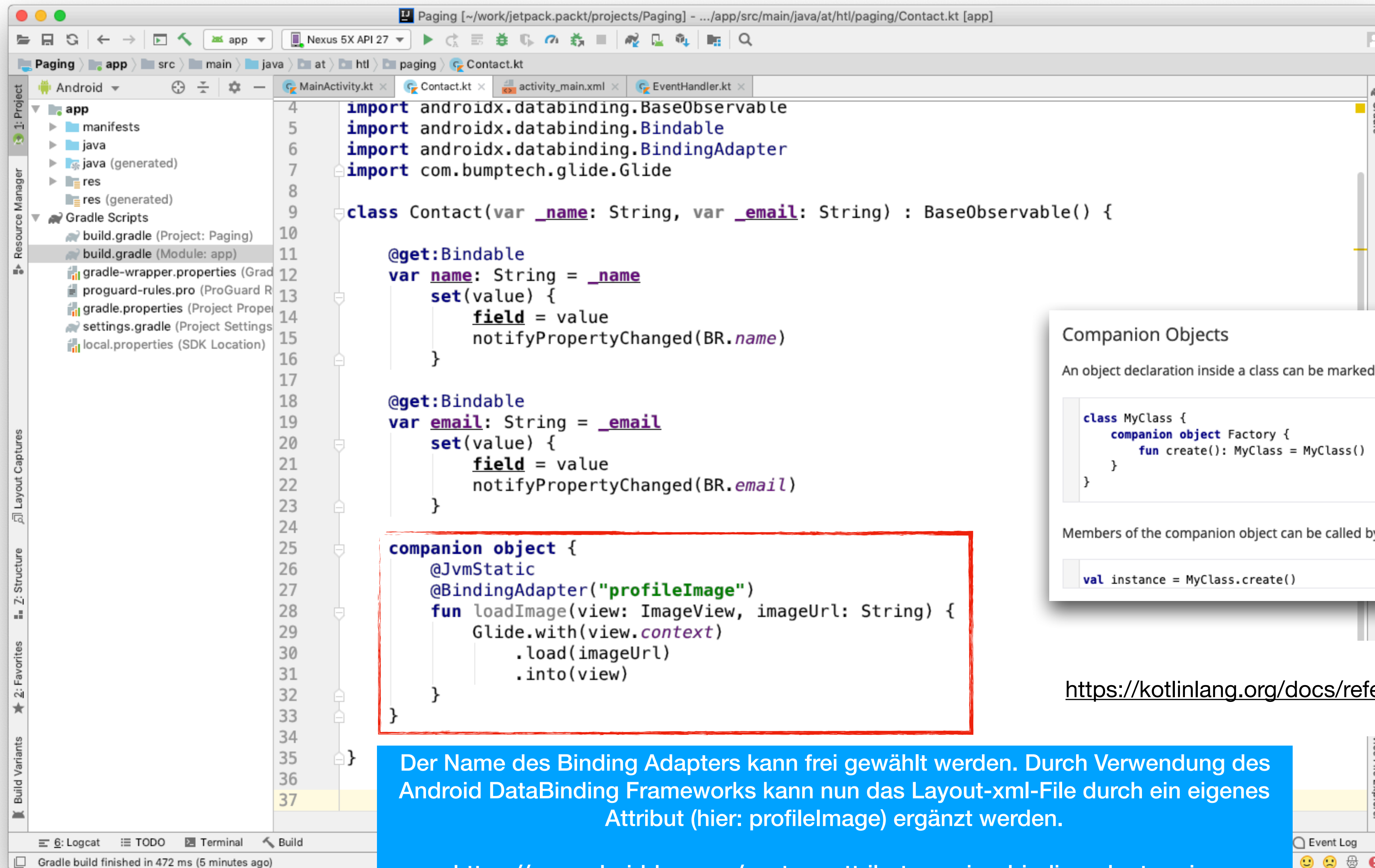
```
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:padding="5dp">
            <TextView
                android:id="@+id/tv_name"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@{contact.name, default=Name}"
                android:textSize="25sp"/>
            <TextView
                android:id="@+id/tv_email"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@{contact.email, default="email@gmail.com"}"
                android:textSize="20sp"/>
            <EditText
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/et_name"
                android:inputType="text"
                android:text="@={contact.name, default=Name}"/>
            <Button
                android:id="@+id/btn_click"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="CLICK ME!"
                android:onClick="@{() -> handler.onButtonClick(contact.name)}"/>
        </LinearLayout>
    </LinearLayout>
</layout>
```







<https://i.redd.it/lhw4vp5yoy121.jpg>



## Companion Objects

An object declaration inside a class can be marked with the `companion` keyword:

```
class MyClass {  
    companion object Factory {  
        fun create(): MyClass = MyClass()  
    }  
}
```

Members of the companion object can be called by using simply the class name as the qualifier:

```
val instance = MyClass.create()
```

<https://kotlinlang.org/docs/reference/object-declarations.html>

Der Name des Binding Adapters kann frei gewählt werden. Durch Verwendung des Android DataBinding Frameworks kann nun das Layout-xml-File durch ein eigenes Attribut (hier: profileImage) ergänzt werden.

<https://proandroiddev.com/custom-attributes-using-bindingadapters-in-kotlin-971ef8fcc259>



# activity\_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto">
```

```
<data>
    <variable
        name="contact"
        type="at.htl.paging.Contact"/>
    <variable
        name="handler"
        type="at.htl.paging.EventHandler"/>
    <variable
        name="imageUrl"
        type="String"/>
</data>
```

Den Namespace durch Alt-Enter halbautomatisch importieren

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp">

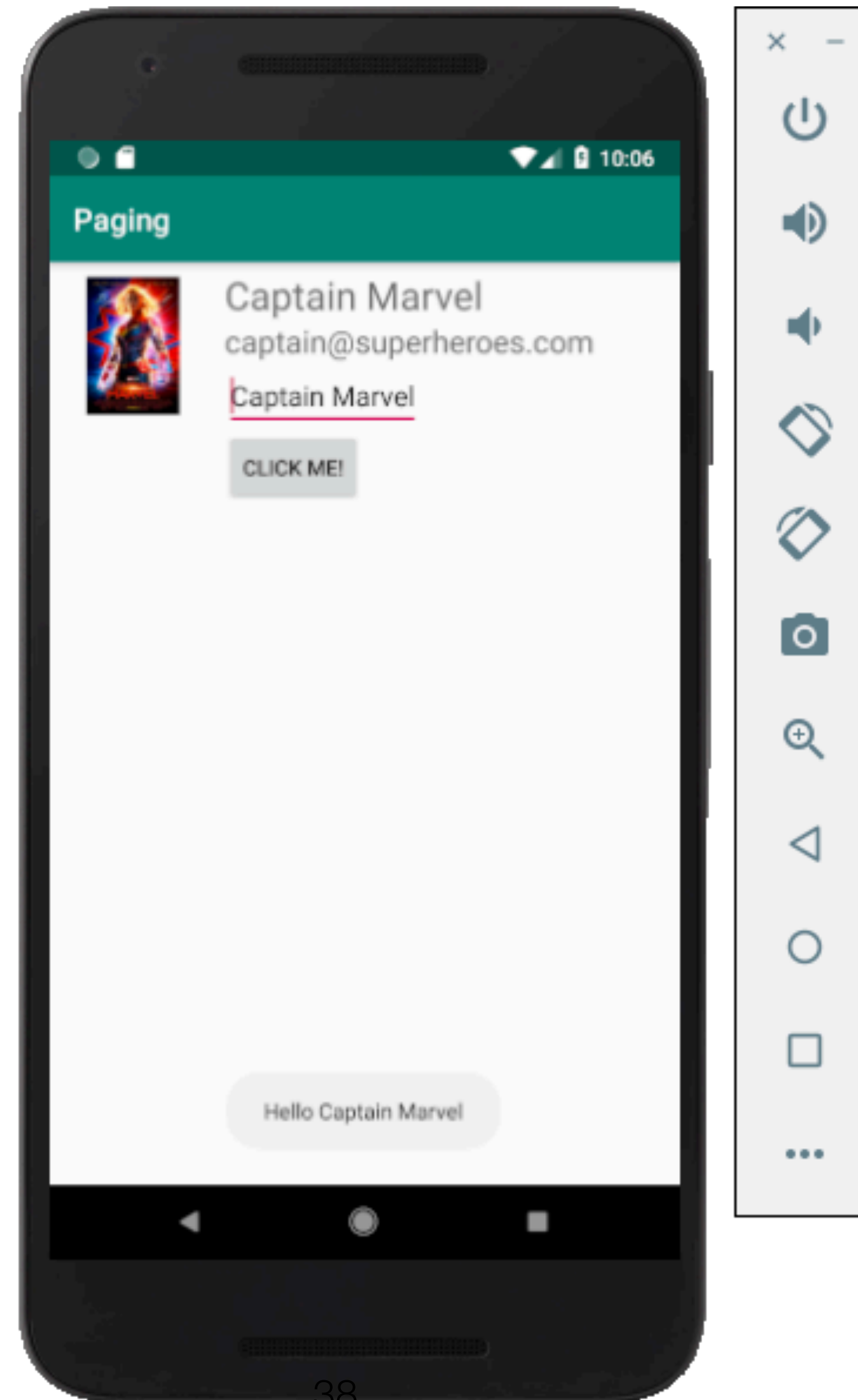
        <ImageView
            android:id="@+id/iv_profile_image"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:padding="5dp"
            app:profileImage="@{imageUrl}"
            android:src="@mipmap/ic_launcher"/>

    </LinearLayout>
```

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp">
    <TextView
        android:id="@+id/tv_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@{contact.name, default=Name}"
        android:textSize="25sp"/>
    <TextView
        android:id="@+id/tv_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@{contact.email, default="email@gmail.com"}"
        android:textSize="20sp"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/et_name"
        android:inputType="text"
        android:text="@={contact.name, default=Name}"/>
    <Button
        android:id="@+id/btn_click"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CLICK ME!"
        android:onClick="@{() -> handler.onButtonClick(contact.name)}"/>
</LinearLayout>
</layout>
```



# Testlauf







Noch  
Fragen?





# Quelle

<https://www.udemy.com/android-jetpack-architecture-components/>

 Categories  [Udemy for Business](#) [Become an instructor](#)  [Log In](#) [Sign Up](#)



Development > Mobile Apps > Android Game Development

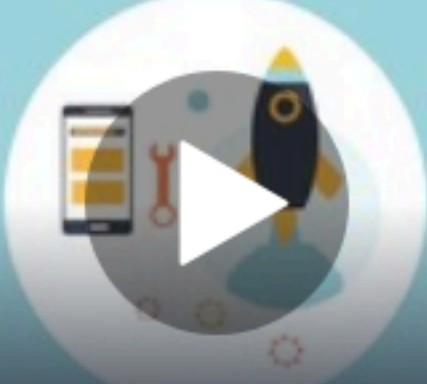
 Gift This Course  Wishlist

## Android Jetpack Architecture Components

Utilize Android Jetpack Architecture components to make your Android application development flexible and maintainable


NEW ★★★★★ 0.0 (0 ratings) 4 students enrolled

Created by Packt Publishing Last updated 2/2019  English  English [Auto-generated]



Preview this course

**€10.99** ~~€124.99~~ 91% off






 5 hours left at this price!

[Add to cart](#)

[Buy now](#)

30-Day Money-Back Guarantee

**Includes**

-  3 hours on-demand video
-  1 downloadable resource
-  Full lifetime access
-  Access on mobile and TV
-  Certificate of Completion

### What you'll learn

- ✓ Get introduced to Android architecture components
- ✓ Provide stability in your app by handling life cycles, view models, and live data
- ✓ Load data gradually and gracefully in RecyclerView by using the Paging library
- ✓ Explore how to perform CRUD operations in the Room database
- ✓ Use the Data Binding library to bind data to the UI
- ✓ Implement effective in-app navigation by using the Navigation architecture component
- ✓ Implement a local database to store structured data by using the Room database
- ✓ Schedule tasks asynchronously by using Work Manager



# Quellen

- <https://codelabs.developers.google.com/codelabs/android-databinding/index.html#0>
- <https://codelabs.developers.google.com/?cat=Android>
- <https://www.vogella.com/tutorials/AndroidDatabinding/article.html>
- <https://proandroiddev.com/exploring-android-data-binding-library-ff467171c756>
-