

# 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

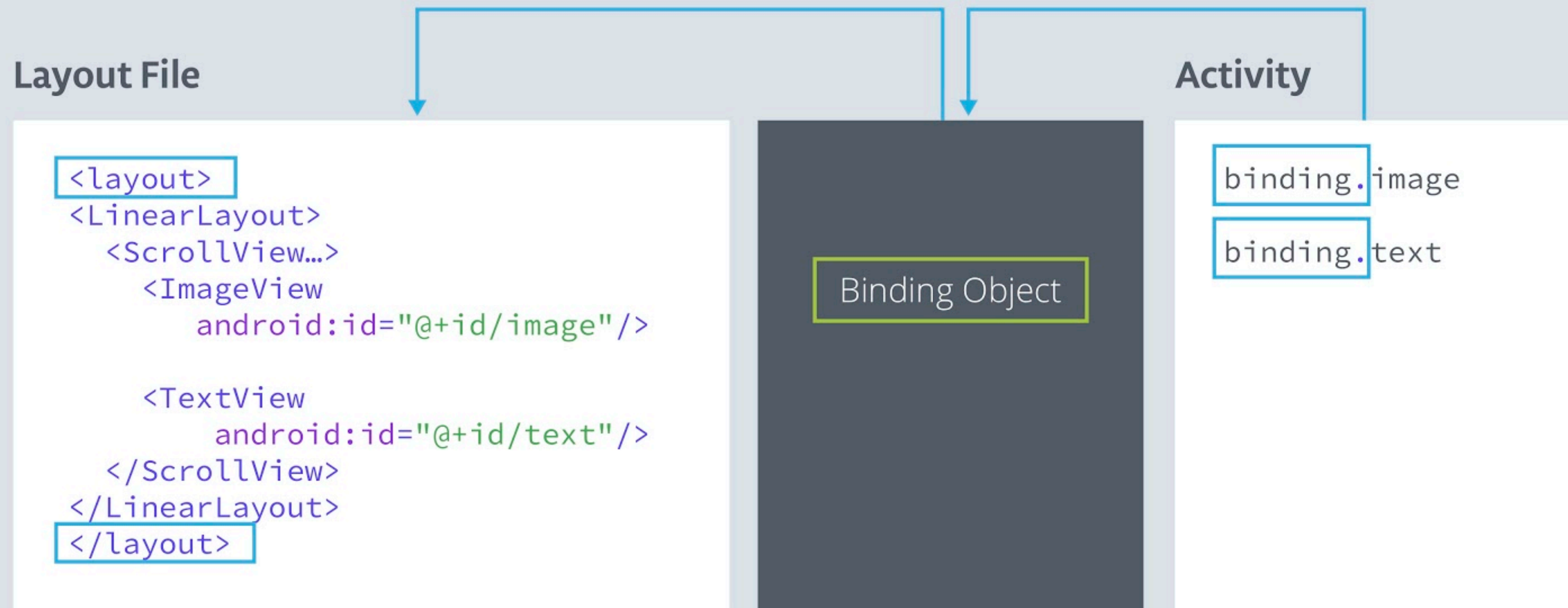
---

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.

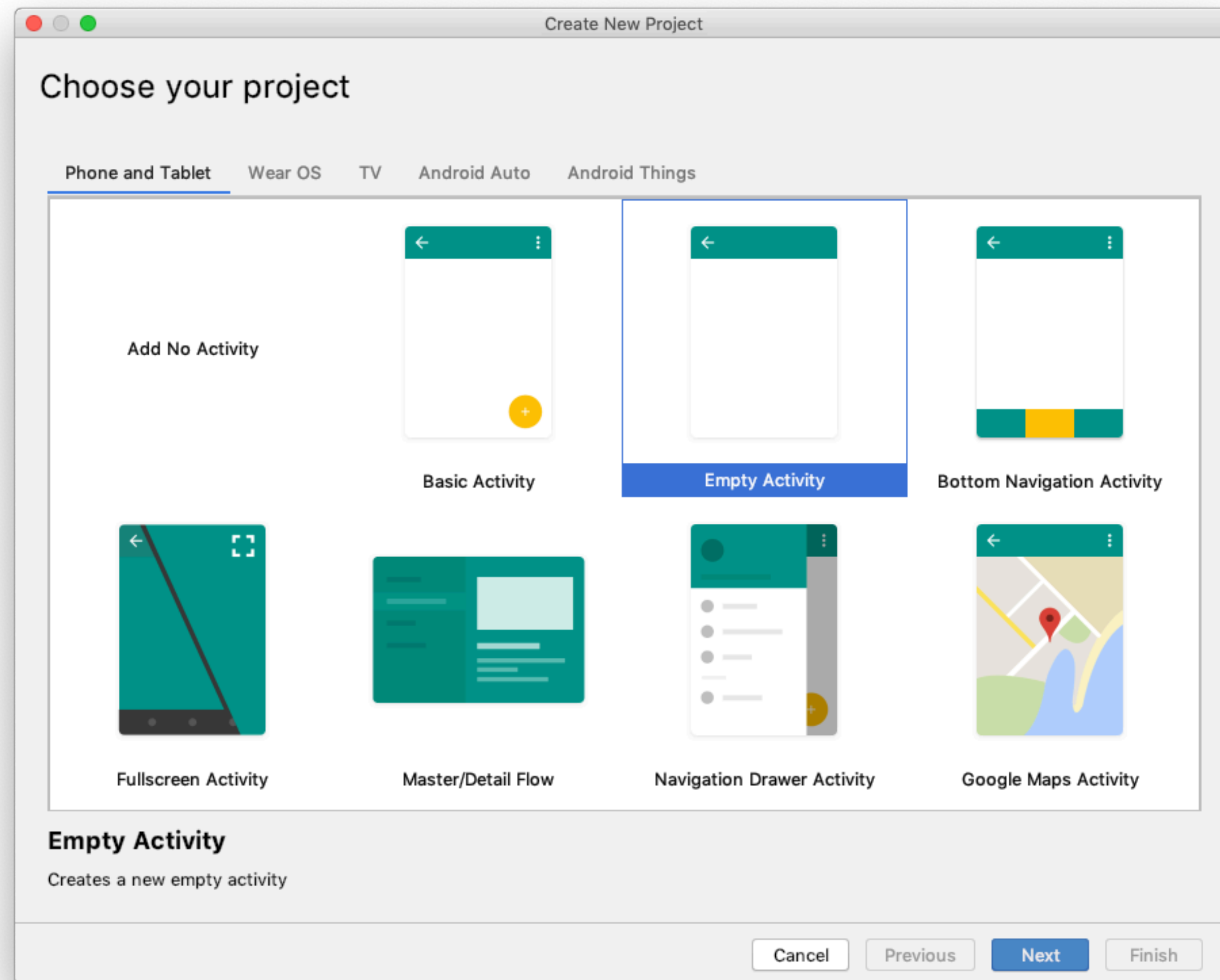
# With Data Binding



# 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“

The screenshot displays an IDE window for the 'Paging' project. The main editor shows the following Kotlin code for MainActivity.kt:

```
1 package at.htl.paging
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
13
```

The left sidebar shows the project structure, with the 'MainActivity' file selected. The bottom panel shows the Build output:

- Build: Sync x
- ❄️ Paging: syncing... Running for 3 m 20 s
  - ✅ Starting Gradle Daemon 1 s
  - ❄️ Run build /Users/stuetz/work/jetpack.packt/projects/Paging Running for 3 m 16 s
    - ✅ Load build 1 s 546 ms
    - ❄️ Configure build Running for 3 m 14 s

The status bar at the bottom indicates 'Gradle sync started with single-variant sync (3 minutes ago)' and '2 processes running... 13:1 LF UTF-8 4 spaces'.

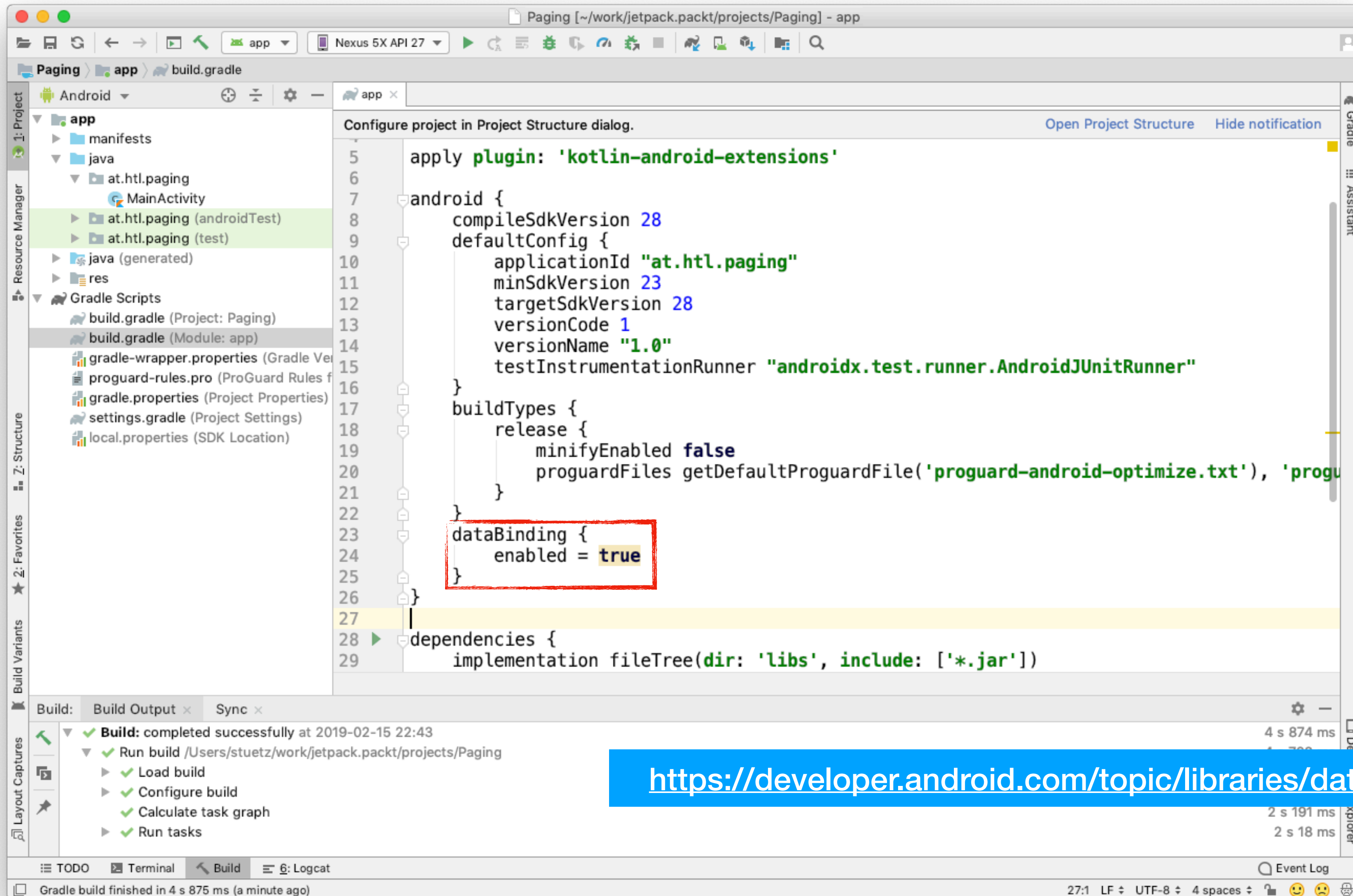


The image shows the 'Preferences for New Projects' dialog in Android Studio. The 'System Settings' > 'Android SDK' section is active, with the 'SDK Tools' sub-tab selected. A table lists various SDK developer tools and their installation status. The 'Support Repository' section is expanded, and the 'Android Support Repository' (version 47.0.0) is highlighted with a red box. A blue callout box at the bottom of the dialog contains the text: 'kontrollieren, ob Support Repository installiert ist'. The dialog also shows the 'Android SDK Location' as '/opt/android-sdk-macosx' and the 'SDK Platforms' and 'SDK Update Sites' tabs. The background shows the Android Studio interface with a project named 'Paging' and a Gradle sync task in progress.

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
<b>Support Repository</b>		
<input checked="" type="checkbox"/> ConstraintLayout for Android		Installed
<input checked="" type="checkbox"/> Solver for ConstraintLayout		Installed
<input checked="" type="checkbox"/> <b>Android Support Repository</b>	<b>47.0.0</b>	<b>Installed</b>
<input checked="" type="checkbox"/> Google Repository	58	Installed

kontrollieren, ob Support Repository installiert ist

# dataBinding in build.xml eintragen



The screenshot shows an IDE window titled "Paging [~/work/jetpack.packt/projects/Paging] - app". The main editor displays the build.gradle file with the following code:

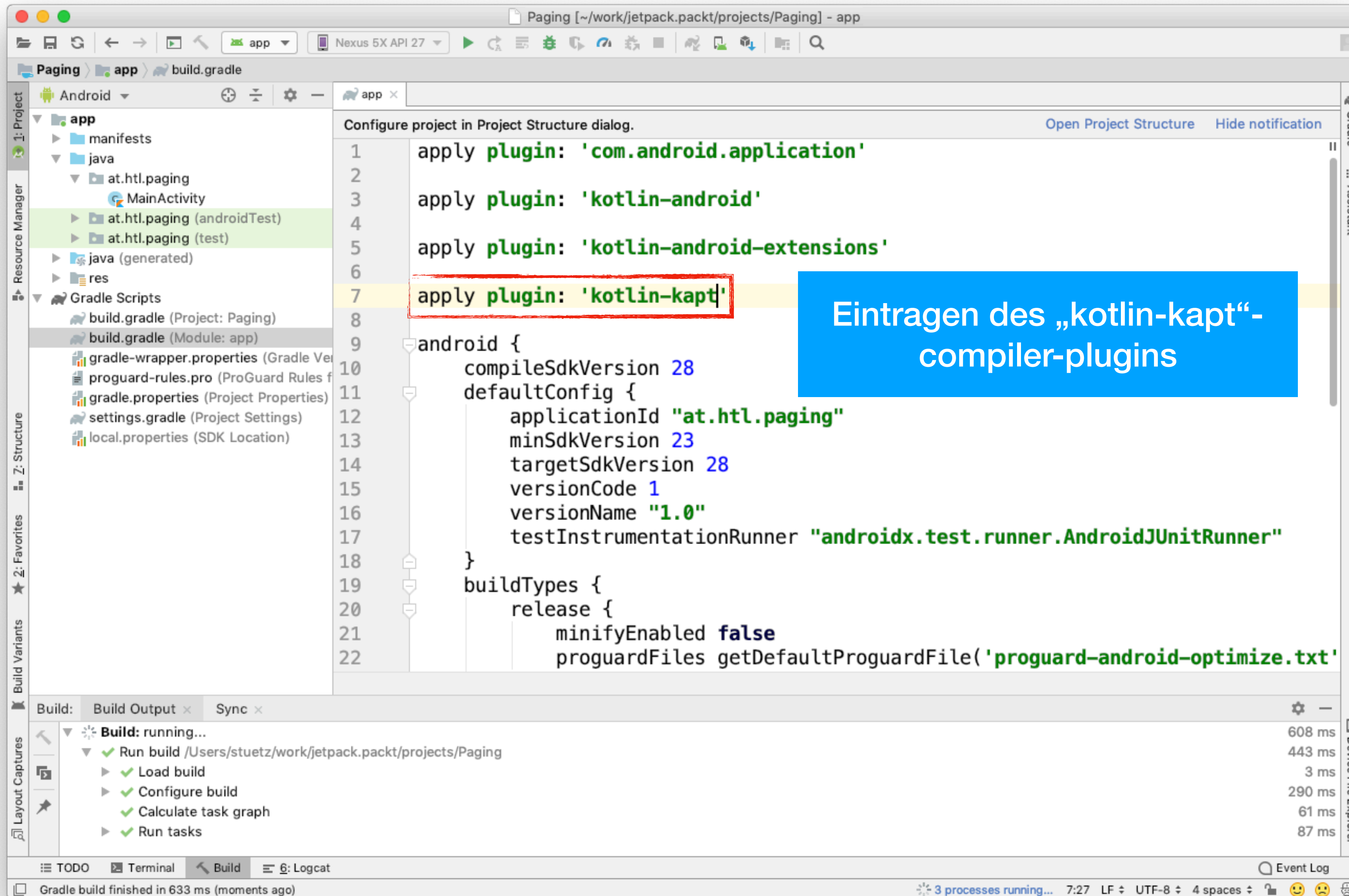
```
5  apply plugin: 'kotlin-android-extensions'
6
7  android {
8      compileSdkVersion 28
9      defaultConfig {
10         applicationId "at.htl.paging"
11         minSdkVersion 23
12         targetSdkVersion 28
13         versionCode 1
14         versionName "1.0"
15         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
16     }
17     buildTypes {
18         release {
19             minifyEnabled false
20             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
21         }
22     }
23     dataBinding {
24         enabled = true
25     }
26 }
27
28 dependencies {
29     implementation fileTree(dir: 'libs', include: ['*.jar'])
30 }
```

The `dataBinding { enabled = true }` block is highlighted with a red box. The IDE interface includes a Project view on the left, a Build Output window at the bottom, and a status bar at the bottom showing "Gradle build finished in 4 s 875 ms (a minute ago)".

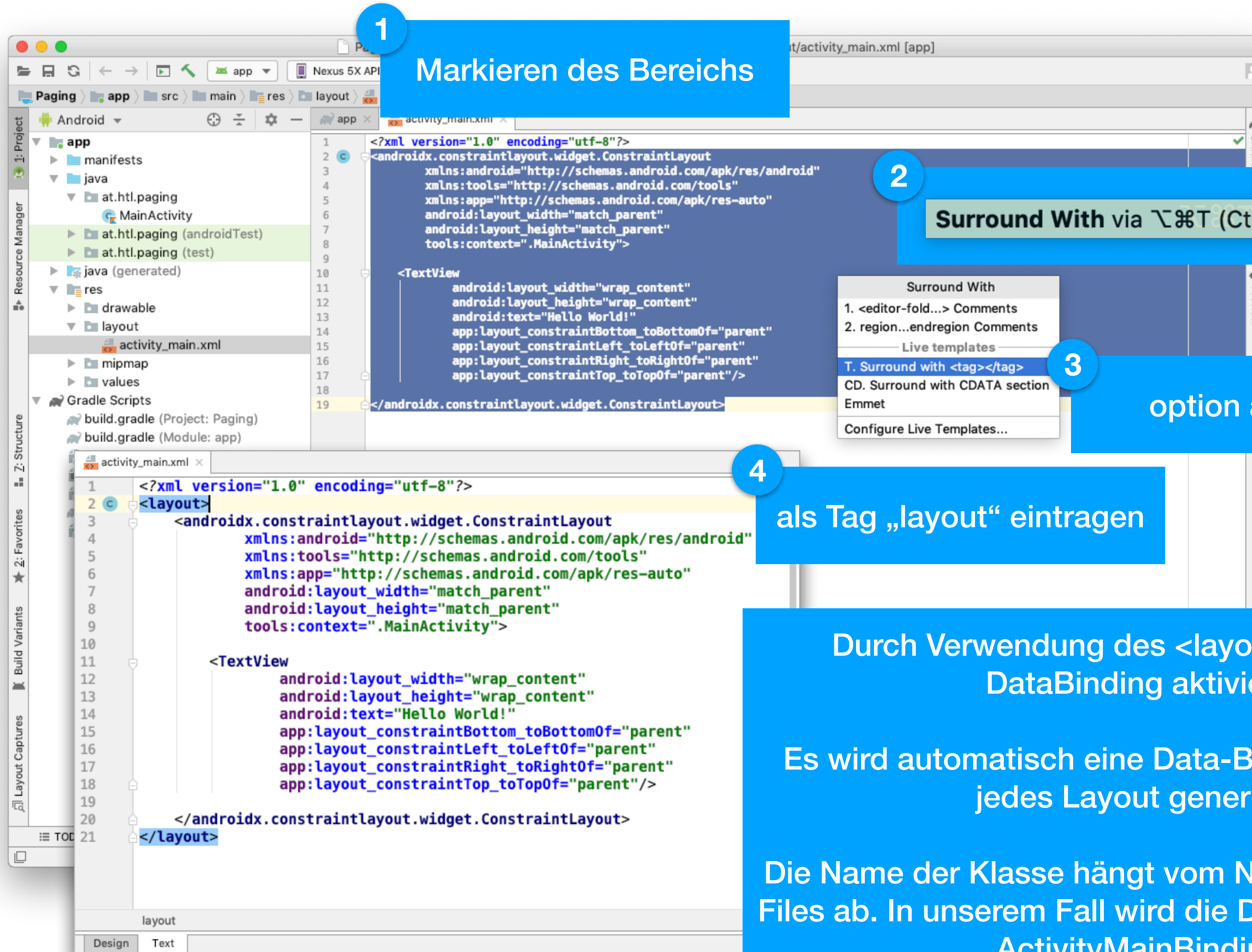
<https://developer.android.com/topic/libraries/data-binding/start>

# Layouts and Binding Expressions





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



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

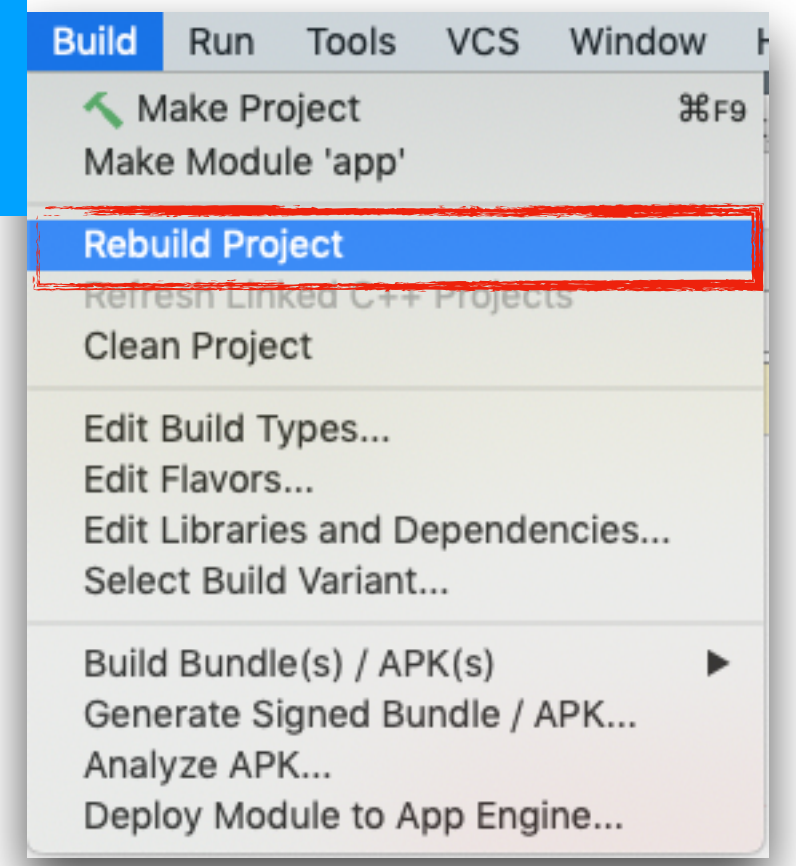
Es wird automatisch eine Data-Binding-Klasse für jedes Layout generiert.

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

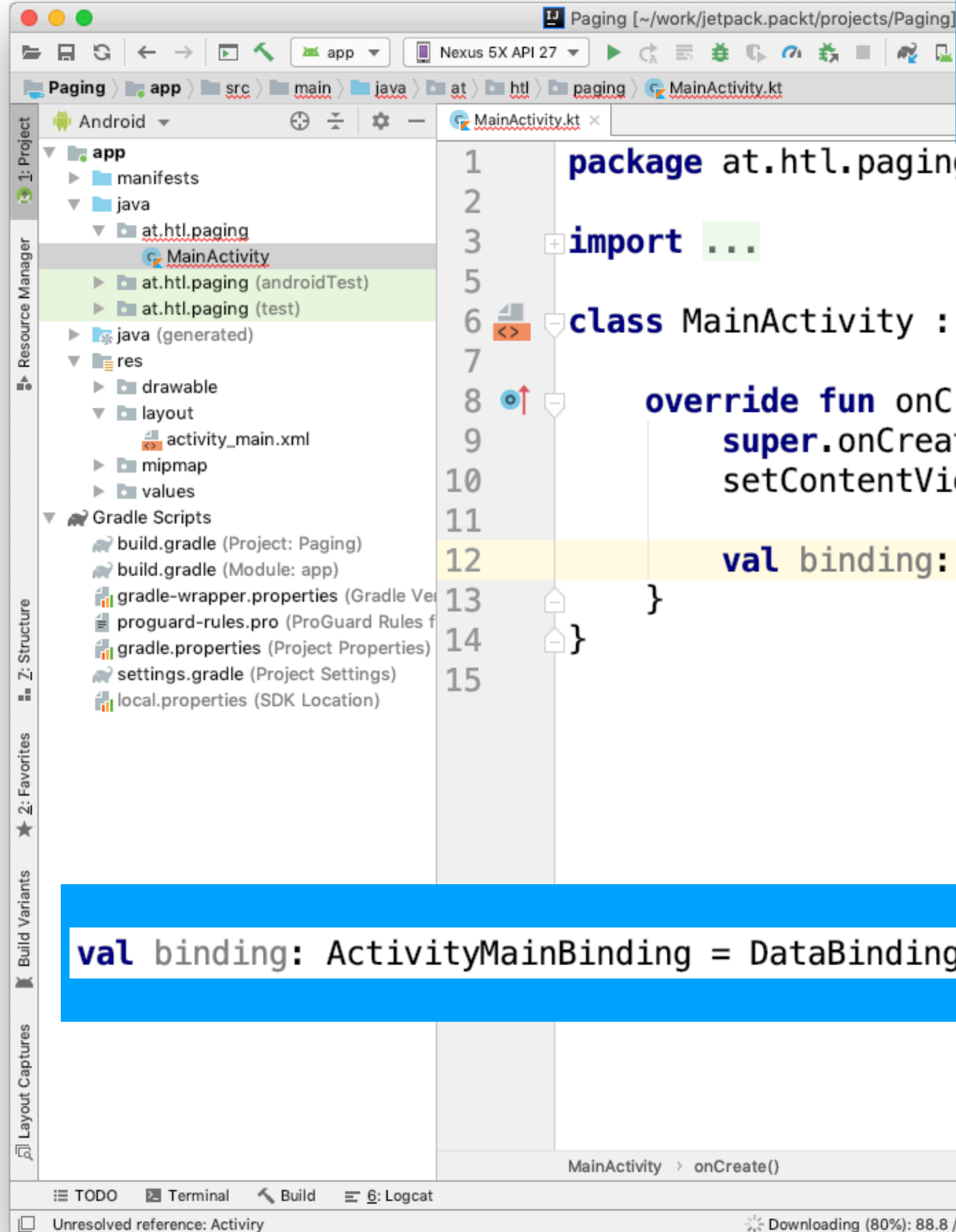
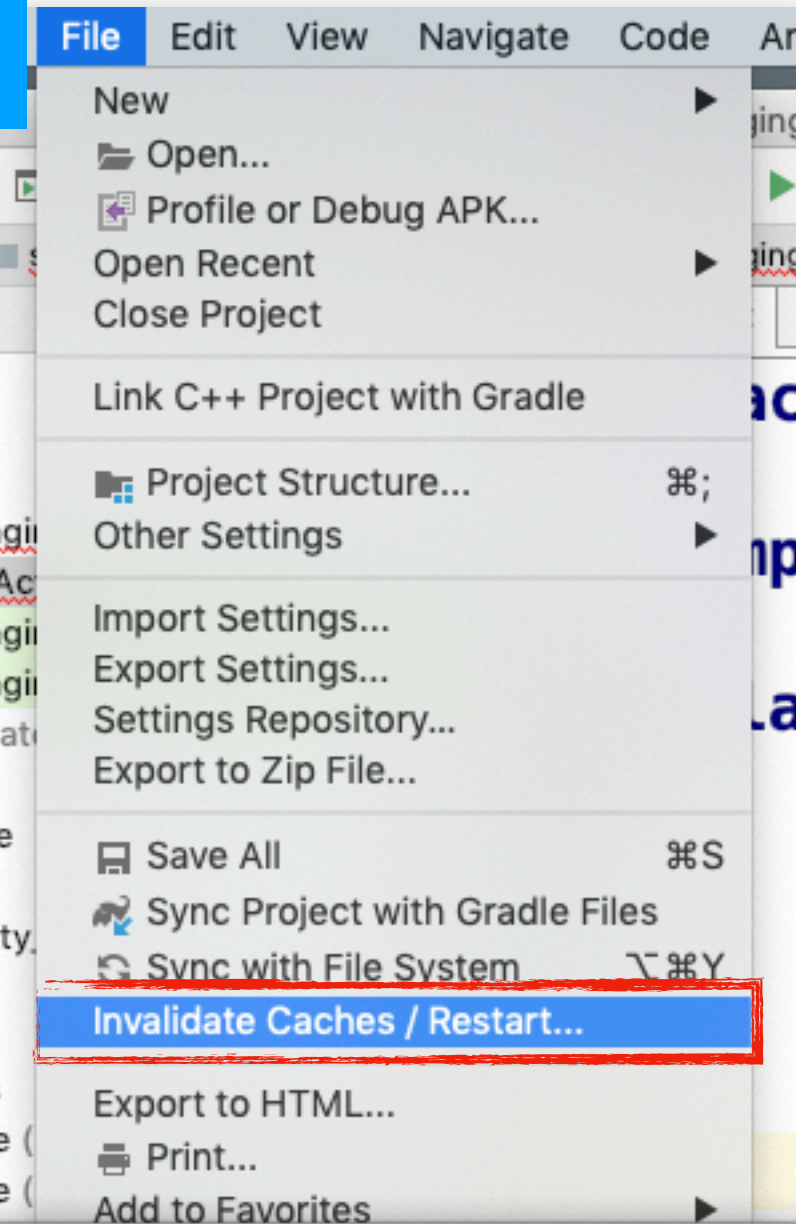


Falls das AutoComplete noch nicht funktioniert, kann man folgendes machen

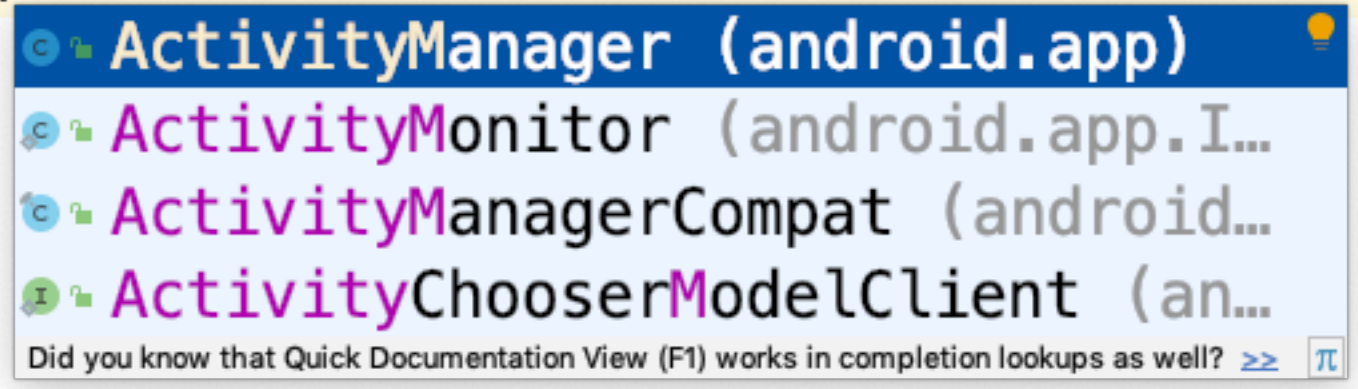
Maßnahme 1: Rebuild



Maßnahme 2: Invalidate Caches



```
1 package at.htl.paging
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11
12        val binding: ActivityM
13    }
14 }
15
```



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





The screenshot shows an IDE window for a project named "Paging". The main editor displays the following Kotlin code for MainActivity.kt:

```
1 package at.htl.paging
2
3 import ...
7
8 class MainActivity : AppCompatActivity() {
9
10 override fun onCreate(savedInstanceState: Bundle?) {
11     super.onCreate(savedInstanceState)
12     setContentView(...)
13     val binding: ActivityMainBinding =
14         DataBindingUtil.setContentView(this, R.layout.activity_main)
15 }
16 }
17
18
```

A blue callout box on the right side of the editor contains the text: **setContentView(...) wurde gelöscht**. This callout is positioned over the line of code that has been removed from the original code.

The bottom panel of the IDE shows the build output:

```
Build: Build Output x Sync x
Build: completed successfully at 2019-02-16 09:36 with 1 warning 14 s 59 ms
  Run build /Users/stuetz/work/jetpack.packt/projects/Paging 13 s 314 ms
    Load build 4 ms
    Configure build 718 ms
    Calculate task graph 551 ms
    Run tasks 11 s 893 ms
  Kotlin compiler: (1 warning)
```

The status bar at the bottom indicates "Gradle build finished in 14 s 66 ms (today 09:36)".

# Erstellen von String Ressourcen

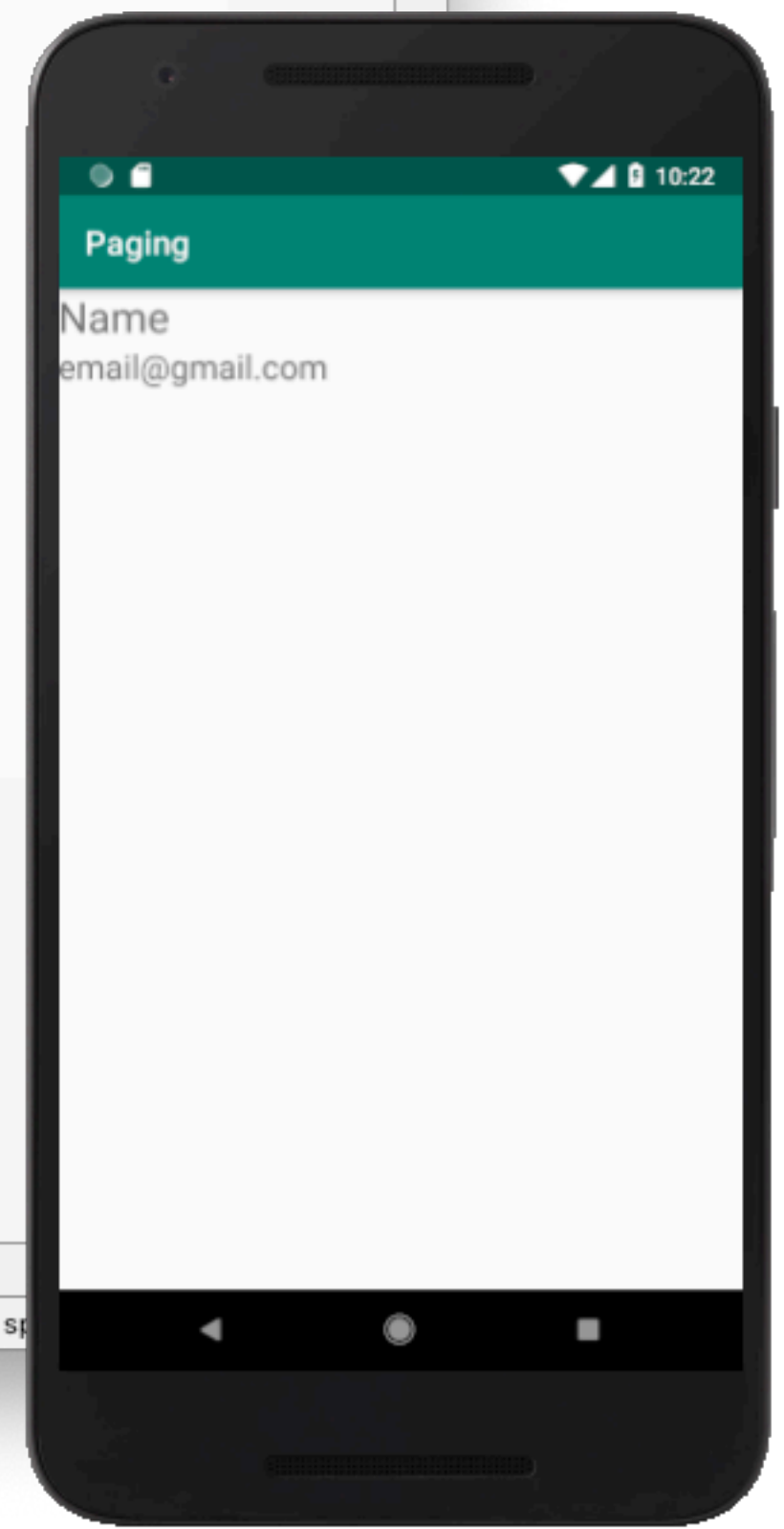
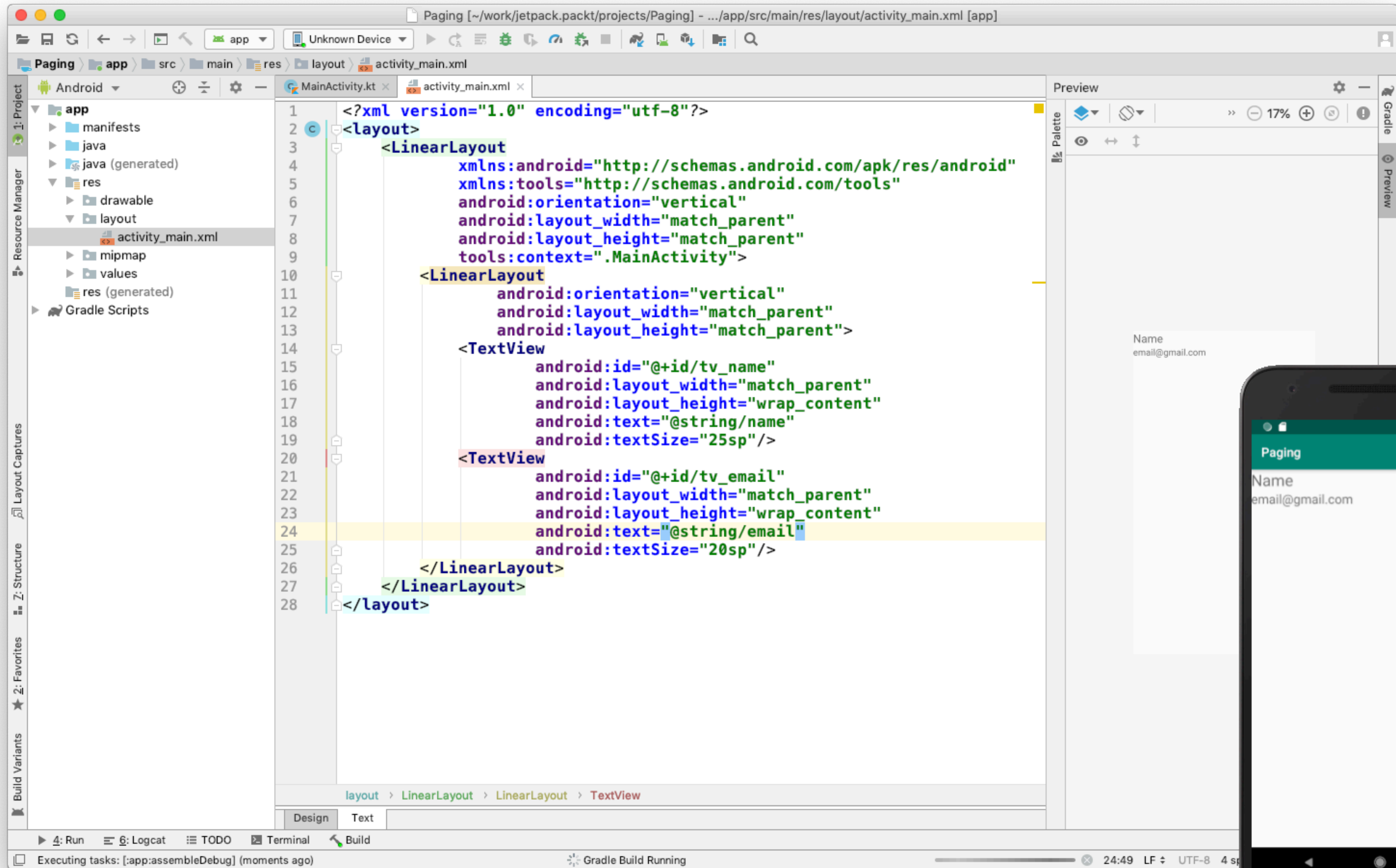
The screenshot shows the Android Studio interface with the following components:

- Project View:** Shows the project structure with folders for manifests, java, res (drawable, layout), mipmap, values, and res (generated).
- Palette:** Lists various UI widgets like TextView, EditText, etc.
- Design View:** Shows a preview of a TextView widget.
- Attributes Panel:** Shows the configuration for the selected TextView, including id, text, and textSize.
- Resources Dialog:** A dialog box for adding new resources, with a search bar and a list of existing resources.
- New String Value Resource Dialog:** A dialog box for creating a new string resource, with fields for Resource name, Resource value, Source set, and File name.

Attributes panel for TextView with id tv\_name. The text attribute is set to @string/name and the textSize is 25sp.

Attributes panel for TextView with id tv\_email. The text attribute is set to email@gmail.com and the textSize is 20sp.







The screenshot shows an IDE window for a project named 'Paging'. The main editor displays the following Kotlin code for MainActivity.kt:

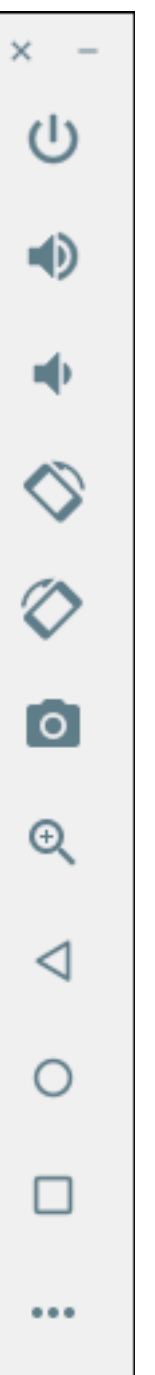
```
1 package at.htl.paging
2
3 import ...
4
5 class MainActivity : AppCompatActivity() {
6
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9
10        val binding: ActivityMainBinding =
11            DataBindingUtil.setContentView(this, R.layout.activity_main)
12
13        binding.tvName.text = "Donald Duck"
14        binding.tvEmail.text = "donaldduck@entenhausen.com"
15    }
16 }
17
18
19
20
21
```

A blue callout box points to the code on line 13, containing the text: "Beachte, daß im Code tv\_name automatisch zu tvName geändert wurde".

A dialog box titled "New Kotlin File/Class" is open, showing "Name: Contact" and "Kind: Class".

A blue callout box at the bottom left contains the text: "Erstellen einer neuen Klasse „Contact“".

The IDE interface includes a Project view on the left, a Resource Manager, and a bottom toolbar with Run, Logcat, TODO, Terminal, Build, and Profiler buttons. A status bar at the bottom shows "Install successful (3 minutes ago)" and "21:1 LF UTF-8".





```
1 package at.htl.paging
```

```
2  
3 class Contact (var name: String, var email: String)
```

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

```
2 <layout>
```

```
3 <data>
```

```
4 <variable
```

```
5 name="contact"
```

```
6 type="at.htl.paging.Contact"/>
```

```
7 </data>
```

```
8 <LinearLayout
```

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

```
10 xmlns:tools="http://schemas.android.com/tools"
```

```
11 android:orientation="vertical"
```

```
12 android:layout_width="match_parent"
```

```
13 android:layout_height="match_parent"
```

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

```
15 <LinearLayout
```

```
16 android:orientation="vertical"
```

```
17 android:layout_width="match_parent"
```

```
18 android:layout_height="match_parent">
```

```
19 <TextView
```

```
20 android:id="@+id/tv_name"
```

```
21 android:layout_width="match_parent"
```

```
22 android:layout_height="wrap_content"
```

```
23 android:text="@{contact.name, default=Name}"
```

```
24 android:textSize="25sp"/>
```

```
25 <TextView
```

```
26 android:id="@+id/tv_email"
```

```
27 android:layout_width="match_parent"
```

```
28 android:layout_height="wrap_content"
```

```
29 android:text="@{contact.email, default="email@gmail.com}"
```

```
30 android:textSize="20sp"/>
```

```
31 </LinearLayout>
```

```
32 </LinearLayout>
```

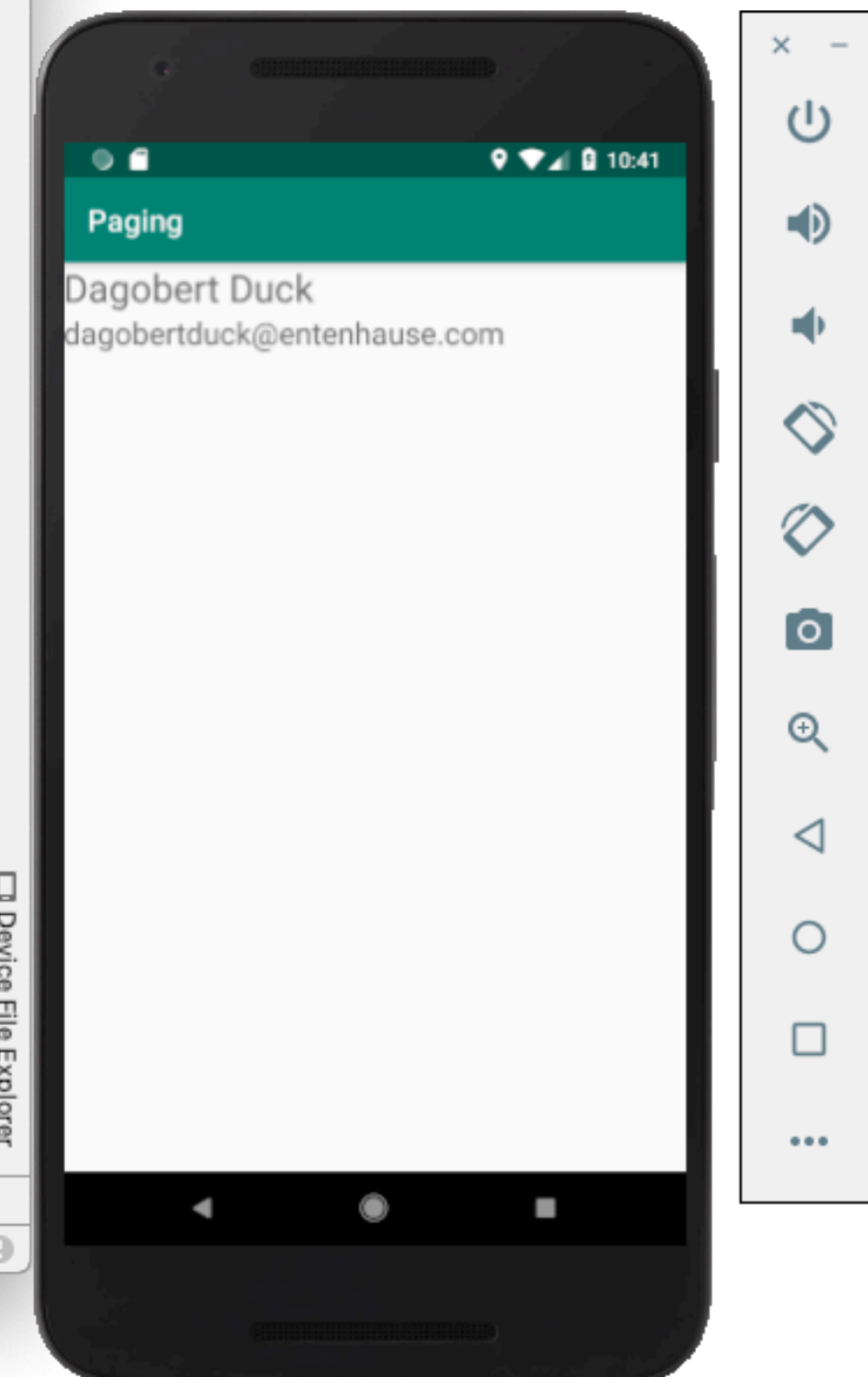
```
33 </layout>
```

Beachten Sie die Verwendung der einfachen Hochkommas



```
1 package at.htl.paging
2
3 import ...
7
8 class MainActivity : AppCompatActivity() {
9
10 override fun onCreate(savedInstanceState: Bundle?) {
11     super.onCreate(savedInstanceState)
12
13     val binding: ActivityMainBinding =
14         DataBindingUtil.setContentview(this, R.layout.activity_main)
15
16     //binding.tvName.text = "Donald Duck"
17     //binding.tvEmail.text = "donaldduck@entenhausen.com"
18
19     binding.contact = Contact("Dagobert Duck", "dagobertduck@entenhouse.com")
20 }
21 }
22
23
```

Wir verwenden einen anderen Namen, um die Veränderung im Emulator ansehen zu können



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

Paging > app > src > main > java > at > htl > paging > EventHandler.kt

MainActivity.kt x Contact.kt x activity\_main.xml x EventHandler.kt x

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

4: Run 6: Logcat TODO Terminal Build Profiler

Event Log

Install successful (12 minutes ago) 16:1 LF UTF-8 4 spaces

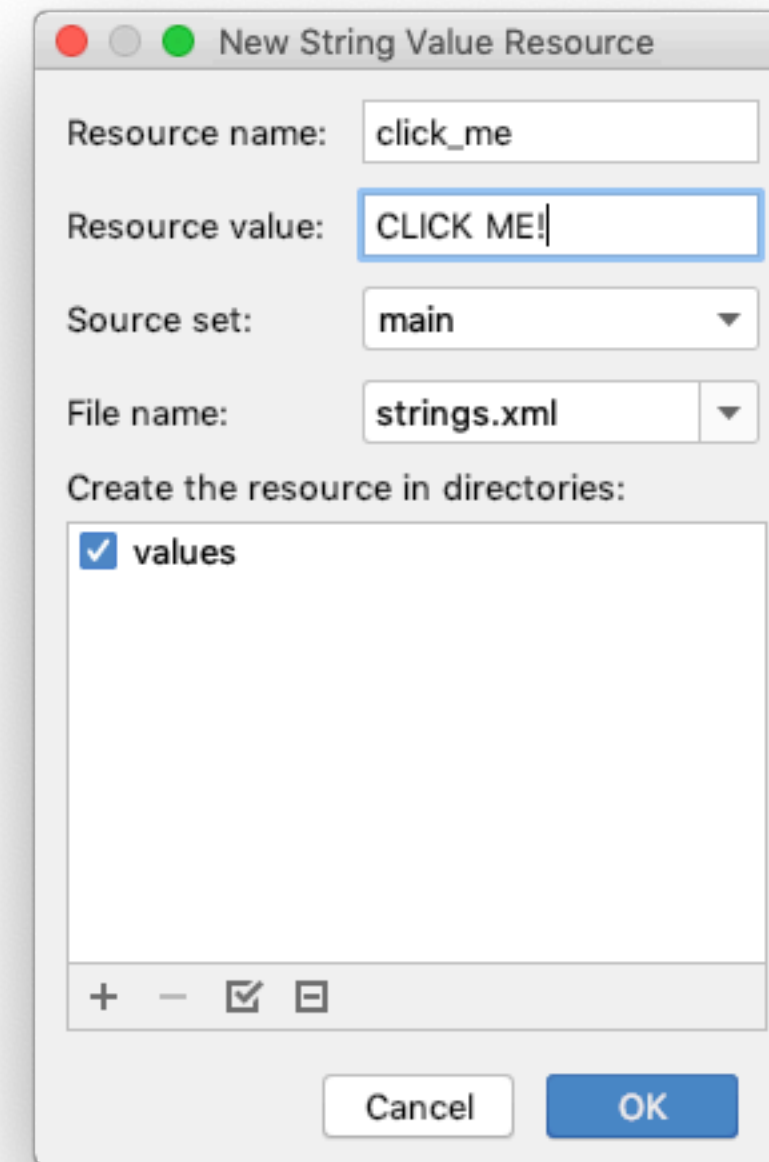


```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
```

```
<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>
```

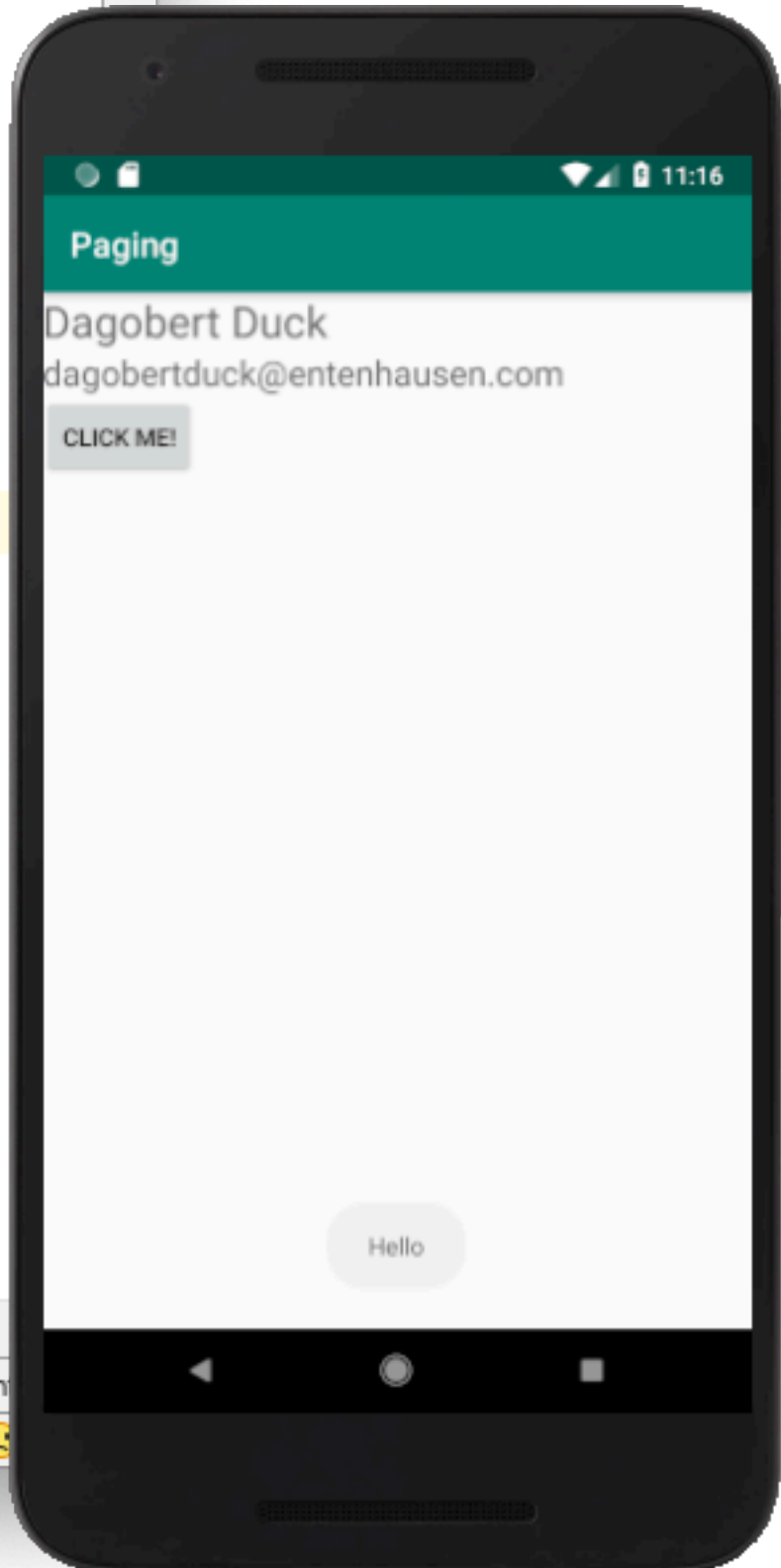
Der EventHandler wird als Variable eingetragen



Ebenso die onClick-Methode

```
1 package at.htl.paging
2
3 import ...
4
5
6
7
8 class MainActivity : AppCompatActivity() {
9
10
11
12
13 val binding: ActivityMainBinding =
14     DataBindingUtil.setContentview(this, R.layout.activity_main)
15
16 //binding.tvName.text = "Donald Duck"
17 //binding.tvEmail.text = "donaldduck@entenhausen.com"
18
19 binding.contact = Contact("Dagobert Duck", "dagobertduck@entenhausen.com")
20
21 binding.handler = EventHandler(this)
22 }
23
24
25
```

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

The screenshot shows an IDE window for a project named 'Paging'. The main editor displays the following Kotlin code for `Contact.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

```

Three blue text boxes provide additional information:

- Top box:** "Beachte: Die Parameterbezeichnungen wurden geändert" (Note: The parameter names have been changed).
- Middle box:** "BR ist eine generierte Klasse, die alle Ressourcen beinhaltet, die für das DataBinding benötigt werden" (BR is a generated class that contains all resources needed for DataBinding).
- Bottom box:** "@get Vgl. <https://developer.android.com/topic/libraries/data-binding/observability>" (Reference to the @get annotation documentation).

The IDE interface includes a Project Manager on the left showing the project structure, a toolbar at the top, and a status bar at the bottom indicating "Gradle build finished in 7 s 895 ms (a minute ago)".



```

<?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>

```

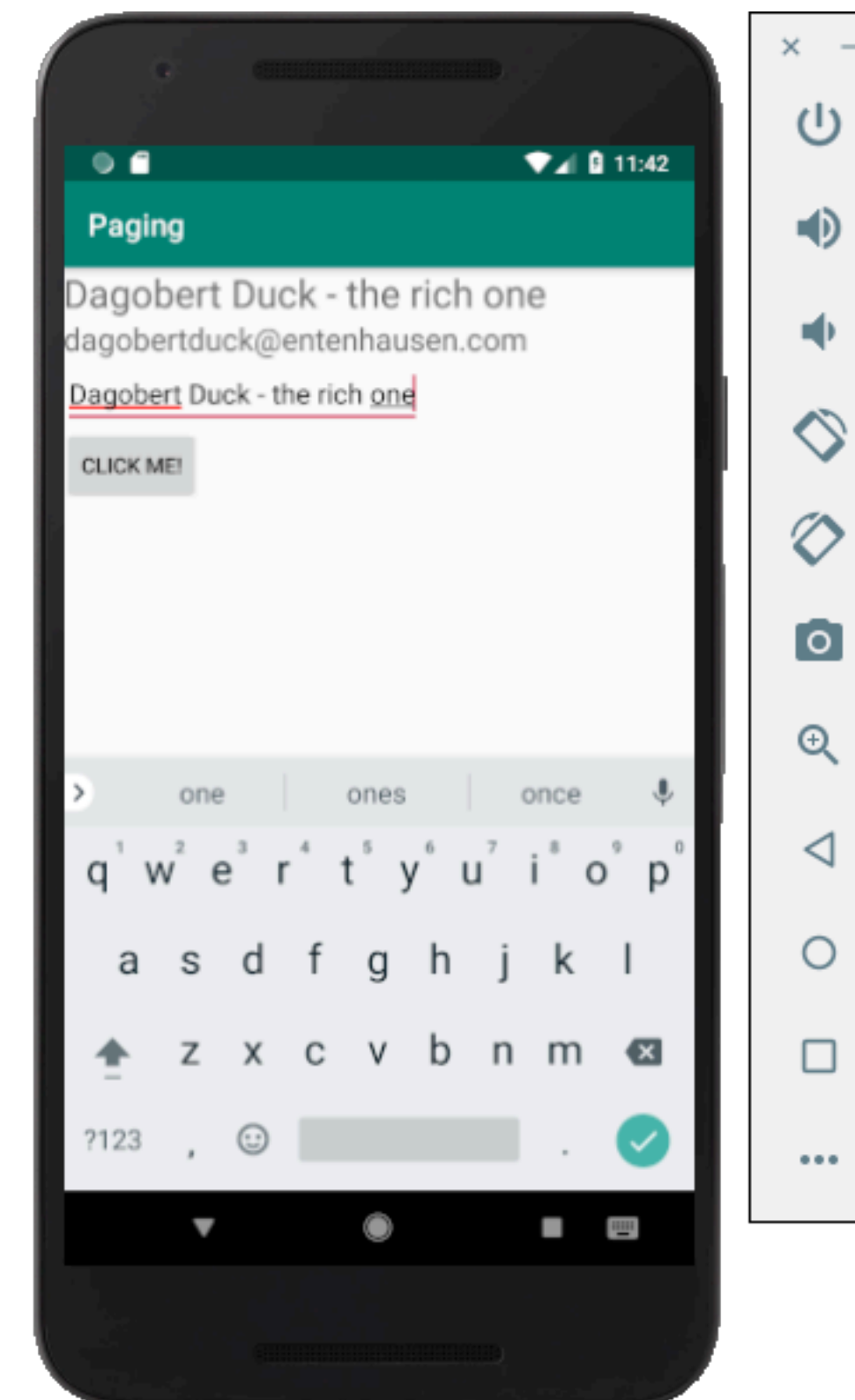
### <EditText

```

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



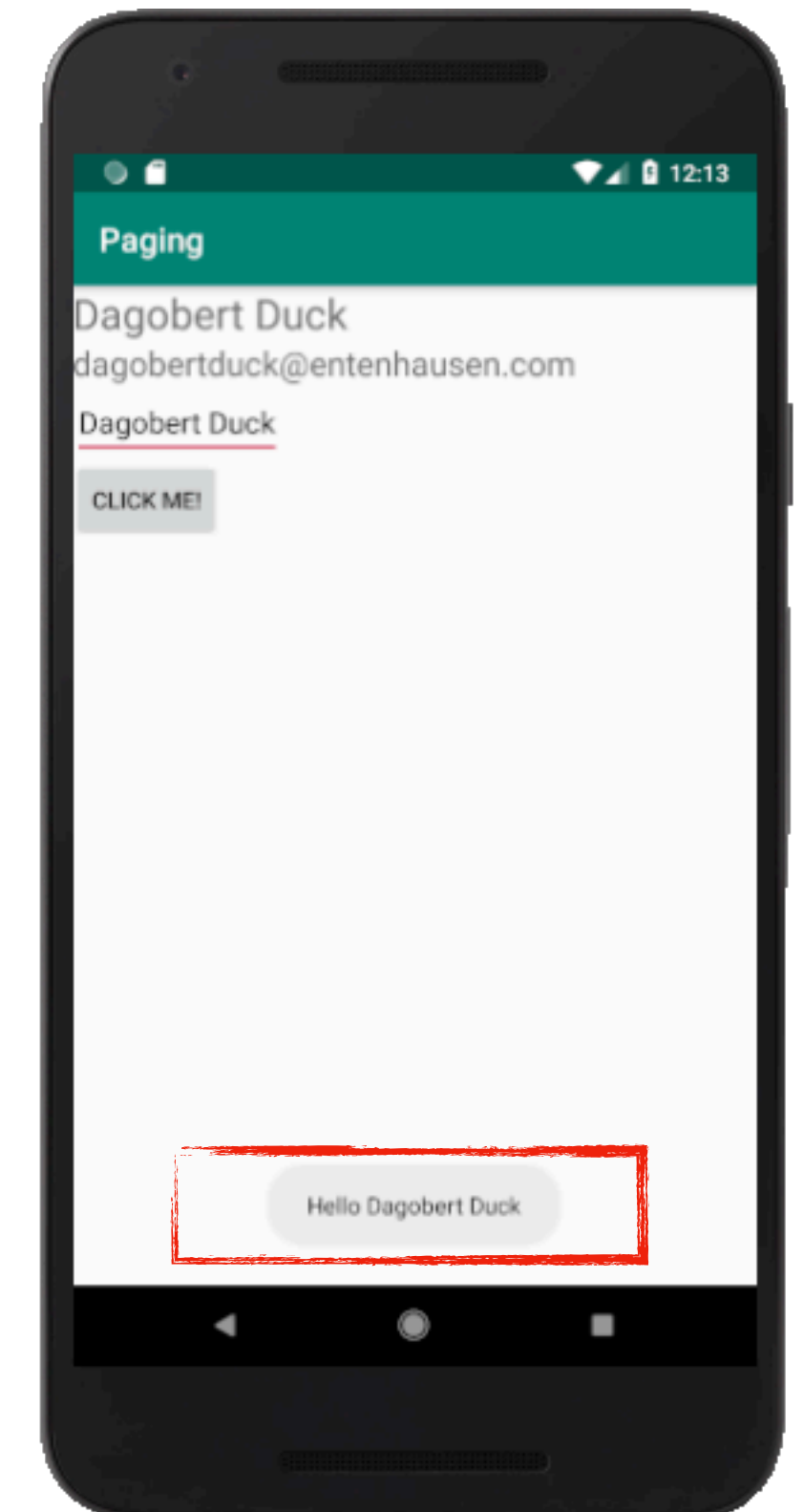
# EventHandler mit Parameter

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

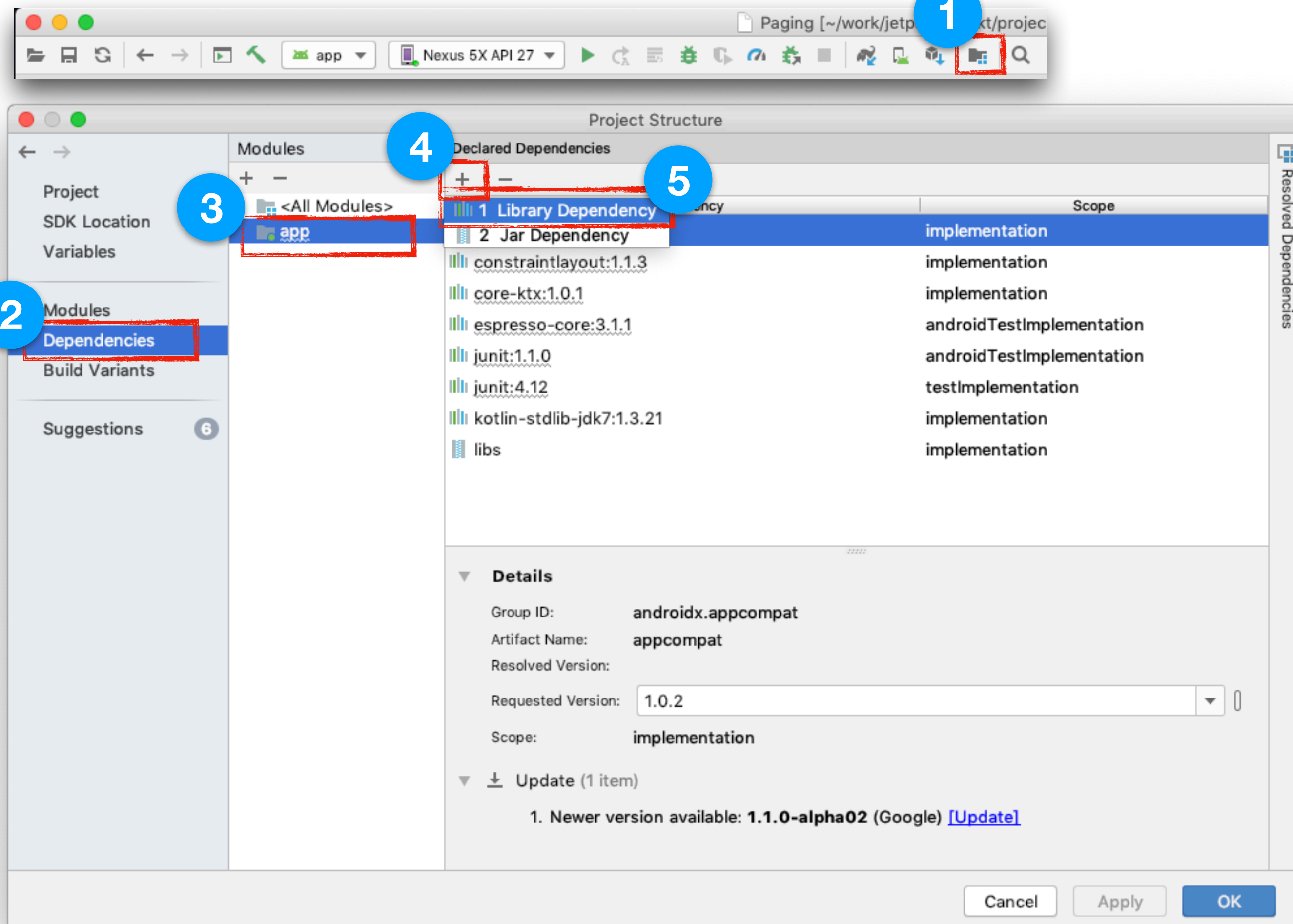
```
open class EventHandler(context: Context) {
    val myContext = context
    fun onButtonClick(name: String) {
        Toast.makeText(myContext, "Hello $name", Toast.LENGTH_SHORT).show()
    }
}
```



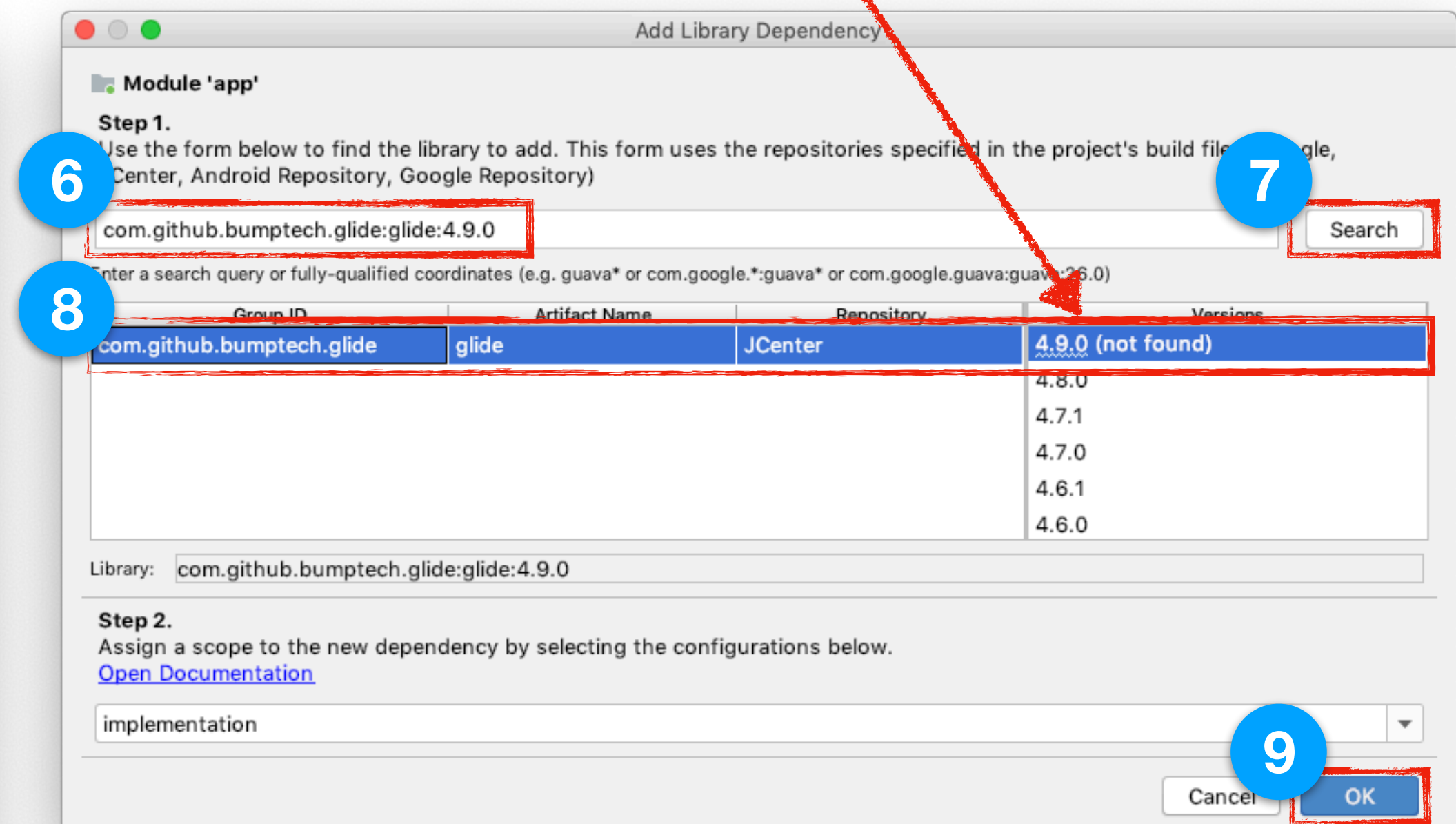
# Loading images from an URL with Glide

# Voraussetzungen

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

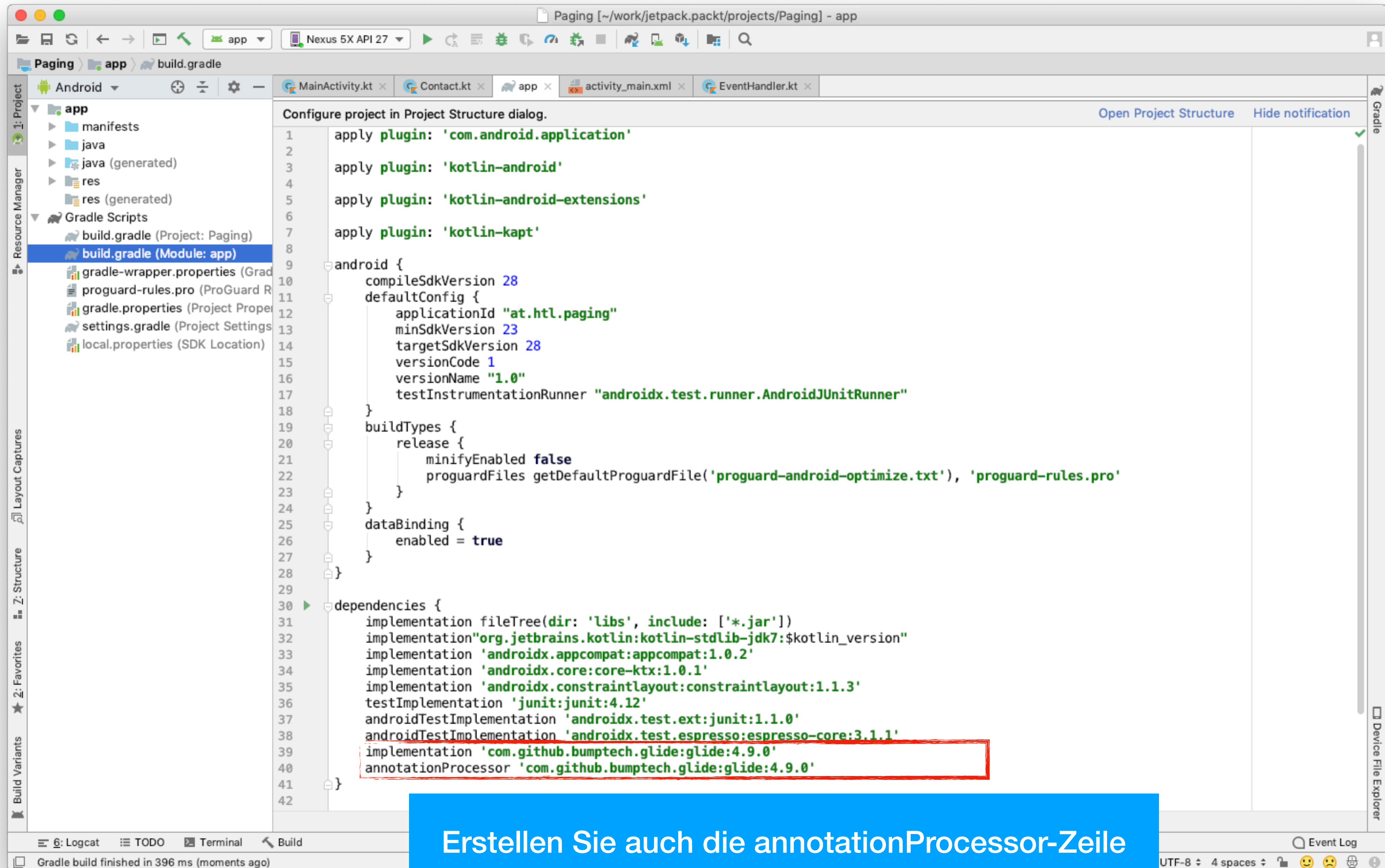


Die Library in der Version 4.9.0 wird nicht gefunden, da sie noch nicht im lokalen maven-repo enthalten ist



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





# manifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="at.htl.paging">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="@string/app_name"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportsRtl="true"
11        android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN"/>
15
16                 <category android:name="android.intent.category.LAUNCHER"/>
17             </intent-filter>
18         </activity>
19     </application>
20
21     <uses-permission android:name="android.permission.INTERNET" />
22
23 </manifest>
24
25
```

Gradle build finished in 472 ms (17 minutes ago)

Event Log

25:1 LF UTF-8 4 spaces



# 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">
```

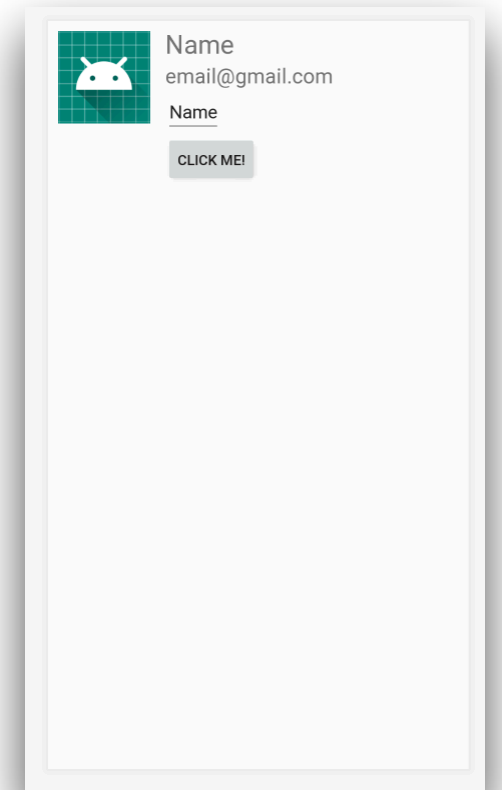
```
    <data>
        <variable
            name="contact"
            type="at.htl.paging.Contact"/>
        <variable
            name="handler"
            type="at.htl.paging.EventHandler"/>
```

```
        <variable
            name="imageUrl"
            type="String"/>
    </data>
```

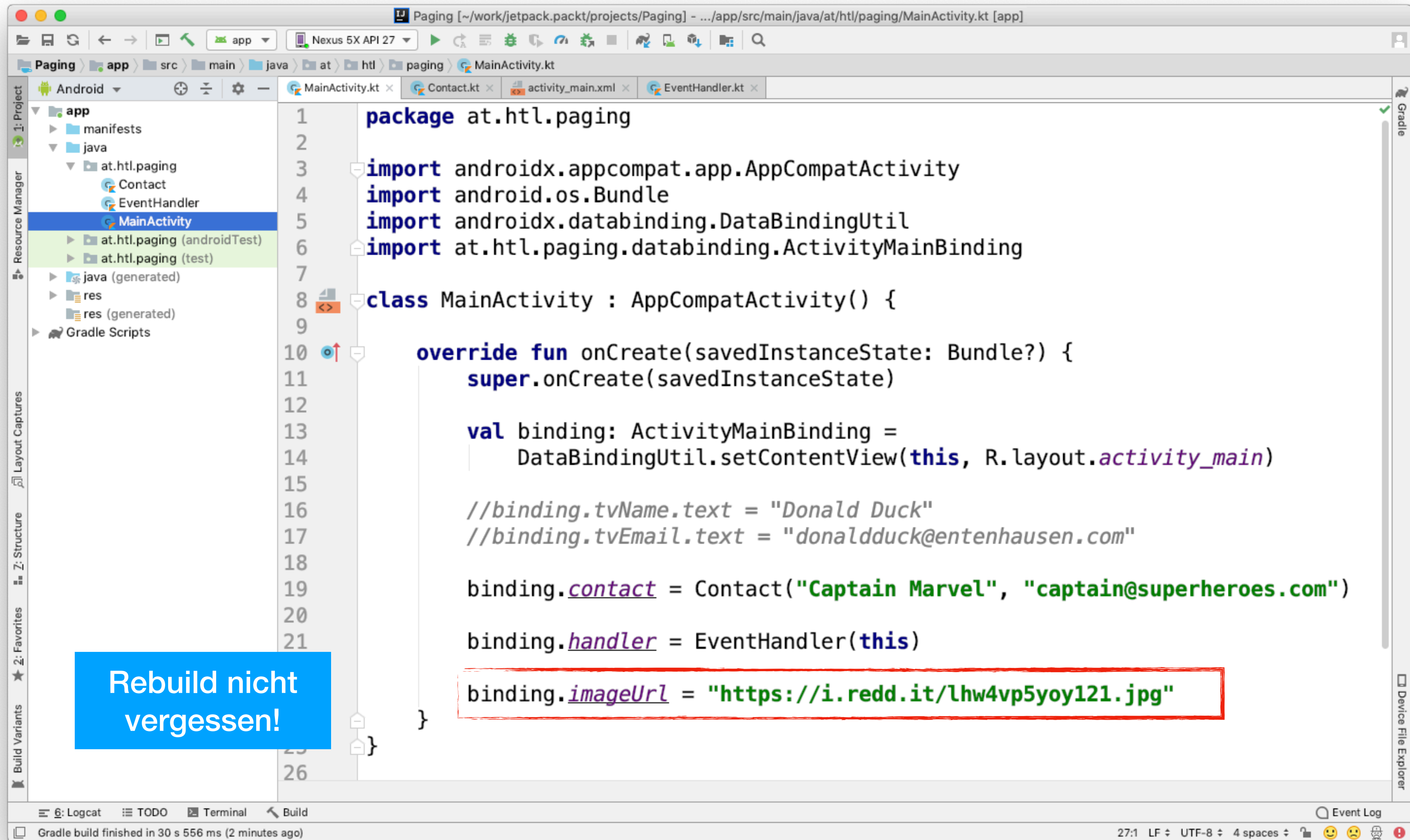
```
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">
```

```
        <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>
```

```
        <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>

```
Paging [~/work/jetpack.packt/projects/Paging] - .../app/src/main/java/at/htl/paging/Contact.kt [app]
Paging > app > src > main > java > at > htl > paging > Contact.kt
MainActivity.kt x Contact.kt x activity_main.xml x EventHandler.kt x
app
├── manifests
├── java
├── java (generated)
├── res
├── res (generated)
├── Gradle Scripts
│   ├── build.gradle (Project: Paging)
│   └── build.gradle (Module: app)
│       ├── gradle-wrapper.properties (Gradle)
│       ├── proguard-rules.pro (ProGuard Rules)
│       ├── gradle.properties (Project Properties)
│       ├── settings.gradle (Project Settings)
│       └── local.properties (SDK Location)
└── ...

4  import androidx.databinding.BaseObservable
5  import androidx.databinding.Bindable
6  import androidx.databinding.BindingAdapter
7  import com.bumptech.glide.Glide
8
9  class Contact(var _name: String, var _email: String) : BaseObservable() {
10
11     @get:Bindable
12     var name: String = _name
13     set(value) {
14         field = value
15         notifyPropertyChanged(BR.name)
16     }
17
18     @get:Bindable
19     var email: String = _email
20     set(value) {
21         field = value
22         notifyPropertyChanged(BR.email)
23     }
24
25     companion object {
26         @JvmStatic
27         @BindingAdapter("profileImage")
28         fun loadImage(view: ImageView, imageUrl: String) {
29             Glide.with(view.context)
30                 .load(imageUrl)
31                 .into(view)
32         }
33     }
34
35 }
36
37
```

### 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">

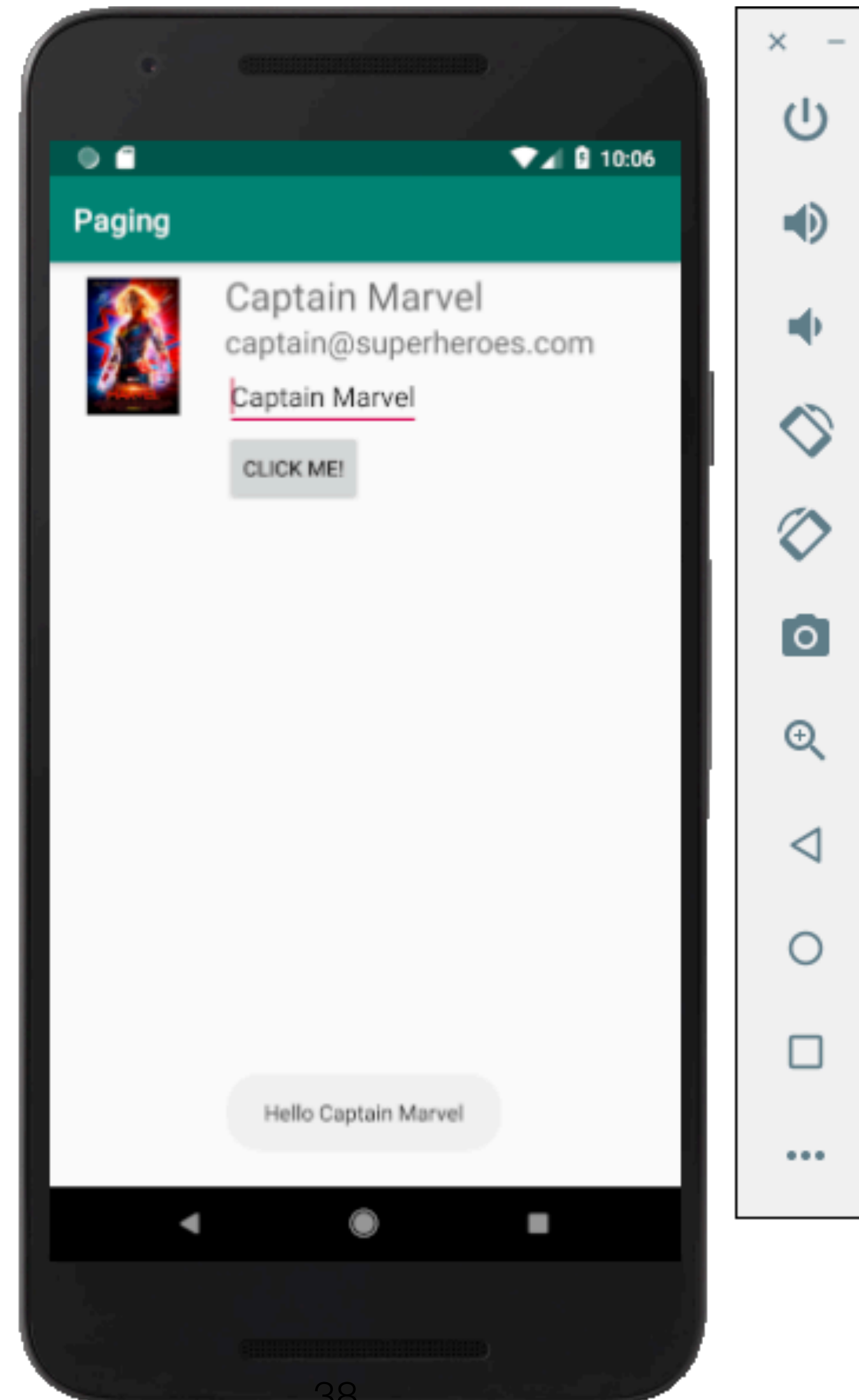
    <ImageView
      android:id="@+id/iv_profile_image"
      android:layout_width="100dp"
      android:layout_height="100dp"
      android:padding="5dp"
      app:profileImage="@{imageUrl}"
      android:src="@mipmap/ic_launcher"/>

  </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/>

The screenshot shows the UdeMy course page for 'Android Jetpack Architecture Components'. At the top, there is a navigation bar with the UdeMy logo, a search bar, and links for 'UdeMy for Business', 'Become an instructor', 'Log In', and 'Sign Up'. Below the navigation bar, the breadcrumb trail reads 'Development > Mobile Apps > Android Game Development'. The course title 'Android Jetpack Architecture Components' is prominently displayed, followed by the subtitle 'Utilize Android Jetpack Architecture components to make your Android application development flexible and maintainable'. A 'NEW' badge is present, along with a star rating of 0.0 (0 ratings) and '4 students enrolled'. The course is created by Packt Publishing and was last updated in 2/2019. It is available in English and English [Auto-generated]. On the right side, there are buttons for 'Gift This Course' and 'Wishlist'. A video preview player is shown with a play button and the text 'Preview this course'.

### 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

**€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



# 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>
-