Datenanbindung

Daniel Bersenkowitsch

20. Dezember 2015

Inhaltsverzeichnis

1 Was ist Datenanbindung?

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. [1]
Beim Anzeigen von Daten in einer Listenansicht beispielsweise muss man das Steuerungselement nach jeder Veränderung der Daten aktualisieren. Wenn man aber die Technologie des Datenanbindung verwendet, braucht man nur die Objektliste an das Steuerungselement mit einem einfachen Zuweisungsbefehl anbinden. Dadurch erneuert sich die Listenansicht der Daten jedes Mal automatisch, sobald sich die Objektliste auch verändert.

1.1 Wozu Databinding in Android verwenden?

In erster Linie werden dem Programmierer dadurch viele Zeilen Code erspart. Datenanbindung trennt einen großen Teil des UI codes von den Aktivitäten und Fragmenten, wodurch eine bessere Übersicht über das Projekt verschafft wird. Zusätzlich, dadurch, dass die XML-Layoutdatei direkt auf die gebundenen Objekte und ihre Attribute zugreift, erspart man sich umständliche findViewById Codiereungen, die sehr performancelastig sind. [2]

2 Vorher - Nacher

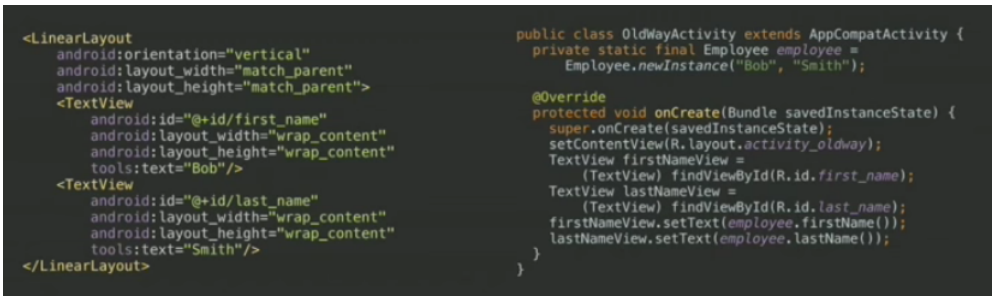
2.1 Vorher

Hier auf der untigen Grafik sehen wir eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität ohne Datenanbindung. Wir sehen ein LinearLayout mit zwei enthaltenen TextViews. Diesen wird in der Aktivität den Vornamen und den Nachnamen eines Employee Models zugewiesen. Ein Problem ist es, jedes Element eine ID zuweisen zu müssen. Wenn man jetzt mehrere Views mit unterschiedlichen Layout XML-Dateien und gleichnamigen IDs hat und man später die Refactor-Funktion verwenden will, benennt man alle neu, ohne das man es will. Man muss sich für jedes Element eine Unterschiedliche ID einfallen lassen, obwohl manche die selbe Funktion haben. Dadurch entstehen lange und unübersichtliche ID-Namen, die man sich

[1] https://www.it-visions.de/

[2] "Droidcon NYC 2015 - Data Binding Techniques"
https://www.youtube.com/watch?v=WdUbXWztKNY

nicht merken kann. Das Problem ist: Es muss immer darauf geachtet werden, keine doppelten IDs zu vergeben.



```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/first_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:text="Bob"/>
    <TextView
        android:id="@+id/last_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:text="Smith"/>
</LinearLayout>
```

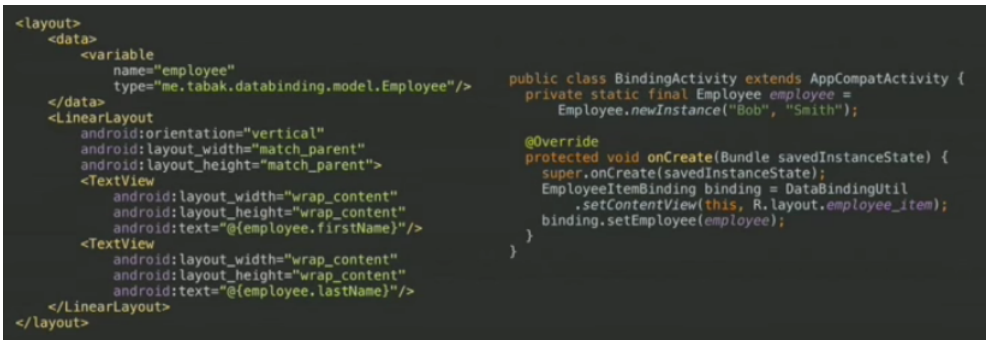
```
public class OldWayActivity extends AppCompatActivity {
  private static final Employee employee =
      Employee.newInstance("Bob", "Smith");

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_oldway);
    TextView firstNameView =
        (TextView) findViewById(R.id.first_name);
    TextView lastNameView =
        (TextView) findViewById(R.id.last_name);
    firstNameView.setText(employee.firstName());
    lastNameView.setText(employee.lastName());
  }
}
```

Abbildung 1: Screenshot des Codes wie er ohne Datenanbindung aussieht

Ein weitere Umständlichkeit ist es, für jedes Element ein findViewById-casting vornehmen zu müssen, wie wir es im Aktivitätencode vorfinden. Dieses Zugriffsverfahren ist, wie bereits oben erwähnt, unnötig performancelastig und kann vermieden werden.

2.2 Nacher



```
<layout>
    <data>
        <variable
            name="employee"
            type="me.tabak.databinding.model.Employee"/>
    </data>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{employee.firstName}"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{employee.lastName}"/>
    </LinearLayout>
</layout>
```

```
public class BindingActivity extends AppCompatActivity {
  private static final Employee employee =
      Employee.newInstance("Bob", "Smith");

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EmployeeItemBinding binding = DataBindingUtil
        .setContentView(this, R.layout.employee_item);
    binding.setEmployee(employee);
  }
}
```

Abbildung 2: Screenshot des Codes wie er mit Datenanbindung aussieht.

Auf dieser Grafik sehen wir nun eine Android XML-Layoutdatei mit dem Code in der zugehörigen Aktivität mit der Verwendung von Datenanbindung. Der Unterschied ist, sich keine einzelnen IDs für jedes vorkommende Element mehr ausdenken zu müssen. Ein großer Vorteil ist es, keine findViewById aufrufe mehr machen zu müssen. Man übergibt nur noch das zu bindende Objekt, an dessen Attribute sich die in der Layoutdatei befindenen Elemente orientieren. Ein weitere Pluspunkt: Man erkennt sofort für was welches Steuerelement zuständig ist. Ein Programmierer der sich gerade in ein

Projekt einarbeitet erkennt sofort, dass bei der ersten TextView der Vorname eines Employeeobjektes angezeigt wird und bei der anderen der Nachname.[3]

3 Wie wendet man Datenanbindung an?

Bei der verwendung von Datenanbindung muss man darauf achten, eine aktuelle Gradle version zu benutzen (min. 1.3). Zusätzlich muss man in der build.gradle (Module App) Datei innerhalb des android{}-Bereichs dataBinding auf enabled=true (dataBinding{enabled=true}) setzen, um Datenanbindung möglich zu machen.

3.1 In der .xml Datei

Gehen wir davon aus, unsere .xml Datei besteht aus einem Linearlayout und zwei TextViews: Remo

Listing 1: Layoutcode ohne jedliche Datenanbindung.

```
<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout xmlns:android="http://schemas.
   android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        <TextView
            android:id="@+id/vorname"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Max" />
        <TextView
            android:id="@+id/nachname"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Mustermann" />
</LinearLayout>
```

Nun, um Datenanbindung zu ermöglichen, brauchen wir eine Objektklasse auf die wir Referenzieren. In diesem Fall erstellen wir eine Klasse "User"mit den String-Attributen firstName und lastName. Die Klasse besteht weiters aus einem Konstruktorfeld mit den Attributen und jeweiliger getter-Felder:

[3]"Droidcon NYC 2015 - Data Binding Techniques"
https://www.youtube.com/watch?v=WdUbXWztKNY

Listing 2: Unsere Objektklasse die bei der Datenanbindung referenziert wird.

```
package com.example.daniel.showcasedatabinding;
public class User {
    private final String firstName;
    private final String lastName;

    public User(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    public String getFirstName() {
        return this.firstName;
    }
    public String getLastName() {
        return this.lastName;
    }
}
```

Haben wir diese erstellt, kann bei der Layoutdatei weitergemacht werden. Wir erstellen nun um unser Layout einen "layoutTag. Es folgt ein variable Tag innerhalb eines data Tags. Darin definieren wir eine Variable auf die man innerhalb der Elemente zugreifen kann.

Listing 3: Die XML Datei nach der Integration einer Datenanbindung.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/
   apk/res/android">
    <data>
        <variable name="user" type="com.example.
   daniel.showcasedatabinding.User"/>
    </data>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{user.firstName}" />

        <TextView
```

```
17              android:layout_width="wrap_content"
18              android:layout_height="wrap_content"
19              android:text="@{user.lastName}" />
20          </LinearLayout>
21  </layout>
```

Jetzt können wir unsere ID-Vergabe bei den Elementen löschen und im Text-Tag auf die jeweiligen Attribute des Userobjekts verweisen.

3.2 In der .java Klasse

Die Datenanbindungstechnologie von Android generiert automatisch spezielle Bindingklasse all jener Layoutdateien die es verwenden. Der Name ergibt sich aus dem Namen der .xml Datei + "binding". Also wenn eine Layoutdatei activity_main.xml benannt ist, wird daraus die Klasse ActivityMainBinding generiert, die wir nun verwenden können:

Listing 4: Die Aktivitätenklasse nach der Integration einer Datenanbindung.

```
1  package com.example.daniel.showcasedatabinding;
2
3  import android.databinding.DataBindingUtil;
4  import android.os.Bundle;
5  import android.support.v7.app.AppCompatActivity;
6
7  import com.example.daniel.showcasedatabinding.
       databinding.ActivityMainBinding;
8
9  public class MainActivity extends AppCompatActivity
       {
10
11     @Override
12     protected void onCreate(Bundle
       savedInstanceState) {
13         super.onCreate(savedInstanceState);
14
15         ActivityMainBinding binding =
       DataBindingUtil.setContentView(this, R.layout.
       activity_main);
16         User user = new User("Max", "Mustermann");
17         binding.setUser(user);
18     }
19 }
```

Es wird eine Instanz der generierten Klasse erstellt, welche dann ein Objekt angebunden bekommt, dessen Attribute die Elemente bekommen. Jede Änderung der verwendeten Eigenschaften der erstellten Userinstanz bedeutet auch eine Änderung des Elements, automatisch. [4]

[4] http://developer.android.com/tools/data-binding/guide.html

4 Quellen:

- https://www.it-visions.de/
- "Droidcon NYC 2015 - Data Binding Techniques"
  https://www.youtube.com/watch?v=WdUbXWztKNY
- http://developer.android.com/tools/data-binding/guide.html