

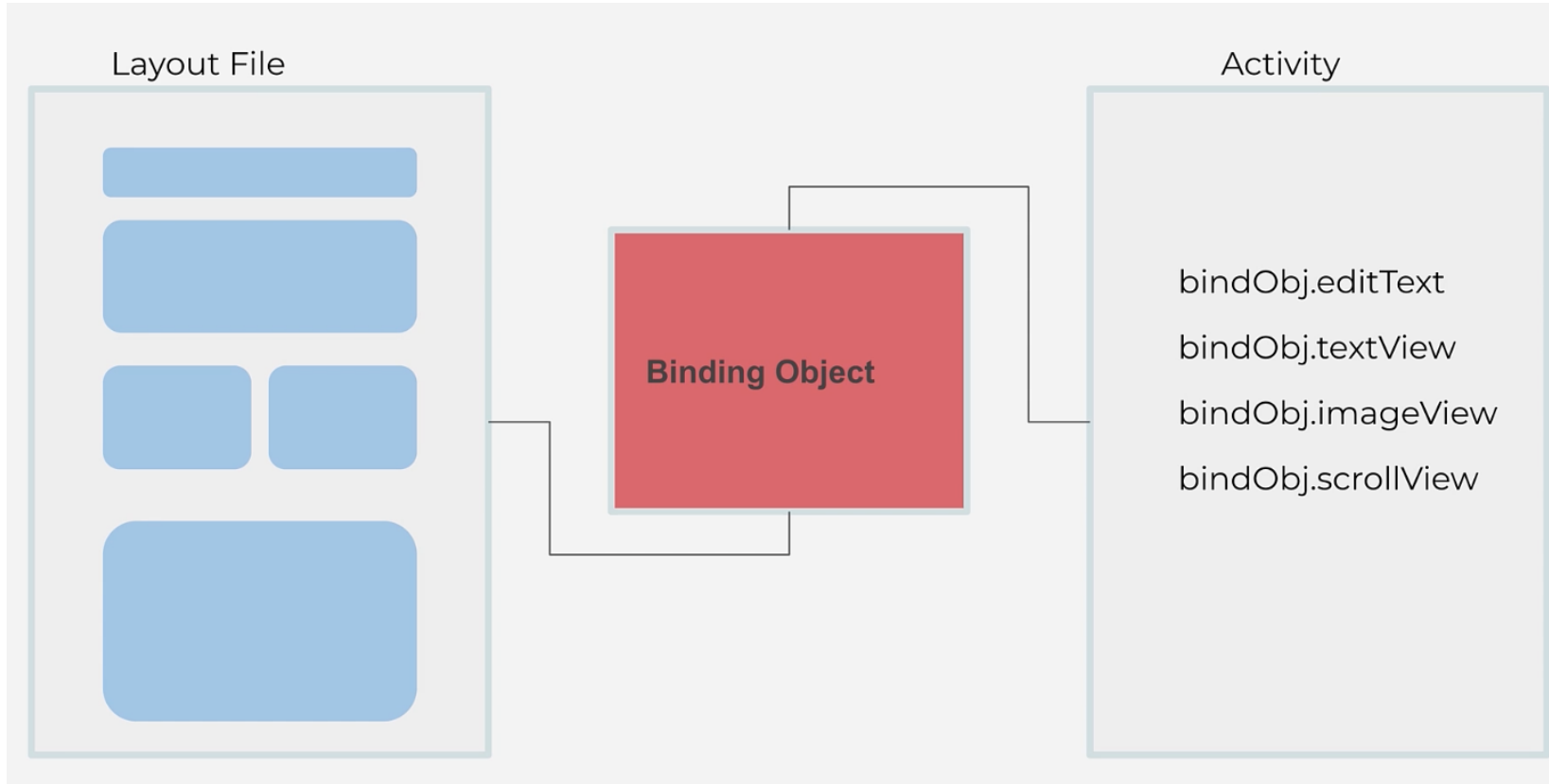
APLIKACJE MOBILNE

Wykład 03

dr Artur Bartoszewski

View Binding

View Binding - to mechanizm w Androidzie, który pozwala na łatwiejsze i bezpieczniejsze odwoływanie się do widoków w layoutach XML z poziomu kodu Java



Po włączeniu View Binding, dla każdego pliku layoutu XML zostanie wygenerowana klasa bindingu (np. `ActivityMainBinding` dla `activity_main.xml`).

Zalety View Binding:

- Zamiast używać `findViewById()` do wyszukiwania widoków po id, View Binding generuje klasę z odwołaniami do wszystkich widoków w layoucie.
- View Binding zapewnia bezpieczeństwo typów, co oznacza, że kompilator wykryje błędy, jeśli spróbujesz odwołać się do widoku o niewłaściwym typie.
- View Binding eliminuje ryzyko wystąpienia `NullPointerException`, ponieważ generowane klasy zawierają tylko odwołania do widoków, które istnieją w layoucie.
- View Binding sprawia, że kod jest bardziej czytelny i łatwiejszy w utrzymaniu.

1. Uruchomienie mechanizmu View Binding:

- W pliku build.gradle (Module :app) dodać należy wpis:

```
android {  
    namespace = "com.example.bindungtest"  
    compileSdk = 34  
    .....  
    .....  
    buildFeatures {  
        viewBinding = true  
    }  
}
```

2. Utworzenie obiektu binding

```
public class MainActivity extends AppCompatActivity {
```

```
    private ActivityMainBinding binding;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);
```

```
        binding = ActivityMainBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());
```

```
        // Teraz możesz odwoływać się do widoków z layoutu używając binding
```

```
    }
```

```
}
```

używamy funkcji
ActivityMainBinding.inflate() do
utworzenia obiektu binding.

ustawiamy widok główny layoutu jako
content view aktywności za pomocą
setContentView(binding.getRoot()).

3. Odwoływanie się do kontrolek za pomocą obiektu binding

```
binding.textView01.setText("Tekst do wstawienia");
```

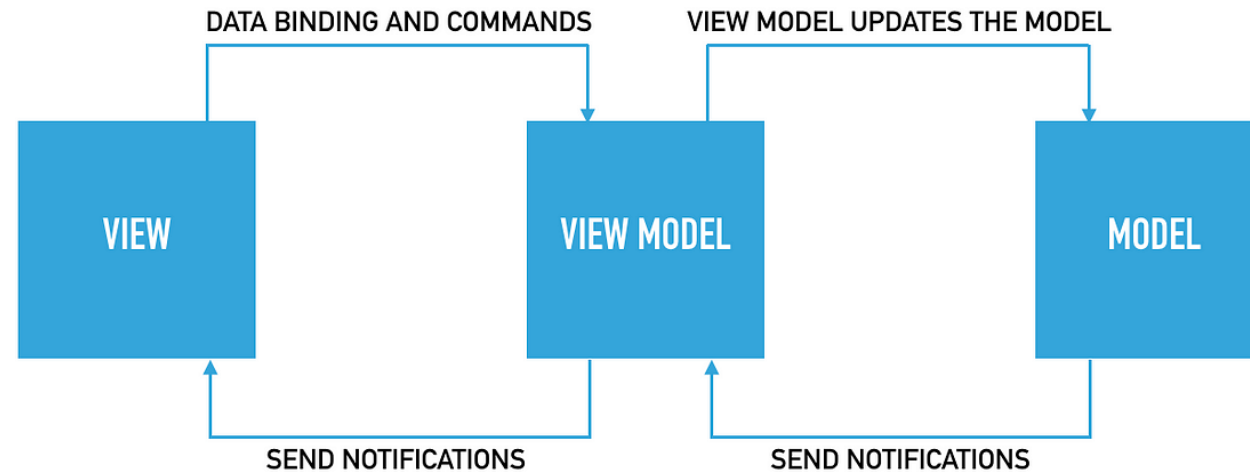
```
String pobrany_tekst = binding.editText01.getText().toString();
```

```
binding.button01.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
  
    }  
});
```

Przykład odwołania się do pola tekstowego, pola edycyjnego oraz przycisku za pomocą obiektu binding.

Data Binding

Data Binding - to bardziej zaawansowane rozwiązanie, które pozwala na bezpośrednie łączenie danych z widokami i definiowanie akcji w layoucie XML.



1. Uruchomienie mechanizmu Data Binding:

- W pliku build.gradle (Module :app) dodać należy wpis:

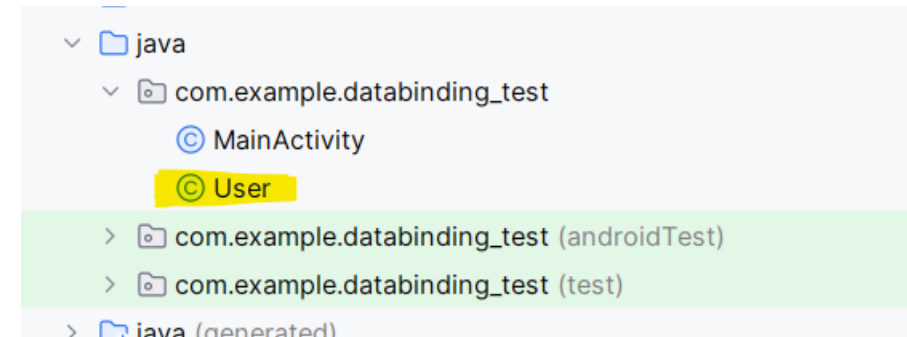
```
android {  
    namespace = "com.example.bindungtest"  
    compileSdk = 34  
    .....  
    .....  
    buildFeatures {  
        dataBinding = true  
    }  
}
```

2. Tworzymy klasę będącą modelem danych, na którym będziemy pracować:

```
package com.example.databinding_test;
```

```
public class User {  
    public String imie;  
    public String nazwisko;
```

```
    public User(String imie, String nazwisko) {  
        this.imie = imie;  
        this.nazwisko = nazwisko;  
    }  
}
```



3. Definiujemy layout XML z Data Binding::

```
<?xml version="1.0" encoding="utf-8"?>
<layout>
  <data>
    <variable
      name="user"
      type="com.example.databinding_test.User" />
    </data>
  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ...
  >
```

W layoucie XML używamy tagu <layout> i <data> do definiowania zmiennych.

Wewnątrz tagu <data> definiujemy zmienną user typu User.

```
<TextView
  android:text="@{user.imie + ' ' + user.nazwisko}"
  ... />
```


W TextView używamy wyrażenia @{user.imie + ' ' + user.nazwisko} do wyświetlenia imienia i nazwiska użytkownika.

```
</androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

2. Utworzenie obiektu binding

```
public class MainActivity extends AppCompatActivity {  
  
    private ActivityMainBinding binding;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);  
        User user = new User("Jan", "Kowalski");  
        binding.setUser(user);  
  
    }  
}
```

W kodzie Java tworzymy obiekt User i ustawiamy go jako zmienną w obiekcie binding.



View Binding VS Data Binding

View Binding

Cel: Uproszczenie dostępu do widoków w kodzie Java.

Działanie: Generuje klasę dla każdego pliku layoutu XML, która zawiera odwołania do wszystkich widoków z id.

Korzyści:

- Eliminuje findViewById().
- Bezpieczeństwo typów.
- Null safety.

Czytelność kodu.

Ograniczenia:

- Nie obsługuje bezpośredniego wiązania danych z obiektami.
- Nie pozwala na definiowanie akcji w layoutcie XML.

Data Binding

Cel: Bezpośrednie łączenie danych z widokami w layoutcie XML.

Działanie: Umożliwia definiowanie zmiennych w layoutcie XML i łączenie ich z widokami za pomocą wyrażeń.

Korzyści:

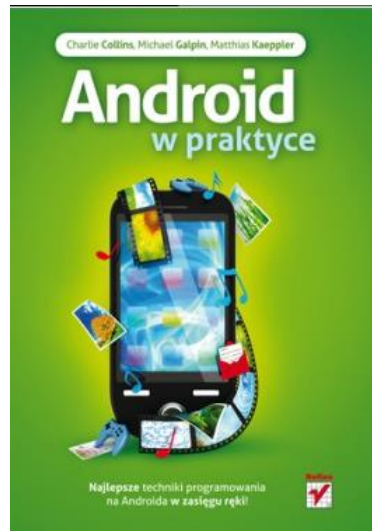
- Wyświetlanie danych z obiektów bezpośrednio w widokach.
- Obsługa zdarzeń w layoutcie XML.
- Dwukierunkowe wiązanie danych.

Ograniczenia:

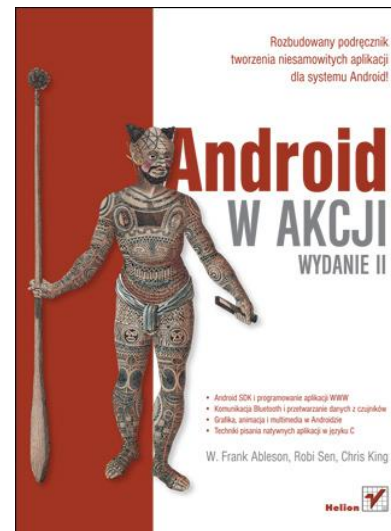
- Bardziej złożona konfiguracja.
- Może wpływać na wydajność, jeśli jest nadużywany.

Które wybrać?

- Jeśli potrzebujesz tylko prostego sposobu na dostęp do widoków, View Binding jest wystarczający.
- Jeśli chcesz korzystać z dwukierunkowego wiązania danych lub definiować akcje w layoucie XML, użyj Data Binding.



<https://developer.android.com>



<https://javastart.pl/baza-wiedzy/android/>

<https://forum.android.com.pl>

