Android Programmierung mit Kotlin und JetPack

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://mwilhelm.hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Gliederung

Überblick:

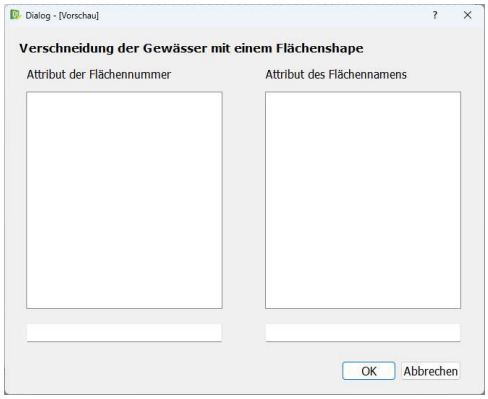
- **Jetpack**
  - Struktur
  - „Views“
  - Row
  - Column
  - Text, TextField
  - Button, TextButton, ToggleButton
  - CheckBox, RadioButton, Switch, Range
  - BottomNavigation
  - TopAppBar

Links

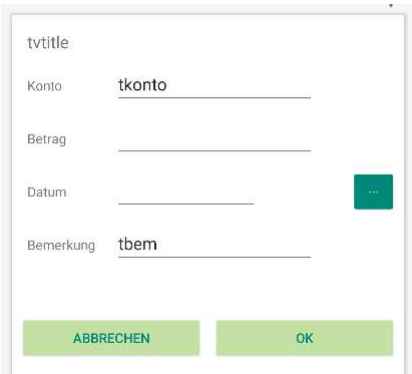
- https://developer.android.com/jetpack/compose/documentation
- https://developer.android.com/jetpack/compose/tutorial
- https://www.jetpackcompose.net
- https://www.geeksforgeeks.org/listview-in-android-using-jetpack-compose/

Klassischer Ansatz:

- Man hat eine „Pinwand“, auf der man Elemente einträgt:
  - Delphi, WPF, Android, Xcode, Qt
- Man braucht eine Verknüpfung von UI-Element zur Variablen (findViewById)
- Komponentenbaum



Klassischer Ansatz:



Vererbung:

- **Hauptklasse:** **View**
- Abgeleitet: Labelfeld
- Abgeleitet: TextField

- **Hauptklasse:** **View**
- Abgeleitet: Labelfeld
- Abgeleitet: Button mit onClick-Event

- Die UI-Klassen sind festdefiniert und können nur mittels Vererbung erweitert werden.

- **Für einen ImageButton gilt folgender „Baum“:**
- Hauptklasse: View
- Abgeleitet: Image
- Abgeleitet: ImageButton mit onClick-Event

JetPack: Deklarative Programierung

Nr	Compose	Erläuterung
1	Text("Hallo World")	Anzeige eines Labelfelds
2	Text("Hallo World1") Text(text ="Hallo World2")	Anzeige zweier Labelfelder, die übereinander liegen.

3	Column {
	Text("Hallo World1")
	Text(text ="Hallo World2")
	}
		Anzeige zweier Labelfelder, die durch Column untereinander liegen.
		Column nach unten
		Row nach rechts

4	Column {
	Text("Überschrift ")
	Row {
	Text(text ="Eingabe")
	TextField()
	}
	}
		Anzeige eines Dialogs mit einer Überschrift und darunter die Beschrifung und die Eingabe.

JetPack: Deklarative Programierung

Nr	Compose	Erläuterung

5	Button(
	onClick = {
	println("Klick")
	},
	enabled = true
	) {
	Text(text = "Klick mich")
	}
		Ein Button muss immer ein Klick-Event haben. Nach der Definition kann man weitere Eigenschaften hinzufügen (hier enabled).
		Die Beschriftung erfolgt dann in geschweiften Klammern.

6	Button(
	onClick = {
	println("Klick")
	},
	enabled = true
	) {
	Column {
	Text(text = "Klick mich")
	Text(text = "An")
	}
	}
		Diese Variante hat durch die Funktion Column zwei Label als Beschriftung. Die Ausrichtung der beiden Texte ist linksbündig.

JetPack: Deklarative Programierung

```
Button(
   onClick = {
      println("Klick")
   },
    enabled = true
) {
   Column(horizontalAlignment = Alignment.CenterHorizontally) {
      Text(text = "Klick mich")
      Text(text = "An")
    }
}
```

Diese Variante hat durch die Funktion Column zwei Label als Beschriftung. Die Ausrichtung der beiden Texte ist zentriert.

JetPack: Deklarative Programierung

Nr	Compose	Erläuterung

5	Button(
	onClick = {
	println("Klick")
	},
	enabled = true
	) {
	TextField(value = "abc",
	onValueChange =
	{println("111")})
	}
		Diese Variante hat ein TextField als Beschriftung (ähnlich in WPF). Damit kann man flexibel die UI-Elemente aufbauen. Zum Beispiel mit einem Image und einem Text-Element in einer ROW.

JetPack: Model View Controller

In JetPack erhalten die UI-Elemente keine Namen. Die Statusänderung erfolgt mittels Statusvariablen, die im Prinzip ein MVC sind.

- **Statusvariable:**

```
val text1 = remember { mutableStateOf("12345") }
TextField(
    value = text1,
    onValueChange = { text1.value = it }
)
```


JetPack: Model View Controller

```
Modell modell = new Modell("12345")
text1 : String
modell.add(text1)
t1: MyTextField = new MyTextField(modell)

class MyTextField:TextField  implements IUpdate {
  public MyTextField(Modell modell) {
    modell.add(this)
  }
   public void IUpdate() {
      super.value = modell.getValue()
   }
   onValueChange() {
     modell.value = super.value
   }
```

JetPack: Text, Eigenschaften

- fontSize = TextUnit(value = 20.0F, type = TextUnitType.Sp)
- style = MaterialTheme.typography
  - h1 // größte Schriftgröße
  - h6 // sehr klein
  - overline
  - caption
  - body1
  - body2
  - button
  - subtitle1
  - subtitle2

JetPack: Text, Eigenschaften

- color=Color.Blue
  - color=Color(android.graphics.Color.parseColor("#FF00FF")) // Magenta
- textAlign = TextAlign.Center,
- fontFamily = FontFamily.Serif FontFamily.SansSerif
- fontWeight =FontWeight
  - Bold
  - Light
  - ExtraBold
  - Medium
  - ExtraLight
  - Normal
  - SemiBold
  - Thin
  - W100 bis W900

JetPack: Text, Eigenschaften

- fontStyle =FontStyle
  - Italic
  - Normal
- maxLines=2
- overflow = TextOverflow.
- Ellipsis
  - der Text wird abgeschnitten und drei Punkte signalisieren, dass der Text abgeschnitten wurde.
- Clip
  - Es werden nur die Worte angezeigt, die angezeigt werden können.
  - Gibt es nur ein Wort wird abgeschnitten.
- Visible

JetPack: Text, Beispiele

Minimalbeispiel

- Text(text ="Hier ist ein Text")

Beispiel mit Schriftgröße und Farbe

- Text(
- text ="Hier ist ein Text"
- style = MaterialTheme.typography.h3,
- color= Color.Blue,
- fontWeight = FontWeight.Bold
-)

JetPack: Text, Beispiele

Beispiel mit zusätzlichen Klick-Event

```
Text(
  text ="Hier ist ein Text"
  modifier = Modifier
    .padding(8.dp)
    .align(alignment = Alignment.CenterVertically)
    .clickable {
        // hier AktioncheckedSwitch.value = !checkedSwitch.value
     },
  style = MaterialTheme.typography.h3,
  color= Color.Blue,
  fontWeight = FontWeight.Bold
)
```


JetPack: Text, Beispiele

Beispiel mit einem Text und einer Status-Variablen

```
var text1 by remember { mutableStateOf("12345") }
Text(
  text =text1,
  modifier = Modifier
    .padding(8.dp)
  style = MaterialTheme.typography.h3,
  color= Color.Blue,
  fontWeight = FontWeight.Bold
)
```

JetPack: TextField: Eigenschaften

- value = rememberVariable (TextFieldValue)
- modifier: Modifier = Modifier
  - modifier = Modifier
    - .alignByBaseline()
    - .weight(1.0F),
    - .clickable {
    - expanded = true
    - state.value = !state.value
    - },
    - .clickable {
      - color = Color.Blue
    - }

JetPack: TextField: Eigenschaften

- colors
  - colors = TextFieldDefaults.textFieldColors(
    - textColor = Color.Green,
    - cursorColor = Color.Cyan),
- enabled: Boolean = true,
- readOnly: Boolean = false,
- textStyle: TextStyle = LocalTextStyle.current,
- label: @Composable (() -> Unit)? = null,
  - label = {
    - Text(
      - text = "Matrikelnr 12345",
      - color = Color.Red,
      - style = MaterialTheme.typography.caption
    -)
  - },

JetPack: TextField: Eigenschaften

- placeholder: @Composable (() -> Unit)? = null,
  - placeholder = {
    - Text(text = "Bitte Text eingeben", color = Color.Blue)
  - },
- singleLine = true,
- maxLines=1,2 oder Int.MAX_VALUE,
- isError: Boolean = false,
  - wenn true, daan wird ein roter Strich unten angezeigt

JetPack: TextField, Beispiele

```
TextField(
  value = text1.value,
  onValueChange = { text1.value = it },
  placeholder = {
    Text(text = "Bitte Text eingeben", color = Color.Blue)
  },
  label = {
    Text(
      text = "Eingabe",
      color = Color.Red,
      style = MaterialTheme.typography.caption
    )
  }
) // TextField
```

JetPack: Button, Eigenschaften

- enabled
- Modifier
  - .align(alignment = Alignment.CenterVertically)
    - .padding(8.dp)
  - border = BorderStroke(1.dp, Color.Red),
  - shape
    - shape = RoundedCornerShape(20.dp)
    - shape = RectangleShape
- colors
  - colors = ButtonDefaults.buttonColors(
    - backgroundColor = Color.DarkGray,
    - contentColor = Color.Green,
    - disabledBackgroundColor = Color.LightGray,
    - disabledContentColor = Color.Yellow
  -),

JetPack: Button, Eigenschaften

- Elevation (Erhebung)
  - elevation = ButtonDefaults.elevation(
    - defaultElevation = 10.dp,
    - pressedElevation = 15.dp,
    - disabledElevation = 0.dp
- onClick

JetPack: Button, Beispiele

Minimaler Quellcode:

```
Button(
  onClick = {
    println("Klick: "+text1.value)
  }
) {
  Text(text = "Klick mich")
}
```


JetPack: Button, Beispiele

Hier ein zweizeiliger Text

```
Button(
  onClick = {
    println("Klick: "+text1.value)
  }
) {
  Column(
    horizontalAlignment = Alignment.CenterHorizontally,
  ) {
    Text(text = "Klick mich")
    Text(text = "An")
  }
}
```