04

04-2. 상태 다루기

Compose

22-2 상태 다루기

상태란?

- 컴포저블 함수는 함수 내에 상태가 선언되어 어떻게 사용되는지에 따라 상태 컴 포저블(Stateful Composable)과 비상태 컴 포저블(Stateless Composable)로 구분

mutableStateOf 함수로 상태 선언

- 변수 자체를 상 태라고 하지는 않습니다.

변수 선언의 예

```
var data1 = 0
```

22-2 상태 다루기

mutableStateOf 함수로 상태 선언

- 컴포저블 함수 내에서 상태 선언은 다음과 같이 mutableStateOf 함수로 합니다.
- mutableStateOf()의 매개변숫값(여기서는 0)은 선언된 상태의 초깃값
- 첫 번째는 상태를 이용하기 위한 변수이며 두 번째는 상태를 변경하기 위한 함수

• mutableStateOf 함수로 상태 선언

```
var (data2, setData2) = mutableStateOf(0)
fun changeData2() {
    setData2(Random.nextInt(0, 100))
}
```

- setter 함수를 호출하면 상태가 변경되고 화면이 재구성
- 상태값은 변경되고 컴포저블 함수가 다시 호출되기는 하지만 상태값은 항상 0
- 상태값을 변경해서 재구성이 이뤄지지만 변경된 상태값이 유지되지 않았기 때문

remember

- 컴포저블이 다시 컴포지션될 때 유실되면 안되는 데이터를 저장하기 위한 함수이다.
- remember 자체가 상태는 아니며 데이터를 저장하기 위한 용도이다.
- remember에 주로 저장하는 것이 상태인 것 뿐이지 상태가 아닌 다른 것을 저장해서 이용해도 된다.

- val mutableState = remember { mutableStateOf(default) }
- val value by remember { mutableStateOf(default) }
- val (value, setValue) = remember { mutableStateOf(default) }

22-2 상태 다루기

remember 함수를 사용해 상태값 유지

- remember를 사용할 때 by를 함께 사용하면 remember에 의해 상태가 선언, 반환된다.

• remember 함수를 사용한 예(by를 함께 사용한 예)

```
var data3 by remember {
    mutableStateOf(0)
}
fun changeData3() {
    data3 = Random.nextInt(0, 100)
}
```

22-2 상태 다루기

remember 함수를 사용해 상태값 유지

- by를 사용하지 않고 remember () 함수를 호출하면 MutableState 객체 반환
- MutableState는 상태의 이용 또는 변경을 감지하는 컴포즈의 클래스
- 상태값을 획득하거나 변경하기 위해서는 이 객체의 value 프로퍼티를 사용

• remember 함수를 사용한 예(by를 사용하지 않은 예)

```
val data4: MutableState<Int> = remember {
    mutableStateOf(0)
}
fun changeData4() {
    data4.value = Random.nextInt(0, 100)
}
```

rememberSaveable

rememberSaveable 함수

- 화면 회전과 같은 액티비티의 상태 변경에도 상태가 그대 로 유지되게 하고 싶은 경우 rememberSaveable 함수 를 이용해 상태를 선언

• rememberSaveable 함수로 상태 선언

```
var aData by remember {
    mutableStateOf(0)
}
var bData by rememberSaveable {
    mutableStateOf(0)
}
Column {
    Row {
        Text("remeber count : ${aData.toString()}", fontSize = 30.sp)
        Button(onClick = { aData++ }) {
            Text(text = "increment")
        }
    }
    Spacer(modifier = Modifier.height(10.dp))
    Row {
        Text("remeber count : ${bData.toString()}", fontSize = 30.sp)
        Button(onClick = { bData++ }) {
            Text(text = "increment")
        }
    }
}
```

rememberSaveable

- rememberSaveable 은 내부적으로 액티비티의 saveInstanceState를 이용하는 것이다. 즉 Bundle 을 이용한다. 그럼으로 Bundle 에 저장할 수 있는 데이터만 가능하다는 한계가 있다.

```
data class User(val name: String, val email: String)

.....

var user by rememberSaveable {
    mutableStateOf<User>(User("kim", "a@a.com"))
}
```

```
java.lang.IllegalArgumentException: MutableState containing User
```

rememberSaveable

방법1 : Parcelize 이용

- 객체 직렬화 가능하게 처리

```
plugins {
    ……
    id("kotlin-parcelize")
}
```

```
@Parcelize
data class User(val name: String, val email: String): Parcelable

……
var user by rememberSaveable {
    mutableStateOf<User>(User("kim", "a@a.com"))
}
```

rememberSaveable

방법2 : mapSaver 활용

- Parcelable 을 사용하지 않고 객체를 저장하는 방법 중 하나
- mapSaver() 는 compose 에서 제공하는 API 로 객체의 내용을 key-value 형태로 Map 에 저장했다가 복원하는 방법을 도와준다.
- mapSaver() 에 save, restore 매개변수를 지정한다. 각각 함수로 지정된다.
- save 함수에는 객체의 내용을 Map 으로 전환하는 규칙
- restore 에는 Map 을 객체로 전환하는 규칙을 명시하면 된다.

rememberSaveable

방법2 : mapSaver 활용

```
data class User(val name: String, val email: String)

val UserSaver = run {
    val nameKey = "name"
    val emailKey = "email"
    mapSaver(
        save = { mapOf(nameKey to it.name, emailKey to it.email) },
        restore = { User(it[nameKey] as String, it[emailKey] as String)}
    )
}

.............................
var user by rememberSaveable(stateSaver = UserSaver) {
    mutableStateOf<User>(User("kim", "a@a.com"))
}
```

rememberSaveable

방법2 : listSaver 활용

- mapSaver 와 비슷하며 Map 이 아닌 List 에 객체의 내용을 저장하게 하기 위한 Saver 이다.

```
data class User(val name: String, val email: String)
val UserSaver = listSaver<User, Any>(
    save = { listOf(it.name, it.email) },
    restore = {User(it[0] as String, it[1] as String)}
)

.....................
var user by rememberSaveable(stateSaver = UserSaver) {
    mutableStateOf<User>(User("kim", "a@a.com"))
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare