Navigation & State

2024.11.13.Wed | Android Compose Study

지영

Navigation

화면 전환

Navigation | 정의

Navigation refers to the interactions that let users navigate across, into, and back out from the different pieces of content within your app.

즉, 화면 전환이나 특정 콘텐츠로의 이동을 처리하는 핵심 요소

Navigation | 특징

애니메이션 및 전환	애니메이션 및 전환을 위한 표준화된 리소스를 제공함
딥 링크(Deep linking)	사용자가 특정 목적지로 바로 이동할 수 있도록 딥 링크를 구현하고 처리함 • 앱의 외부 링크(ex) 알림, 웹 링크 등)를 통해 앱의 특정 화면으로 바로 연결될 수 있음
UI 패턴	내비게이션 드로어, 하단 내비게이션 등 다양한 UI 패턴을 지원해주기에, 흔히 쓰는 UI를 쉽게 추가/관리할 수 있음
타입 안정성(Type safety)	목적지 간 데이터를 type safety하게 전달할 수 있도록 지원함
ViewModel 지원	내비게이션 그래프에 ViewModel을 스코핑하여 그래프의 여러 목적지 간 UI 관련 데이터를 공유할 수 있게 함 • 여러 화면이 동일한 데이터를 공유할 수 있도록 ViewModel을 연결해 데이터 일관성을 유지함
프래그먼트 트랜잭션	프래그먼트 트랜잭션을 완벽하게 지원하고 처리함
뒤로 및 위로 이동	기본적으로 뒤로 가기와 위로 가기 동작을 올바르게 처리함 • 앱이 예측 가능한 방식으로 동작하도록 자동으로 설정됨

Navigation | 사용법

NavController	앱의 화면간 **이동을 담당**함
NavGraph	이동할 컴포저블 대상을 **매핑**하는 역할
NavHost	NavGraph의 **현재 대상을 표시**하는 컨테이너 역할을 하는 컴포저블

1. build.gradle 파일에 라이브러리 추가하기

```
dependencies {
    val nav_version = "2.8.3"

    implementation("androidx.navigation:navigation-compose:$nav_version")
}
```

Navigation | 사용법

NavController	앱의 화면간 이동을 담당함
NavGraph	이동할 컴포저블 대상을 **매핑**하는 역할
NavHost	NavGraph의 현재 대상을 표시하는 컨테이너 역할을 하는 컴포저블

2. 코드 작성

```
val navController : NavHostController = rememberNavController()
```
객체 선언

처음 보여줄 화면 설정

```
NavHost(navController=navController, startDestination = "Home"){ this: NavGraphBuilder
    composable(route="Home"){ this: AnimatedContentScope it: NavBackStackEntry
        MainScreen1(navController)
    }
    composable(route="A"){ this: AnimatedContentScope it: NavBackStackEntry
        ScreenA1(navController)
    }
    composable(route="B"){ this: AnimatedContentScope it: NavBackStackEntry
        ScreenB1(navController)
    }
    composable(route="C"){ this: AnimatedContentScope it: NavBackStackEntry
        ScreenC1(navController)
    }
}
```

주어진 람다식으로 NavGraph 생성

식별하는 키

호출할 때는 아래처럼 작성해 사용

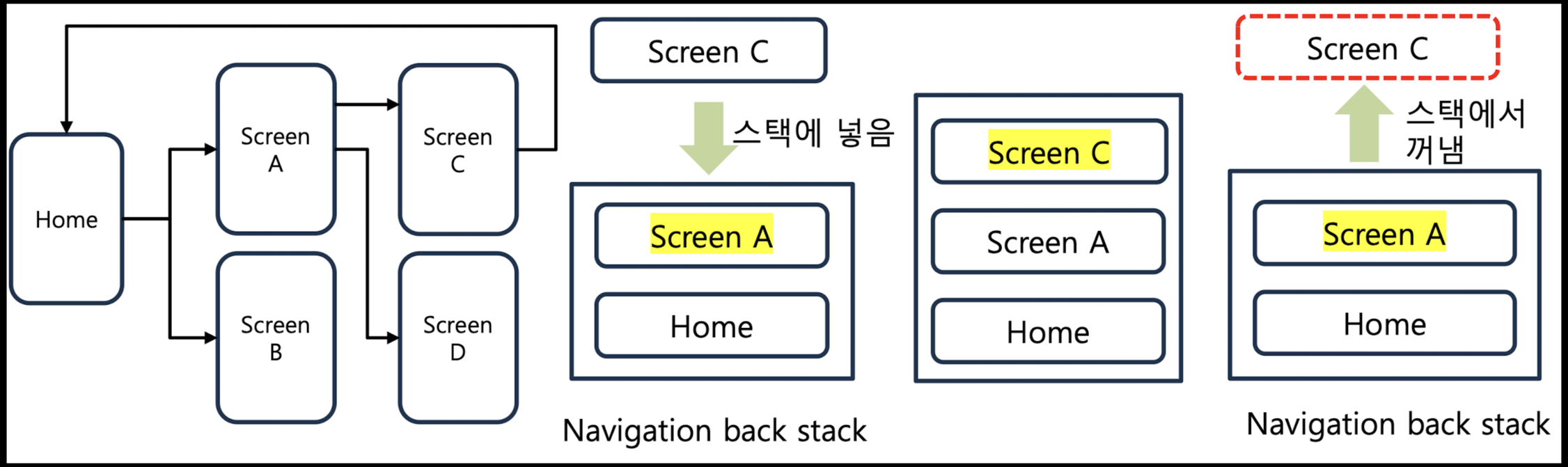
```
navController.navigate( route: "A")
```

NavController의 navigate 함수를 통해 화면 이동

+) Navigation Back Stack

Navigation Back Stack을 이용해 목적지에 이르는 경로를 추적함

- 화면을 이동할 때마다 화면들이 Navigation Back Stack에 쌓임
- Navigation Back Stack의 Top에 있는 화면 = 현재 목적지



+) Navigation Back Stack | 옵션

현재 백 스택 지우기 (popUpTo)

- 지정한 특정 화면 전까지의 모든 화면이 백 스택에서 제거됨
- ex) Home 전까지의 컴포저블을 꺼냄

```
navController.navigate(“C”){
    popUpTo(“Home”)
}
```

목적지 “C”로 이동하면서, ‘Home’ 화면 전까지의 모든 화면을 백 스택에서 제거함

백 스택 모두 지우기 (inclusive)

- popUpTo로 지정한 화면을 포함해 해당 화면까지의 모든 화면이 백 스택에서 제거됨
- ex) Home을 포함한 컴포저블을 꺼냄

```
navController.navigate(“C”){
    popUpTo(“Home”){inclusive = true}
}
```

목적지 “C”로 이동하면서, ‘Home’ 화면을 포함해 이전의 모든 화면을 백 스택에서 제거함

백 스택 상단에 복사본 방지하기 (launchSingleTop)

- 해당 목적지가 이미 back stack의 top에 있는 경우, 동일한 화면을 복사하지 않고 기존 인스턴스를 그대로 재사용함
- ex) 목적지 “C”의 중복 생성 방지

```
navController.navigate(“C”){
   launchSingleTop = true
}
```

목적지 “C”가 이미 back stack의 top에 있는 경우,
새로운 인스턴스를 생성하지 말고 기존 “C” 화면을 그대로 사용해라

State

상태

State | 정의 및 예시

State in an app is any value that can change over time.

시간이 지남에 따라 변할 수 있는 값

매우 광범위한 정의로서, Android app state의 예시는 다음과 같음

- 네트워크 연결을 설정할 수 없을 때 표시되는 스낵바
- 블로그 게시물 및 관련 댓글
- 사용자가 클릭하면 버튼에서 재생되는 물결 애니메이션
- 사용자가 이미지 위에 그릴 수 있는 스티커

State | 특징

- Jetpack Compose ⇒ 상태 기반 UI 라이브러리임

- 따라서, Jetpack Compose에서 State와 UI는 서로 밀접한 관계를 갖고 있음
  - State가 변경될 때마다 Compose가 이를 감지하고, 자동으로 UI를 업데이트함 (Recomposition)
  - 이때, Compose는 필요한 컴포저블 함수만 다시 그려 성능을 최적화함
    (전체 UI가 아닌, 변경된 State에 연결된 컴포저블만 재구성된다는 뜻)

State | 사용법 - state를 저장하는 방법

- 기본 데이터 타입은 자동으로 Bundle에 저장됨

- 기본 타입이 아닌 경우 아래와 같은 Custom Saver를 제공해야 함
  - Parcelize
  - listSaver
  - mapSaver

State | 사용법 - state를 저장하는 방법

Parcelize

- 객체에 @Parcelize 주석을 추가 → 객체가 parceable이 되며 번들로 제공될 수 있음
- ex) 예시 코드에서는, parceable City 데이터 유형을 만들어 상태를 저장하고 있음

```
@Parcelize
data class City(val name: String, val country: String) : Parcelable

@Composable
fun CityScreen() {
    var selectedCity = rememberSaveable {
        mutableStateOf(City("Madrid", "Spain"))
    }
}
```

State | 사용법 - state를 저장하는 방법

ListSaver

- map의 key를 정의할 필요가 없으며, 대신 index를 사용함
- ex) 예시 코드에서는, 값을 저장할 때 save 함수(City 객체 → List 형식)를 호출, 값을 복원할 때 restore 함수(List 형식 → City 객체)를 호출하고 있음

```
data class City(val name: String, val country: String)

val CitySaver = listSaver<City, Any>(
    save = { listOf(it.name, it.country) },
    restore = { City(it[0] as String, it[1] as String) }
)

@Composable
fun CityScreen() {
    var selectedCity = rememberSaveable(stateSaver = CitySaver) {
        mutableStateOf(City("Madrid", "Spain"))
    }
}
```

State | 사용법 - state를 저장하는 방법

MapSaver

- 시스템이 Bundle에 저장할 수 있는 값 집합으로 객체를 변환하는 고유한 규칙을 정의함
- ex) 예시 코드에서는, 값을 저장할 때 save 함수(City 객체 → Map 형식)를 호출, 값을 복원할 때 restore 함수(Map 형식 → City 객체)를 호출하고 있음

```
data class City(val name: String, val country: String)

val CitySaver = run {
    val nameKey = "Name"
    val countryKey = "Country"
    mapSaver(
        save = { mapOf(nameKey to it.name, countryKey to it.country) },
        restore = { City(it[nameKey] as String, it[countryKey] as String) }
    )
}

@Composable
fun CityScreen() {
    var selectedCity = rememberSaveable(stateSaver = CitySaver) {
        mutableStateOf(City("Madrid", "Spain"))
    }
}
```

참고 문헌

Android Developers | Navigation with Compose
https://developer.android.com/develop/ui/compose/navigation#kts

Android Developers | Navigation
https://developer.android.com/guide/navigation

Android Developers | State and Jetpack Compose
https://developer.android.com/develop/ui/compose/state

모바일프로그래밍 | 05주차 강의자료

모바일프로그래밍 | 07주차 강의자료

감사합니다 :)

2024.11.13.Wed | Android Compose Study

지영