02

02-1. SharedPreference

데이터 영속화

17-3 공유된 프리퍼런스에 보관하기

공유된 프리퍼런스 이용하기

- 공유된 프리퍼런스는 플랫폼 API에서 제공하는 클래스로, 데이터를 키-값 형태로 저장
- SharedPreferences 객체를 얻는 방법은 다음 2가지를 제공
  - Activity.getPreferences(int mode)
  - Context.getSharedPreferences(String name, int mode)
- Activity.getPreferences() 함수는 액티비티 단위로 데이터를 저장할 때 사용

• 액티비티의 데이터 저장

```
val sharedPref = getPreferences(Context.MODE_PRIVATE)
```

- 앱 전체의 데이터를 키-값 형태로 저장하려고 SharedPreferences 객체를 얻을 때는 Context.getSharedPreferences() 함수를 이용

• 앱 전체의 데이터 저장

```
val sharedPref = getSharedPreferences("my_prefs", Context.MODE_PRIVATE)
```

17-3 공유된 프리퍼런스에 보관하기

- 데이터를 저장하려면 다음과 같은 SharedPreferences.Editor 클래스의 함수를 이용
  - putBoolean(String key, boolean value)
  - putInt(String key, int value)
  - putFloat(String key, float value)
  - putLong(String key, long value)
  - putString(String key, String value)

• 프리퍼런스에 데이터 저장

```
sharedPref.edit().run {
    putString("data1", "hello")
    putInt("data2", 10)
    commit()
}
```

17-3 공유된 프리퍼런스에 보관하기

- 데이터를 저장을 위해 최종 commit() 혹은 apply() 함수 호출한다.

특성	commit()	apply()
실행 타입	동기식(Synchronous)	비동기식(Asynchronous)
반환 값	성공/실패 여부(boolean)	없음(void)
메인 스레드	블로킹	블로킹하지 않음
디스크 쓰기	즉시 실행	백그라운드에서 실행
메모리 캐시	즉시 업데이트	즉시 업데이트

- 대부분의 경우 apply() 함수로 데이터를 저장하며 데이터 저장의 성공 여부를 확인해야 하는 경우에 한해서 commit() 함수를 이용한다.

17-3 공유된 프리퍼런스에 보관하기

- 저장된 데이터를 가져오려면 SharedPreferences의 게터 함수를 이용
  - getBoolean(String key, boolean defValue)
  - getFloat(String key, float defValue)
  - getInt(String key, int defValue)
  - getLong(String key, long defValue)
  - getString(String key, String defValue)

• 프리퍼런스에서 데이터 가져오기

```
val data1 = sharedPref.getString("data1", "world")
val data2 = sharedPref.getInt("data2", 10)
```

DataStore

- Preferences DataStore : 코루틴과 Flow를 사용한 비동기 API

특성	SharedPreferences	Preferences DataStore
API 타입	동기식 (+ 비동기 콜백)	완전 비동기식 (코루틴 기반)
메인 스레드 안전성	위험 (commit() 사용 시 블로킹)	안전 (메인 스레드 블로킹 없음)
데이터 일관성	부분적 보장	트랜잭션 기반으로 완전 보장
다중 프로세스	지원 부족 (충돌 가능)	안정적 지원
업데이트 감지	리스너 패턴 (콜백)	Flow를 통한 반응형 스트림
에러 처리	제한적	구조화된 예외 처리
타입 안전성	없음	키 정의를 통한 기본적 타입 안전성

DataStore

```
implementation("androidx.datastore:datastore-preferences:1.1.4")
```

- DataStore 객체 획득

```
val dataStore: DataStore<Preferences> by preferencesDataStore(name = "my_prefs")
```

- 키 준비
- API 함수를 이용해 키를 준비해야 한다. 문자열로 키를 직접 지정하는 것은 허용하지 않는다.
- stringPreferencesKey("key")
- intPreferencesKey("key")
- doublePreferencesKey("key")
- floatPreferencesKey("key")
- booleanPreferencesKey("key")
- longPreferencesKey("key")
- stringSetPreferencesKey("key") : Set<String> 타입의 데이터를 저장할 수 있는 키

```
val USER_NAME = stringPreferencesKey("user_name")
```

DataStore

- 데이터 저장
- 내부적으로 코루틴에 의해 실행됨으로 코루틴 API 와 같이 사용해야 한다.

```
dataStore.edit { preferences ->
    preferences[USER_NAME] = "kim"
}
```

- 데이터 획득

```
dataStore.data.collect {
    val userName = it[USER_NAME] ?: ""
}
```

감사합니다

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

윌리엄 셰익스피어
William Shakespeare