02

02-2. 앱 설정 구현

데이터 영속화

17-3 공유된 프리퍼런스에 보관하기

앱 설정 화면 만들기

- 플랫폼 API에서 이처럼 앱의 설정 기능을 자동화해주는 API는 많았지만 안드로이드 10 버전(API 레벨 29)부터 모두 deprecated
- AndroidX의 Preference를 이용할 것을 권장

• AndroidX의 프리퍼런스 사용 선언

```
implementation("androidx.preference:preference-ktx:1.2.1")
```

17-3 공유된 프리퍼런스에 보관하기

- 프리퍼런스 이용 방법
  - res/xml 디렉터리에 설정과 관련된 XML 파일을 만들어야 합니다.
  - 루트 태그가 <PreferenceScreen>
  - 하위에 <SwitchPreferenceCompat>, <Preference> 등의 태그를 이용해 설정 항목을 준비

• 설정 XML 파일

```
<PreferenceScreen xmlns:app="http://schemas.android.com/apk/res-auto">
    <SwitchPreferenceCompat
        app:key="notifications"
        app:title="Enable message notifications" />
    <Preference
        app:key="feedback"
        app:title="Send feedback"
        app:summary="Report technical issues or suggest new features" />
</PreferenceScreen>
```

17-3 공유된 프리퍼런스에 보관하기

- 설정 항목의 key 속성값이 데이터의 키
- title 속성은 설정 화면에 출력되는 문자열
- XML 파일을 코드에서 적용해야 하는데 이때 PreferenceFragmentCompat 클래스를 이용
- PreferenceFragmentCompat을 상속받은 프래그먼트 클래스는 onCreatePreferences() 함수를 재정의해서 작성

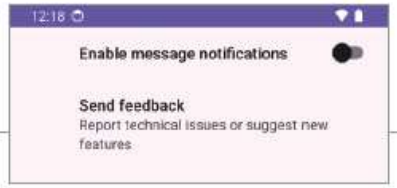
• 설정 XML 파일 적용

```
class MySettingFragment : PreferenceFragmentCompat() {
    override fun onCreatePreferences(savedInstanceState: Bundle?, rootKey: String?) {
        setPreferencesFromResource(R.xml.settings, rootKey)
    }
}
```

• 액티비티에서 프래그먼트 출력

```
<androidx.fragment.app.FragmentContainerView xmlns:android="http://schemas.android.com/
apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/setting_content"/>
```

▶ 실행 결과



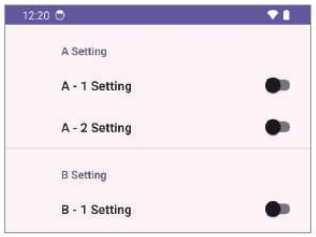
17-3 공유된 프리퍼런스에 보관하기

- 설정 화면 구성
  - <PreferenceCategory> 태그를 이용하면 한 화면에 보이는 항목끼리 구분 지어 출력

• 항목끼리 묶기

```
<PreferenceScreen xmlns:app="http://schemas.android.com/apk/res-auto">
    <PreferenceCategory
        app:key="a_category"
        app:title="A Setting">
        <SwitchPreferenceCompat
            app:key="a1"
            app:title="A - 1 Setting" />
        <SwitchPreferenceCompat
            app:key="a2"
            app:title="A - 2 Setting" />
    </PreferenceCategory>
    <PreferenceCategory
        app:key="B_category"
        app:title="B Setting">
        <SwitchPreferenceCompat
            app:key="b1"
            app:title="B - 1 Setting" />
    </PreferenceCategory>
</PreferenceScreen>
```

▶ 실행 결과



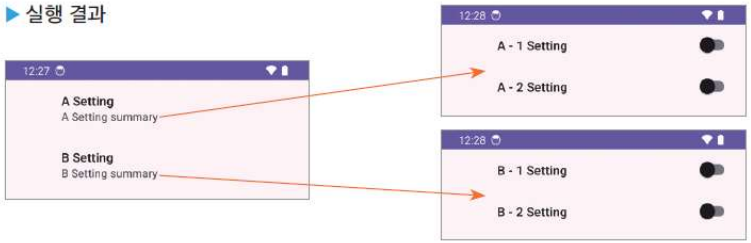
17-3 공유된 프리퍼런스에 보관하기

- 설정 항목이 더 많을 때는 화면을 여러 개로 분리하는 방법
- XML에서 각 설정 화면은 <Preference> 태그로 지정

• 두 화면을 포함하는 설정 화면

```
<PreferenceScreen xmlns:app="http://schemas.android.com/apk/res-auto">
    <Preference
        app:key="a"
        app:summary="A Setting summary"
        app:title="A Setting"
        app:fragment="com.example.test17.ASettingFragment" />
    <Preference
        app:key="b"
        app:summary="B Setting summary"
        app:title="B Setting"
        app:fragment="com.example.test17.BSettingFragment" />
</PreferenceScreen>
```

▶ 실행 결과



17-3 공유된 프리퍼런스에 보관하기

- <Preference> 태그를 이용해 설정 화면을 분할했다면 액티비티에서 PreferenceFragmentCompat.OnPreferenceStartFragmentCallback 인터페이스를 구현하고 onPreferenceStart Fragment() 함수를 재정의해서 작성
- onPreferenceStartFragment()는 설정 화면이 바뀔 때마다 호출되는 함수

• 분할 설정 화면을 보여주는 액티비티 코드

```
class SettingActivity : AppCompatActivity(),
    PreferenceFragmentCompat.OnPreferenceStartFragmentCallback {
    (... 생략 ...)
    override fun onPreferenceStartFragment(caller: PreferenceFragmentCompat,
                                           pref: Preference
    ): Boolean {
        // 새로운 프래그먼트 인스턴스화
        val args = pref.extras
        val fragment = supportFragmentManager.fragmentFactory.instantiate(
            classLoader,
            pref.fragment as String)
        fragment.arguments = args
        supportFragmentManager.beginTransaction()
            .replace(R.id.setting_content, fragment)
            .addToBackStack(null)
            .commit()
        return true
    }
}
```

17-3 공유된 프리퍼런스에 보관하기

- 메인 설정 화면에서 인텐트를 이용해 하위 설정 화면을 띄우는 방법

• 인텐트로 설정 화면 실행

```
<Preference
    app:key="activity"
    app:title="Launch activity">
    <intent
        android:targetClass="com.example.test17.SomeActivity"
        android:targetPackage="com.example.test17" />
</Preference>
```

17-3 공유된 프리퍼런스에 보관하기

- <intent> 태그 하위에 <extra> 태그로 인텐트에 포함해서 전달할 엑스트라 데이터를 설정
- 명시적 인텐트 정보뿐만 아니라 암시적 인텐트 정보도 설정할 수 있습니다.

• 인텐트에 엑스트라 데이터 포함

```
<intent
    android:targetClass="com.example.test17.SomeActivity"
    android:targetPackage="com.example.test17">
    <extra
        android:name="example_key"
        android:value="example_value" />
</intent>
```

• 암시적 인텐트 사용

```
<intent
    android:action="android.intent.action.VIEW"
    android:data="http://www.google.com" />
```

17-3 공유된 프리퍼런스에 보관하기

- 설정 제어
  - 사용자가 설정 항목을 클릭한 순간의 이벤트를 처리하거나 설정값을 설정 항목 옆에 나타나게 하는 방법
  - 설정 항목에 해당하는 객체를 findPreference() 함수로 얻어야 합니다.

• 글을 입력받는 설정

```
<EditTextPreference
    app:key="id"
    app:title="ID 설정"
    app:isPreferenceVisible="false" />
```

• 설정값을 코드에서 바꾸기

```
override fun onCreatePreferences(savedInstanceState: Bundle?, rootKey: String?) {
    setPreferencesFromResource(R.xml.settings, rootKey)
    val idPreference: EditTextPreference? = findPreference("id")
    idPreference?.isVisible = true

    idPreference?.summary="code summary"
    idPreference?.title="code title"
}
```

17-3 공유된 프리퍼런스에 보관하기

- <EditTextPreference>나 제시한 목록에서 선택하는 <ListPreference>는 설정값을 summary 속성에 자동으로 지정
- SimpleSummaryProvider를 사용

• 설정 XML 예

```
<EditTextPreference
    app:key="id"
    app:title="ID 설정"/>

<ListPreference
    app:key="color"
    app:title="색상 선택"
    android:entries="@array/my_color"
    app:entryValues="@array/my_color_values" />
```

• 설정값 자동 적용

```
val idPreference: EditTextPreference? = findPreference("id")
val colorPreference: ListPreference? = findPreference("color")

idPreference?.summaryProvider =
    EditTextPreference.SimpleSummaryProvider.getInstance()
colorPreference?.summaryProvider =
    ListPreference.SimpleSummaryProvider.getInstance()
```

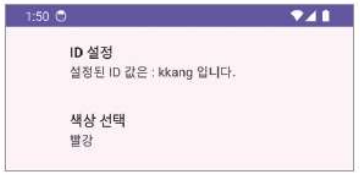
17-3 공유된 프리퍼런스에 보관하기

- SummaryProvider의 하위 클래스를 만들어 코드에서 원하는 대로 summary가 지정되게 할 수도 있습니다.

• 코드에서 설정값 표시하기

```
idPreference?.summaryProvider =
    Preference.SummaryProvider<EditTextPreference> { preference ->
        val text = preference.text
        if (TextUtils.isEmpty(text)) {
            "설정이 되지 않았습니다. "
        } else {
            "설정된 ID 값은 : $text 입니다."
        }
    }
```

▶ 실행 결과



17-3 공유된 프리퍼런스에 보관하기

- 설정 항목에 이벤트를 추가

• 이벤트 핸들러 지정

```
idPreference?.setOnPreferenceClickListener { preference ->
    Log.d("kkang", "preference key : ${preference.key}")
    true
}
```

17-3 공유된 프리퍼런스에 보관하기

- 설정한 값 가져오기
  - 설정값을 가져올 때는 PreferenceManager.getDefaultSharedPreferences() 함수를 이용

• 설정값 가져오기

```
val sharedPreferences = PreferenceManager.getDefaultSharedPreferences(activity)
val id = sharedPreferences.getString("id", "")
```

17-3 공유된 프리퍼런스에 보관하기

- 설정 변경 순간 감지
  - Preference.OnPreference ChangeListener를 이용하는 방법 은 프리퍼런스 객체마다 이벤트 핸들러를 직접 지정하여 객체의 설정 내용이 변경되는 순간의 이벤트를 처리
  - SharedPreferences.OnSharedPrefe renceChangeListener를 이용하는 방법은 설정 객체의 변경을 하나의 이벤트 핸들러에서 처리

• 프리퍼런스를 이용한 이벤트 처리

```
idPreference?.setOnPreferenceChangeListener { preference, newValue ->
    Log.d("kkang", "preference key : ${preference.key}, newValue : $newValue")
    true
}
```

17-3 공유된 프리퍼런스에 보관하기

- SharedPreferences.OnSharedPreferenceChangeListener를 이용

• 공유된 프리퍼런스를 이용한 이벤트 처리

```
class MySettingFragment : PreferenceFragmentCompat(), SharedPreferences.
OnSharedPreferenceChangeListener {
    (... 생략 ...)
    override fun onSharedPreferenceChanged(sharedPreferences:
                                           SharedPreferences?, key: String?) {
        if (key == "id") {
            Log.i("kkang", "newValue : " + sharedPreferences?.getString("id", ""))
        }
    }
    override fun onResume() {
        super.onResume()
        preferenceManager.sharedPreferences
            ?.registerOnSharedPreferenceChangeListener(this)
    }
    override fun onPause() {
        super.onPause()
        preferenceManager.sharedPreferences
            ?.unregisterOnSharedPreferenceChangeListener(this)
    }
}
```

감사합니다

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

윌리엄 셰익스피어
William Shakespeare