

# SavedState ViewModelを使ってデータを保持する



Kenji Abe Following  
Mar 16, 2019 · 4 min read



ViewModel を使うことで画面回転等のConfigurationが変更されたときもデータを保持することが可能になりました。しかし、OSからプロセスKillされてActivityの再生成に対応できません。この場合は `onSaveInstanceState` を使うことで対応することになります。

ほとんどケースでUIの状態はViewModelと多いので、このあたりもViewModelのほうで対応できたらだいぶ楽になると思います。

これを `SavedStateHandle` というのをを使って保存と取得を行えるようになりました。

まだalpha01なので今後変更はあるかもしれませんが。

## 環境

```
implementation 'androidx.lifecycle:lifecycle-viewmodel-savedstate:1.0.0-alpha01'
```

## 実装

### ViewModel定義

ViewModel自体はコンストラクタに `SavedStateHandle` もらうようにするだけです。

### ViewModel生成

普通のViewModelの生成と変わらないのですが、Factoryに `SavedStateVMFactory` を使います。

これだけで準備完了です。

### データを保存と取得

保存と取得は非常に簡単で、`SavedStateHandle` を上記のように使うだけです。これでプロセスがKillされてもsetした値はActivity再生成後も取得できます。

### LiveDataを保存と取得

`SavedStateHandle` は更にLiveDataに対応しています。

`getLiveData` を使うことでLiveDataとして取得することが可能です。更に値をsetしたときには`setValue`と同じで、`observe`へ通知がいきます。

**注意:** コードを見る限りLiveData#setValueを使ってるのでBackgroundスレッド等からは呼び出さないようにしてください。

### 扱えるデータ型

`Bundle` に保存できるものと同じです。

## おまけ: Daggerを使う場合

Daggerを使う場合はいつものようなViewModelFactoryが使えません。ちょっと別のアプローチが必要です。

例えば、次のようにViewModleにRepositoryを渡したい場合です。

`SavedStateHandle` は更にLiveDataに対応しています。

`getLiveData` を使うことでLiveDataとして取得することが可能です。更に値をsetしたときには`setValue`と同じで、`observe`へ通知がいきます。

**注意:** コードを見る限りLiveData#setValueを使ってるのでBackgroundスレッド等からは呼び出さないようにしてください。

### 扱えるデータ型

`Bundle` に保存できるものと同じです。

## おまけ: Daggerを使う場合

Daggerを使う場合はいつものようなViewModelFactoryが使えません。ちょっと別のアプローチが必要です。

例えば、次のようにViewModleにRepositoryを渡したい場合です。

```
1 class MainViewModel(  
2     private val savedStateHandle: SavedStateHandle,  
3     private val userRepository: UserRepository  
4 ) : ViewModel() {  
5  
6 }  
MainViewModel.kt hosted with ❤️ by GitHub view raw
```

## おまけ: Daggerを使う場合

Daggerを使う場合はいつものようなViewModelFactoryが使えません。ちょっと別のアプローチが必要です。

例えば、次のようにViewModleにRepositoryを渡したい場合です。

```
1 class MainViewModel(  
2     private val savedStateHandle: SavedStateHandle,  
3     private val userRepository: UserRepository  
4 ) : ViewModel() {  
5  
6 }  
MainViewModel.kt hosted with ❤️ by GitHub view raw
```

この場合はViewModelごとにFactoryを作る必要があります。以下のような感じです。

```
1 class MainViewModelFactory @Inject constructor(  
2     owner: SavedStateRegistryOwner,  
3     private val userRepository: UserRepository  
4 ) : AbstractSavedStateVMFactory(owner, null) {  
5  
6     override fun <T : ViewModel?> create(key: String, modelClass: Class<T>, handle: SavedStateHandle): T {  
7         @Suppress("UNCHECKED_CAST")  
8         return MainViewModel(handle, userRepository) as T  
9     }  
10 }  
MainViewModelFactory.kt hosted with ❤️ by GitHub view raw
```

`AbstractSavedStateVMFactory` を継承したFactoryを作ります。このコンストラクタには、`SavedStateRegistryOwner` が必要になります。これはActivityとFragmentが実装してるので、それを渡せば大丈夫です。

Moduleでその `SavedStateRegistryOwner` を解決できるようにしてあげます。

```
1 @Module  
2 class FeatureModule(private val owner: SavedStateRegistryOwner) {  
3     @FeaturesScope  
4     @Provides  
5     fun provideOwner() = owner  
6 }  
FeatureModule.kt hosted with ❤️ by GitHub view raw
```

実際は他にも色々必要ですが、最終的にActivityはこんな感じになります。

```
1 class MainActivity : AppCompatActivity() {  
2  
3     @Inject  
4     lateinit var viewModelFactory: MainViewModelFactory  
5  
6     private val viewModel by lazy {  
7         // injectされたViewModelFactoryを使ってViewModelを作る  
8         ViewModelProviders.of(this, viewModelFactory).get(MainViewModel::class.java)  
9     }  
10  
11     override fun onCreate(savedInstanceState: Bundle?) {  
12         super.onCreate(savedInstanceState)  
13         setContentView(R.layout.activity_main)  
14  
15         val appComponent = (application as App).appComponent  
16         val featureComponent = DaggerFeatureComponent.builder()  
17             .appComponent(appComponent)  
18             .featureModule(FeatureModule(this))  
19             .build()  
20         // ViewModelFactoryをinject  
21         featureComponent.inject(this)  
22  
23         viewModel.value.observe(this) {  
24             // ...  
25         }  
26     }  
27 }
```

## 参考

### Saved State module for ViewModel | Android Developers

Module for handling Saved State in ViewModel objects  
[developer.android.com](#)

### Android lifecycle-aware components codelab

Architecture components are a set of Android libraries that help you structure your app in a way that is robust...

[codelabs.developers.google.com](#)



Android

38 claps



WRITTEN BY

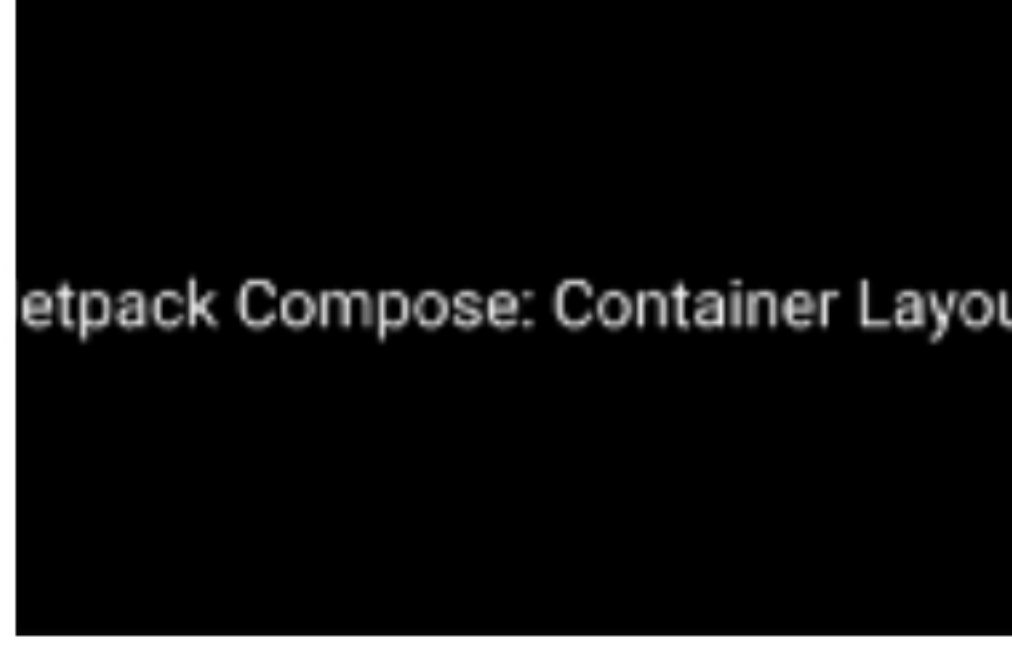
**Kenji Abe**

Programmer / Gamer / Google Developers Expert for Android,  
Kotlin / @STAR\_ZERO

Following

### More From Medium

Related reads

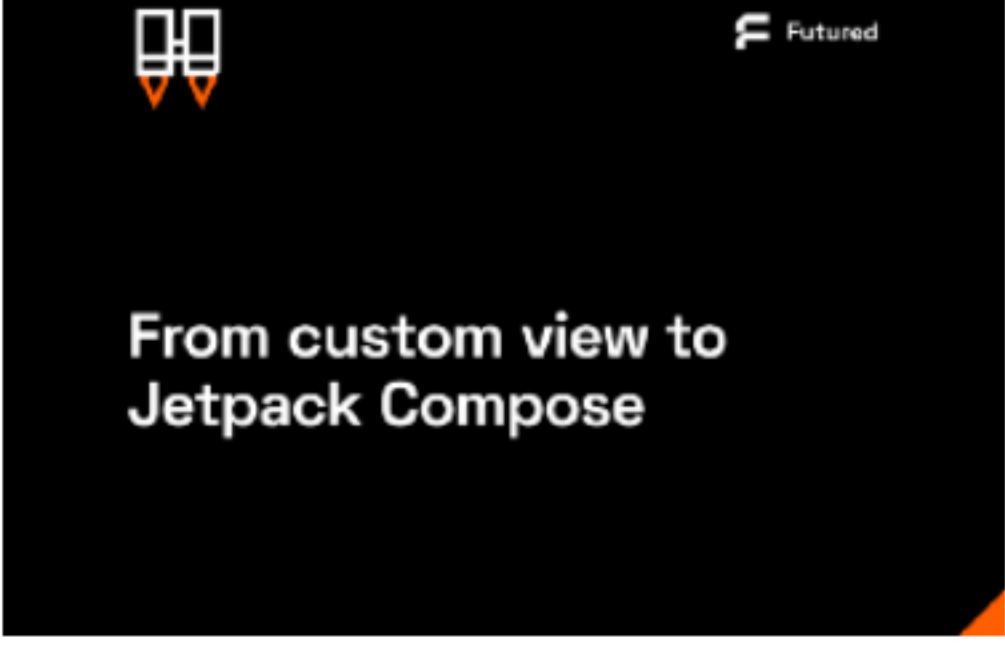


**Jetpack Compose: Container Layout**

Ahmed Rizwan in Level Up...  
Apr 5 · 4 min read

196

Related reads



**From custom view to Jetpack Compose**

Martin Sumera in Futured apps  
May 27 · 10 min read ★

162

Related reads



**Android build and the journey to the end game**

Tuan Kiet in ProAndroidDev  
Sep 1 · 7 min read

210

### Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. [Watch](#)

### Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. [Explore](#)

### Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just \$5/month. [Upgrade](#)