Kotlin的协程与生命周期【转载】


前小小 LV.4
2022年08月03日 21:41 · 阅读 46

关注

文章目录

一、前言

kotlin的协程在有时候还是挺方便的。jetpack提供了足够多的兼容，这里对其部分进行记录

二、引入依赖

arduino 复制代码

```
implementation 'androidx.core:core-ktx:1.7.0'
implementation 'androidx.fragment:fragment-ktx:1.5.0'
implementation 'androidx.activity:activity-ktx:1.5.0'
implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.1'
 implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.5.1'
```

三、代码示例

1、基础用法

凡是实现了 `LifecycleOwner` 接口的类都可以使用以下方式开启协程，例如 `Fragment` 、`ComponentActivity` 及其子类

arduino 复制代码

```
//可以使用以下方式开启协程度，优势在于自己不要处理页面结束后的关闭问题
    lifecycleScope.launch {
}
```

`ViewModel`

markdown 复制代码

```
viewModelScope.launch {

    }
```

2、repeatOnLifecycle

假如有一种需求需要根据页面的生命周期进行处理，比如说 `onStart` 时候启动协程， `onStop` 时候停止协程，可以使用以下方式。这种生命周期被称为**可重启生命周期感知型协程**

kotlin 复制代码

```
override fun onCreate(savedInstanceState: Bundle?) {
        lifecycleListener()
}
    private fun lifecycleListener(){
        lifecycleScope.launch {
            repeatOnLifecycle(Lifecycle.State.STARTED){
                while (isActive){
                    delay(1000)
                    Log.e("YM---->","线程运行中....")
                }
            }
        }
    }
```

需要注意的是官方曾经提过一个函数 `addRepeatingJob` 和这个函数类似，但是 `addRepeatingJob` 已经在 `lifecycle-runtime-ktx:2.4.0-alpha01` 中存在，在 `lifecycle-runtime-ktx:2.4.0-alpha02` 时候已经移除了。参考链接如下:

repeatOnLifecycle API design story

3、flowWithLifecycle

如果是只是对一个数据流 `Flow` 进行监听，可以使用以下方式

kotlin 复制代码

```kotlin
    private fun lifecycleListener(){
        lifecycleScope.launch {
            (1..9).asFlow().flowWithLifecycle(lifecycle,Lifecycle.State.STARTED).collect{
                println("YM---->value:$it")
            }

        }
    }
```

由于 `flowWithLifecycle` 是 `Flow` 的函数，所以可以将其挂在在热流 `StateFlow` 或 `ShareFlow` 上面。多个流的话只能还使用上述 `repeatOnLifecycle` 的方式

4、lifecycle.whenCreated、lifecycle.whenStarted 和 lifecycle.whenResumed

`repeatOnLifecycle` 可以将活动限制在某个范围之内，但是这个会随着生命周期的反复来进行重复启动，但是假如，只使用一次的话则无需如此，可以使用以下方式,

kotlin 复制代码

```kotlin
class MyFragment: Fragment {
    init {
        lifecycleScope.launch {
            whenStarted {}
        }
        //或者
        lifecycleScope.launchWhenCreated {

        }
    }
}
```

5、协程与LiveData

在使用 `LiveData` 时候，我们有时候需要异步获取内容，通常来说是创建 `LiveData` ，然后在 `onCreate()` 中去主动加载数据，以下方式可以简化这一操作

kotlin 复制代码

```kotlin
    private val user: LiveData<Int> = liveData {
//        val data = database.loadUser() // loadUser is a suspend function.
        //delay(5000)
        emit(10)
    }
 override fun onCreate(savedInstanceState: Bundle?) {
        lifecycleListener()
 }
 private fun lifecycleListener(){
        user.observe(this){
            Log.e("YM---->"," 参数:$it")
        }
    }
```

以下引用自官网

> 当 LiveData 变为活动状态时，代码块开始执行；当 LiveData
> 变为非活动状态时，代码块会在可配置的超时过后自动取消。如果代码块在完成前取消，则会在 LiveData
> 再次变为活动状态后重启；如果在上次运行中成功完成，则不会重启。请注意，代码块只有在自动取消的情况下才会重启。如果代码块由于任何其他原因（例如，抛出 CancellationException）而取消，则不会重启。

当页面进入 `onStop` 时候，数据将不在发送直到重新恢复页面，才会发送数据，**需要注意的是，实际测试中，逻辑并不会重新执行，而是在下次恢复时候把已经加载出来的数据直接发送**

四、参考链接

1. kotlin协程Flow的StateFlow和SharedFlow(十二)
2. 将 Kotlin 协程与生命周期感知型组件一起使用
3. Jetpack之LifeCycle
4. kotlin协程Flow的StateFlow和SharedFlow(十二)

本文转自 blog.csdn.net/Mr_Tony/art...，如有侵权，请联系删除。

分类： Android 标签： Android