链滴

Android WorkManager 定时任务

作者：RustFisher

原文链接：https://ld246.com/article/1630297950594

来源网站：链滴

App有时可能需要定期运行某些工作。例如，可能要定期备份数据、上传信息到服务器，定期获取新内容等等。

在app运行期间，我们使用Handler也可以完成定期的功能。在这里我们介绍**WorkManager**使用定任务的方法。

前面我们介绍了**WorkManager**的使用方法，约束和延迟等。本文介绍**WorkManager**的定时任务如何创建定时任务，查看任务状态，取消任务。

本文使用kotlin

注意：可以定义的最短重复间隔是 15 分钟（与 JobScheduler API 相同）。

gradle

kotlin使用**PeriodicWorkRequestBuilder**时候报错误提示

Cannot inline bytecode built with JVM target 1.8 into bytecode that is being built with JVM target 1.6.

Please specify proper '-jvm-target' option

根据提示，在app的gradle配置添加kotlinOptions的jvmTarget

```
android {
    // 其他配置...

    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

创建定时任务

创建定时任务，用到**PeriodicWorkRequestBuilder**，传入定时的参数，build()得到任务实例。

交给**WorkManager**的 enqueue方法即可。

```
val r1 = PeriodicWorkRequestBuilder<UploadWorker2>(15, TimeUnit.MINUTES)
        .addTag("r1").build()
WorkManager.getInstance(applicationContext).enqueue(r1)
```

上面的代码，enqueue后会立刻执行一次任务。

多次创建任务，可以得到多个定时任务。

注意：可以定义的最短重复间隔是 15 分钟（与 JobScheduler API 相同）。

单一任务

如果不想要重复的定时任务，需要用**WorkManager**的 enqueueUniquePeriodicWork方法。

```
val r1 = PeriodicWorkRequestBuilder<UploadWorker2>(15, TimeUnit.MINUTES)
        .addTag("r2").build()
WorkManager.getInstance(applicationContext)
```

```
    .enqueueUniquePeriodicWork(
            "单独的定时任务r2",
            ExistingPeriodicWorkPolicy.KEEP,
            r1)
```

用**PeriodicWorkRequestBuilder**创建出任务后。

调用 enqueueUniquePeriodicWork，此方法需要3个参数：

- uniqueWorkName 唯一的任务名字
- ExistingPeriodicWorkPolicy 发现任务重复时的处理方法
  - REPLACE 把旧的任务停止并删除，然后插入新的任务
  - KEEP 保留原来的任务，不新增任务
- PeriodicWorkRequest 任务对象

enqueueUniquePeriodicWork方法能保证1个名字同时只有一个定时任务（PeriodicWorkRequest
。

如果是同样的名字（uniqueWorkName），插入任务时可能会替换旧任务（REPLACE），或者不影
旧任务（KEEP）。

查看任务

在这里我们用 getWorkInfosByTag来查询任务

```
val status = WorkManager.getInstance(applicationContext).getWorkInfosByTag("r1")
val workInfoList: List<WorkInfo> = status.get()
for (w in workInfoList) {
    Log.d("rustfisher.com", " $w")
}
```

查询结果的例子

```
WorkInfo{mId='83d7d512-8a5d-4613-acbb-e73ee2855212', mState=ENQUEUED, mOutputDa
a=Data {}, mTags=[com.rustfisher.tutorial2020.workmanaer.WorkManagerAct$UploadWorker
, r1], mProgress=Data {}}
```

取消任务

我们有多种取消任务的方法。

取消所有任务

cancelAllWork()，取消所有任务

```
WorkManager.getInstance(applicationContext).cancelAllWork()
```

取消单独的任务

cancelUniqueWork(uniqueWorkName: String)，取消单独的任务，传入 uniqueWorkName

WorkManager.getInstance(applicationContext).cancelUniqueWork("单独的定时任务r2")

取消传入tag的所有任务

cancelAllWorkByTag(tag)，取消传入tag的所有任务

WorkManager.getInstance(applicationContext).cancelAllWorkByTag("r1")

取消特定UUID的任务

cancelWorkById(UUID)，取消特定UUID的任务

WorkManager.getInstance(applicationContext).cancelWorkById(UUID)

取消任务后，再去查询任务状态，会发现 mState=CANCELLED

WorkInfo{mId='7c9e0deb-5267-4ade-8b95-c695e57f274c', mState=CANCELLED, mOutputData=Data {}, mTags=[r2, com.rustfisher.tutorial2020.workmanaer.WorkManagerAct$UploadWorer2], mProgress=Data {}}

参考

- WorkManager使用入门 https://an.rustfisher.com/android/jetpack/workManager/use1/
- WorkManager工作约束，延迟与查询工作 https://an.rustfisher.com/android/jetpack/workManager/use2/
- 定期工作 https://developer.android.com/topic/libraries/architecture/workmanager/how-to define-work#schedule_periodic_work
- 观察工作 https://developer.android.com/topic/libraries/architecture/workmanager/how-to managing-work#observing