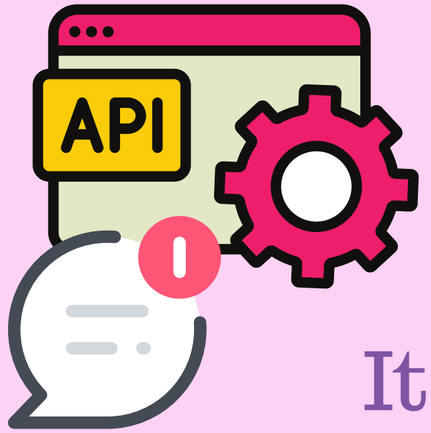
WORK MANAGER

Jetpack Library



WHY WORKMANAGER?

It is an API used for background processing

We can create a notification, numerical operation etc.

We can create a time dependent structure

We can add delay to process and have periodic transaction

Compatible with all android versions

WORKING STRUCTURE

Create a Worker Class

Create a request schedule time

Call request

Run Worker class when it's time

How to use?

implementation "androidx.work:work-runtime-ktx:2.6.0"

```
class MyWorker(
    appContext: Context,
    workerParams: WorkerParameters
) :Worker(appContext,workerParams) {

    override fun doWork(): Result {

        val sumProcess = 10 + 20
        Log.e( tag: "sumProcess: ",  msg: "$sumProcess")

        return Result.success()
    }
}
```

```
private lateinit var request: WorkRequest
```

```
request = OneTimeWorkRequestBuilder<MyWorker>()
    .setInitialDelay( duration: 10, TimeUnit.SECONDS)
    .build()
```

```
request =PeriodicWorkRequestBuilder<MyWorker>( repeatInterval: 15, TimeUnit.MINUTES)
    .setInitialDelay( duration: 10, TimeUnit.SECONDS)
    .build()
```

```
WorkManager.getInstance( context: this@MainActivity)
    .enqueue(request)
```

EXTRAS!

get status

```
WorkManager.getInstance( context: this@MainActivity)
    .getWorkInfoByIdLiveData(request.id)
    .observe( owner: this@MainActivity, { workInfo ->
        val status = workInfo.state.name
        Log.e( tag: "Process state", status)
    })
```

add constraint

```
val workCondition = Constraints.Builder()
    .setRequiredNetworkType(NetworkType.CONNECTED)
    .build()

request = OneTimeWorkRequestBuilder<MyWorker>()
    .setInitialDelay( duration: 10, TimeUnit.SECONDS)
    .setConstraints(workCondition)
    .build()
```

create notification

```
val builder: NotificationCompat.Builder

val notificationManager =
    applicationContext.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

val intent = Intent(applicationContext, MainActivity::class.java)

val contentToGo = PendingIntent.getActivity(
    applicationContext,
    requestCode: 1,
    intent,
    PendingIntent.FLAG_UPDATE_CURRENT
)
```

check version

```kotlin
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    val channelId = "channelId"
    val channelName = "channelName"
    val channelIntroduction = "channelIntroduction"
    val channelPriority = NotificationManager.IMPORTANCE_HIGH

    var channel: NotificationChannel? = notificationManager
        .getNotificationChannel(channelId)

    if (channel == null) {
        channel = NotificationChannel(channelId, channelName, channelPriority)
        channel.description = channelIntroduction
        notificationManager.createNotificationChannel(channel)
    }

    builder = NotificationCompat.Builder(applicationContext, channelId)
    builder.setContentTitle("Title")
        .setContentText("Content text")
        .setSmallIcon(R.drawable.icon)
        .setAutoCancel(true)
        .setContentIntent(contentToGo)

} else {
    builder = NotificationCompat.Builder(applicationContext)
    builder.setContentTitle("Title")
        .setContentText("Content text")
        .setSmallIcon(R.drawable.icon)
        .setAutoCancel(true)
        .setContentIntent(contentToGo)
        .priority = Notification.PRIORITY_HIGH
}
notificationManager.notify( id: 1, builder.build())
```


Happy
Coding!
yagmurerdogan