


developer.android.com/courses/pathways/android-development-with-kotlin-12#quiz-/courses/quizzes/android-development-with-kotlin-12/android-development-with-...

developers 平台 Android Studio Google Play Jetpack Kotlin 文档 新闻 搜索 中文 - 简体 九鼎

## 第 12 课: 仓库模式和 WorkManager

了解仓库模式以及如何使用 WorkManager.

2 项活动 • 1 个测验



第 12 课: 存储库模式和 WorkManager

Badge earned!  
[View profile](#)



### 使用 WorkManager 处理后台工作

Codelab



### 推荐的应用架构

文章

### 测验

完成第 12 课: 仓库模式和 WorkManager, 赢取徽章。

[参加测验](#)

```
android-workmanager-start_kotlin app src main java com example background workers BlurWorker.kt
1 // Copyright (C) 2018 The Android Open Source Project ...
16
17 package com.example.background.workers
18
19 import ...
20
21 private const val TAG = "BlurWorker"
22
23 class BlurWorker(ctx: Context, params: WorkerParameters) : Worker(ctx, params) {
24
25     override fun doWork(): Result {
26         val appContext = applicationContext
27
28         val resourceUri = inputData.getString(KEY_IMAGE_URI)
29
30         makeStatusNotification(message="Blurring image", appContext)
31
32         // This is an utility function added to emulate slower work.
33         sleep()
34
35         return try {
36             if (TextUtils.isEmpty(resourceUri)) {
37                 Log.e(TAG, msg="Invalid input uri")
38                 throw IllegalArgumentException("Invalid input uri")
39             }
40
41             val resolver = appContext.contentResolver
42
43             val picture = BitmapFactory.decodeStream(
44                 resolver.openInputStream(Uri.parse(resourceUri)))
45
46             val output = blurBitmap(picture, appContext)
47
48         } catch (e: Exception) {
49             Log.e(TAG, msg="Error blurring image", e)
50             return Result.failure()
51         }
52     }
53 }
```

```
android-workmanager-start_kotlin app src main java com example background workers CleanupWorker.kt
1 // Copyright (C) 2018 The Android Open Source Project ...
17
18 package com.example.background.workers
19
20 import ...
21
22 /**
23  * Cleans up temporary files generated during blurring process
24  */
25 private const val TAG = "CleanupWorker"
26
27 class CleanupWorker(ctx: Context, params: WorkerParameters) : Worker(ctx, params) {
28
29     override fun doWork(): Result {
30         // Makes a notification when the work starts and slows down the work so that
31         // it's easier to see each WorkRequest start, even on emulated devices
32         makeStatusNotification(message="Cleaning up old temporary files", applicationContext)
33         sleep()
34
35         return try {
36             val outputDirectory = File(applicationContext.filesDir, OUTPUT_PATH)
37             if (outputDirectory.exists()) {
38                 val entries = outputDirectory.listFiles()
39                 if (entries != null) {
40                     for (entry in entries) {
41                         val name = entry.name
42                         if (name.isNotEmpty() && name.endsWith(suffix=".png")) {
43                             val deleted = entry.delete()
44                             Log.i(TAG, msg="Deleted $name - $deleted")
45                         }
46                     }
47                 }
48             }
49             return Result.success()
50         } catch (e: Exception) {
51             Log.e(TAG, msg="Error cleaning up files", e)
52             return Result.failure()
53         }
54     }
55 }
```

```
android-workmanager-start_kotlin app src main java com example background workers SaveImageToFileWorker.kt
38 private val title = "Blurred image"
39 private val dateFormatter = SimpleDateFormat(
40     pattern = "yyyy.MM.dd 'at' HH:mm:ss z",
41     Locale.getDefault()
42 )
43
44 override fun doWork(): Result {
45     // Makes a notification when the work starts and slows down the work so that
46     // it's easier to see each WorkRequest start, even on emulated devices
47     makeStatusNotification(message = "Saving image", applicationContext)
48     sleep()
49
50     val resolver = applicationContext.contentResolver
51     return try {
52         val resourceUri = inputData.getString(KEY_IMAGE_URI)
53         val bitmap = BitmapFactory.decodeStream(
54             resolver.openInputStream(Uri.parse(resourceUri)))
55         val imageUrl = MediaStore.Images.Media.insertImage(
56             resolver, bitmap, title, dateFormatter.format(Date()))
57         if (!imageUrl.isNullOrEmpty()) {
58             val output = workDataOf( ...pairs: KEY_IMAGE_URI to imageUrl)
59
60             Result.success(output)
61         } else {
62             Log.e(TAG, msg: "Writing to MediaStore failed")
63             Result.failure()
64         }
65     } catch (exception: Exception) {
66         exception.printStackTrace()
67         Result.failure()
68     }
```

```
android-workmanager-start_kotlin app src main java com example background workers WorkerUtils.kt
59 private const val TAG = "WorkerUtils"
60
61 fun makeStatusNotification(message: String, context: Context) {
62     // Make a channel if necessary
63     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
64         // Create the NotificationChannel, but only on API 26+ because
65         // the NotificationChannel class is new and not in the support library
66         val name = VERBOSE_NOTIFICATION_CHANNEL_NAME
67         val description = VERBOSE_NOTIFICATION_CHANNEL_DESCRIPTION
68         val importance = NotificationManager.IMPORTANCE_HIGH
69         val channel = NotificationChannel(CHANNEL_ID, name, importance)
70         channel.description = description
71
72         // Add the channel
73         val notificationManager =
74             context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager?
75
76         notificationManager?.createNotificationChannel(channel)
77     }
78
79     // Create the notification
80     val builder = NotificationCompat.Builder(context, CHANNEL_ID)
81         .setSmallIcon(R.drawable.ic_launcher_foreground)
82         .setContentTitle(NOTIFICATION_TITLE)
83         .setContentText(message)
84         .setPriority(NotificationCompat.PRIORITY_HIGH)
85         .setVibrate(LongArray( size: 0))
86
87     // Show the notification
88     NotificationManagerCompat.From(context).notify(NOTIFICATION_ID, builder.build())
```

```
android-workmanager-start_kotlin app src main java com example background BlurActivity
34 super.onCreate(savedInstanceState)
35 binding = ActivityBlurBinding.inflate(layoutInflater)
36 setContentView(binding.root)
37
38 binding.goButton.setOnClickListener { viewModel.applyBlur(blurLevel) }
39
40 // Setup view output image file button
41 binding.sendFileButton.setOnClickListener { it: View?
42     viewModel.outputUri?.let { currentUri ->
43         val actionView = Intent(Intent.ACTION_VIEW, currentUri)
44         actionView.resolveActivity(packageManager)?.run { this: ComponentName
45             startActivity(actionView)
46         }
47     }
48 }
49
50 // Hookup the Cancel button
51 binding.cancelButton.setOnClickListener { viewModel.cancelWork() }
52
53 viewModel.outputWorkInfos.observe( owner: this, workInfosObserver() )
54 }
55
56 private fun workInfosObserver(): Observer<List<WorkInfo>> {
57     return Observer { listOfWorkInfo ->
58
59         // Note that these next few lines grab a single WorkInfo if it exists
60         // This code could be in a Transformation in the ViewModel; they are included here
61         // so that the entire process of displaying a WorkInfo is in one location.
62
63         // If there are no matching work info, do nothing
64         if (listOfWorkInfo.isNullOrEmpty()) {
```

```
100         .setConstraints(constraints)
101         .addTag(TAG_OUTPUT)
102         .build()
103         continuation = continuation.then(save)
104
105         // Actually start the work
106         continuation.enqueue()
107     }
108
109     private fun uriOrNull(uriString: String?): Uri? {
110         return if (!uriString.isNullOrEmpty()) {
111             Uri.parse(uriString)
112         } else {
113             null
114         }
115     }
116
117     private fun getImageUri(context: Context): Uri {
118         val resources = context.resources
119
120         return Uri.Builder()
121             .scheme(ContentResolver.SCHEME_ANDROID_RESOURCE)
122             .authority(resources.getResourcePackageName(R.drawable.android_cupcake))
123             .appendPath(resources.getResourceTypeName(R.drawable.android_cupcake))
124             .appendPath(resources.getResourceEntryName(R.drawable.android_cupcake))
125             .build()
126     }
127
128     internal fun setOutputUri(outputImageUri: String?) {
129         outputUri = uriOrNull(outputImageUri)
130     }
```

```
1  // Copyright (C) 2018 The Android Open Source Project .../
16
17  @file:JvmName("Constants")
18
19  package com.example.background
20
21  // Notification Channel constants
22
23  // Name of Notification Channel for verbose notifications of background work
24  @JvmField val VERBOSE_NOTIFICATION_CHANNEL_NAME: CharSequence =
25      "Verbose WorkManager Notifications"
26  const val VERBOSE_NOTIFICATION_CHANNEL_DESCRIPTION =
27      "Shows notifications whenever work starts"
28  @JvmField val NOTIFICATION_TITLE: CharSequence = "WorkRequest Starting"
29  const val CHANNEL_ID = "VERBOSE_NOTIFICATION"
30  const val NOTIFICATION_ID = 1
31
32  // The name of the image manipulation work
33  const val IMAGE_MANIPULATION_WORK_NAME = "image_manipulation_work"
34
35  // Other keys
36  const val OUTPUT_PATH = "blur_filter_outputs"
37  const val KEY_IMAGE_URI = "KEY_IMAGE_URI"
38  const val TAG_OUTPUT = "OUTPUT"
39
40  const val DELAY_TIME_MILLIS: Long = 3000
41
```