دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گزارش تمرین دوم اندروید

WSBI

گردآورنده:

محمد امین کیانی ۴۰۰۳۶۱۳۰۵۲

استاد درس: جناب اقای شیرمحمدی

نیمسال دوم تحصیلی ۰۴-۱۴۰۳

فهرست مطالب

مستندات

۱- Worker:

یک کامپوننت در Jetpack WorkManager اندروید است که برای انجام عملیات‌های طولانی‌مدت یا غیرهمزمان و قابل تکرار به کار می‌رود. پس عملیات‌هایی که نیاز به زمان زیادی دارند و نباید در رشته اصلی انجام شوند، می‌توانند به Worker سپرده شوند تا عملیات‌های پس‌زمینه‌ای به صورت reliable انجام شوند، حتی زمانی که اپلیکیشن در پس‌زمینه یا بسته است. به طور معمول آن‌ها در پس‌زمینه اجرا می‌شود و پس از اتمام کار نتیجه را به رشته اصلی ارسال می‌کند.

به عنوان مثال، برای انجام یک دانلود طولانی یا پردازش داده‌ها، می‌توان از Worker استفاده کرد. این اطمینان می‌دهد که کار به درستی انجام شده و نتیجه به اپلیکیشن ارسال می‌شود.

نمونه کد از مستندات رسمی Kotlin :

```
val uploadWorkRequest: OneTimeWorkRequest =
OneTimeWorkRequestBuilder<UploadWorker>()
    .setInputData(workDataOf("key" to "value"))
    .build()

WorkManager.getInstance(context).enqueue(uploadWorkRequest)
```

نمونه کد از مثال چگونگی تعریف :

```
// build.gradle
dependencies {
    implementation "androidx.work:work-runtime-ktx:2.8.0"
    implementation "androidx.compose.ui:ui:1.4.0"
    implementation "androidx.compose.material:material:1.4.0"
    implementation "androidx.compose.ui:ui-tooling-preview:1.4.0"
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.4"
}


// SyncUserDataWorker.kt
class SyncUserDataWorker(
    context: Context,
    workerParams: WorkerParameters
) : CoroutineWorker(context, workerParams) {

    override suspend fun doWork(): Result {
        return try {
```

```
            // Simulate the data sync task
            delay(3000) // Simulating a network call or data sync operation
            // Returning success after the operation is done
            Result.success()
        } catch (e: Exception) {
            // Handling failure in case of an exception
            Result.failure()
        }
    }
}


// main.kt

class MainActivity : AppCompatActivity() {

    private val workManager = WorkManager.getInstance(this)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            val syncStatus = remember { mutableStateOf("Not started") }

            // Schedule work on button click
            Button(onClick = {
                scheduleSyncUserDataTask()
                syncStatus.value = "Syncing..."
            }) {
                Text(text = "Start Syncing User Data")
            }

            Text(text = syncStatus.value)
        }
    }

    private fun scheduleSyncUserDataTask() {
        val syncWorkRequest = OneTimeWorkRequestBuilder<SyncUserDataWorker>()
            .build()

        // Enqueue the work request
        workManager.enqueue(syncWorkRequest)

        // Optionally, observe the status of the work
        workManager.getWorkInfoByIdLiveData(syncWorkRequest.id).observe(this) {
workInfo ->
```

```
            if (workInfo.state.isFinished) {
                if (workInfo.state == WorkInfo.State.SUCCEEDED) {
                    println("Data sync successful!")
                } else {
                    println("Data sync failed!")
                }
            }
        }
    }
}
```

۲- Service:

یک کامپوننت در اندروید است که برای انجام وظایف طولانی‌مدت یا به‌صورت پس‌زمینه طراحی شده است. برخلاف Activity که برای تعامل با کاربر است، Service برای انجام کارهایی که نیاز به تعامل با کاربر ندارند، طراحی می‌شود (مانند پخش موسیقی در پس‌زمینه یا دانلود داده‌ها). یک Service می‌تواند به صورت مستقل از رابط کاربری عمل کند و حتی زمانی که کاربر از اپلیکیشن خارج شود، همچنان به کار خود ادامه دهد.

تفاوت اصلی Service با Worker این است که Service برای کارهایی که نیاز به تعامل با سیستم دارند و مدت زمان بیشتری طول می‌کشند، مناسب است.

انواع Service :

۱. **Foreground Service :** سرویسی که عملیاتی را انجام می‌دهد که برای کاربر قابل مشاهده است، مانند پخش موسیقی. این نوع سرویس باید یک اعلان (Notification) نمایش دهد تا کاربر از اجرای آن مطلع باشد .

۲. **Background Service :** سرویسی که عملیاتی را در پس‌زمینه انجام می‌دهد بدون اینکه کاربر مستقیماً از آن آگاه باشد. از اندروید ۸.۰ به بعد، محدودیت‌هایی برای اجرای این نوع سرویس‌ها اعمال شده است .

۳. **Bound Service :** سرویسی که به یک مؤلفه دیگر مانند Activity متصل می‌شود و امکان تعامل و ارسال داده بین آن‌ها را فراهم می‌کند. این سرویس تا زمانی که مؤلفه‌های متصل به آن وجود دارند، فعال می‌ماند .

نمونه کد از مستندات Kotlin :

```
class MyService : Service() {
    override fun onBind(intent: Intent): IBinder? {
        return null
    }
```

```
    override fun onStartCommand(intent: Intent, flags: Int, startId: Int): Int {

        return START_STICKY
    }
}
```

نمونه کد از مثال چگونگی تعریف :

```
// AndroidManifest.xml
<service android:name=".BackgroundTaskService" />
<service android:name=".MyIntentService" />


// .class
import android.app.Service
import android.content.Intent
import android.os.IBinder
import android.util.Log

class BackgroundTaskService : Service() {
    override fun onBind(intent: Intent?): IBinder? {
        // Nah, not today. No binding here!
        return null
    }

    override fun onCreate() {
        super.onCreate()
        log("BackgroundTaskService is ready to conquer!")
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        log("BackgroundTaskService is performing a task! Don't disturb, please!")
        performLongTask()
        return START_STICKY // If the service is killed, it will be automatically
restarted
    }

    private fun performLongTask() {
        // Imagine doing something that takes a long time here
        Thread.sleep(5000)
    }

    override fun onDestroy() {
        super.onDestroy()
```

```
        log("BackgroundTaskService says goodbye!")
    }

    fun log(str:String){
        Log.d("TAG", "log: $str")
    }
}


// .class
import android.app.IntentService
import android.content.Intent
import android.util.Log

class IntentTaskService : IntentService("IntentTaskService") {

    override fun onHandleIntent(intent: Intent?) {
        log("IntentTaskService is on a mission to conquer a task!")
        performLongTask()
    }

    private fun performLongTask() {
        // Imagine doing something that takes a long time here
        Thread.sleep(5000)
    }

    override fun onDestroy() {
        super.onDestroy()
        log("IntentTaskService says farewell!")
    }

    fun log(str:String){
        Log.d("TAG", "log: $str")
    }
}


// main.kt
val serviceIntent = Intent(this, MyService::class.java)
startService(serviceIntent)

val intentServiceIntent = Intent(this, MyIntentService::class.java)
startService(intentServiceIntent)
```

```
//----------------------------------------- fgs:

import android.app.Notification
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.Service
import android.content.Intent
import android.os.IBinder
import androidx.core.app.NotificationCompat
class MyForegroundService : Service() {
    companion object {
        const val CHANNEL_ID = "ForegroundServiceChannel"
    }
    override fun onCreate() {
        super.onCreate()
        createNotificationChannel()
    }
    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        val notification: Notification = NotificationCompat.Builder(this,
CHANNEL_ID)
            .setContentTitle("Foreground Service")
            .setContentText("Service is running")
            .setSmallIcon(R.drawable.ic_service)
            .build()
        startForeground(1, notification)
        // Perform your service task here
        return START_NOT_STICKY
    }
    override fun onBind(intent: Intent?): IBinder? {
        return null
    }
    override fun onDestroy() {
        super.onDestroy()
    }
    private fun createNotificationChannel() {
        val serviceChannel = NotificationChannel(
            CHANNEL_ID,
            "Foreground Service Channel",
            NotificationManager.IMPORTANCE_DEFAULT
        )
        val manager = getSystemService(NotificationManager::class.java)
        manager.createNotificationChannel(serviceChannel)
    }
}
```

```
<service
    android:name=".MyForegroundService"
    android:enabled="true"
    android:exported="true" />

import android.content.Context
import android.content.Intent
import androidx.lifecycle.ViewModel

class MyViewModel(private val context: Context) : ViewModel() {

    fun startForegroundService() {
        val serviceIntent = Intent(context, MyForegroundService::class.java)
        context.startForegroundService(serviceIntent)
    }

    fun stopForegroundService() {
        val serviceIntent = Intent(context, MyForegroundService::class.java)
        context.stopService(serviceIntent)
    }
}

import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.lifecycle.viewmodel.compose.viewModel

@Composable
fun MyServiceScreen(viewModel: MyViewModel = viewModel()) {
    Column {
        Button(onClick = { viewModel.startForegroundService() }) {
            Text("Start Service")
        }
        Button(onClick = { viewModel.stopForegroundService() }) {
            Text("Stop Service")
        }
    }
}
```

۳- Broadcast:

یک مکانیسم است که به اپلیکیشن‌ها اجازه می‌دهد پیغام‌هایی را از یک اپلیکیشن به اپلیکیشن‌های دیگر ارسال کنند و به دو نوع تقسیم می‌شوند: معمولی (Normal) و مرتب‌سازی شده(Ordered)

ازBroadcast ها می‌توان برای ارسال اطلاعیه‌ها، وضعیت‌های خاص سیستم، و سایر رویدادهایی که به همه اپلیکیشن‌ها اطلاع داده می‌شود، استفاده کرد. می‌توان با استفاده از کلاس Intent یک Broadcast ارسال و دریافت کرد. این Broadcast می‌تواند **سیستمی** یا **سفارشی** باشد.

نمونه کد از مستندات Kotlin :

```
val intent = Intent("com.example.broadcast.MY_NOTIFICATION")
sendBroadcast(intent)
```

نمونه کد از مثال چگونگی تعریف :

```
// Sending a Broadcast:

val intent = Intent("com.example.MY_CUSTOM_ACTION")
intent.putExtra("message", "Hello, this is a broadcast!")
sendBroadcast(intent)


// Receiving a Broadcast:

class MyBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val message = intent.getStringExtra("message")
        Log.d("MyBroadcastReceiver", "Received: $message")
    }
}



// Manifest vs. Dynamic Registration:

<receiver android:name=".MyBroadcastReceiver" android:exported="true">
    <intent-filter>
        <action android:name="com.example.MY_CUSTOM_ACTION" />
    </intent-filter>
</receiver>


val filter = IntentFilter("com.example.MY_CUSTOM_ACTION")
registerReceiver(MyBroadcastReceiver(), filter)
```

٤- Intent:

در اندروید یک شیء است که برای انجام درخواست‌های مختلف بین کامپوننت‌های مختلف اپلیکیشن استفاده می‌شود. پس ابزاری است که برای انجام درخواست‌ها و تعاملات بین کامپوننت‌ها مانندActivity ها، Serviceها وBroadcast Receiver ها است. Intentها می‌توانند داده‌هایی را از یک کامپوننت به دیگری ارسال کنند یا حتی به فعالیت‌های مختلف جهت اجرا هدایت کنند.همچنین می‌توانند داده‌هایی را برای انتقال بین کامپوننت‌ها به همراه داشته باشند.

نکته: با اینکه استفاده از Intent برای شروع Activity ها در Jetpack Compose ممکن است، اما بهتر است از کتابخانه **Navigation Component** برای مدیریت ناوبری بین کامپوننت‌ها استفاده کنیم، زیرا این روش امکانات پیشرفته‌تری برای مدیریت مسیرها و ارسال داده بین کامپوننت‌ها فراهم می‌کند.

نمونه کد از مستندات Kotlin :

```
val intent = Intent(this, SecondActivity::class.java)
intent.putExtra("key", "value")
startActivity(intent)
```

نمونه کد از مثال چگونگی تعریف :

```
import android.content.Intent
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.unit.dp

@Composable
fun MainScreen() {
    val context = LocalContext.current

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center
    ) {
        Button(onClick = {
            val intent = Intent(context, SecondActivity::class.java)
            context.startActivity(intent)
        }) {
            Text(text = " open 2nd activity")
```

```
        }
    }
}
```

٥- مراجع:

[1] developer.android.com/reference/androidx/work/WorkManager
[2] Worker | API reference | Android Developers
[3] https://medium.com/@YodgorbekKomilo/working-with-workmanager-in-android-implementing-with-jetpack-compose-and-kotlin-coroutines-flow-7e8de4a20a30
[4] developer.android.com/reference/android/app/Service
[5] https://medium.com/@fierydinesh/understanding-service-and-intentservice-in-android-with-kotlin-cea76512ec16
[6] https://medium.com/@YodgorbekKomilo/understanding-android-services-types-lifecycle-and-jetpack-compose-example-5d64b996bd83
[7] https://developer.android.com/develop/background-work/services/fgs/declare
[8] https://developer.android.com/develop/background-work/services/bound-services
[9] developer.android.com/reference/android/content/BroadcastReceiver
[10] https://medium.com/@zekromvishwa56789/understanding-android-broadcasts-a-practical-guide-for-developers-c05c1f431892
[11] developer.android.com/reference/android/content/Intent
[12] https://blog.devgenius.io/android-development-adding-new-activity-explicit-intent-and-top-app-bar-with-jetpack-compose-71d01f01bc58
[13] https://rommansabbir.com/android-intent-a-comprehensive-guide-with-examples
[14] https://medium.com/@EazSoftware/exploring-activity-in-android-jetpack-compose-00b4ef88b439