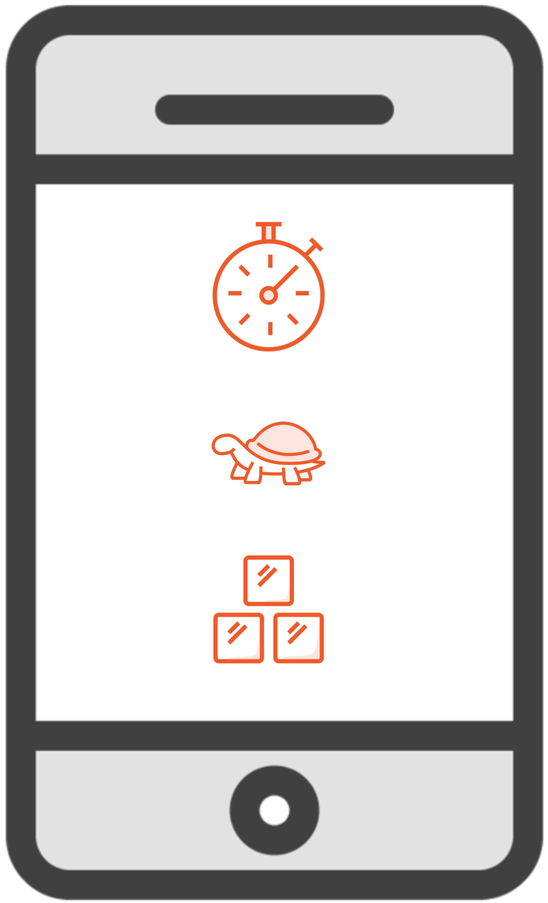
WorkManager

- Part of the Android Architecture Components in Jetpack
- Provides a consistent API for scheduling background tasks across multiple OS versions
- Worker classes describe the background task, or ‘work’
- Flexible scheduling
- Work constraints
- Extensions for Kotlin

Why Run Tasks in the Background?



WorkManager Gets the Work Done

Earlier Background Task APIs

AlarmManager

AsyncTask

JobScheduler

WorkManager

Consistent API

Backward compatible across multiple OS versions

API Level 23 and up will use JobScheduler

Earlier versions may use AlarmManager

Write only one codebase!

Using WorkManager

Describe the work to be done

Create a work request

WorkManager will schedule the work request

Implementing WorkManager

MyWorker.kt

```
class MyWorker: Worker {
  override fun doWork(): Result {
    // do the work!
    return Result.success()
  }
}
```

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
  val workManager = WorkManager
    .getInstance(this)

  fun startWork() {
    val workRequest = OneTimeWorkRequest
      .Builder(...)
      .build()
    workManager.enqueue(workRequest)
  }
}
```

Inputs and Outputs

MyWorker.kt

```
class MyWorker: Worker {
  override fun doWork(): Result {
    // do the work!
    return Result.success()
  }
}
```

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
  val workManager = WorkManager
    .getInstance(this)

  fun startWork() {
    val data = Data.Builder(...).build()
    val workRequest = OneTimeWorkRequest
      .Builder(...)
      .setInputData(data)
      .build()
    workManager.enqueue(workRequest)
  }
}
```

Inputs and Outputs

MyWorker.kt

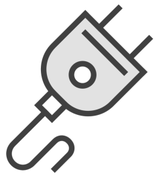
```
class MyWorker: Worker {
  override fun doWork(): Result {
    // do the work!
    val data = inputData.getString(“KEY”)
    return Result.success()
  }
}
```

MainActivity.kt

```
class MainActivity : AppCompatActivity() {
  val workManager = WorkManager
    .getInstance(this)

  fun startWork() {
    val data = Data.Builder(...).build()
    val workRequest = OneTimeWorkRequest
      .Builder(...)
      .setInputData(data)
      .build()
    workManager.enqueue(workRequest)
  }
}
```

Work Constraints



Power Source

Remaining Battery

Available Storage

Network Type

Idle Device

Work Constraints

```
val constraints = Constraints.Builder()
  .setRequiresCharging(true)
  .setRequiredNetworkType(NetworkType.CONNECTED) // NetworkType.NOT_ROAMING
  .build()

val workRequest = OneTimeWorkRequestBuilder<MyWorker>()
  .setConstraints(constraints)
  .build()
```

Work Constraints

```
val constraints = Constraints.Builder()
   .setRequiresCharging(true)
   .setRequiredNetworkType(NetworkType.CONNECTED) // NetworkType.NOT_ROAMING
   .build()

val workRequest = OneTimeWorkRequestBuilder<MyWorker>()
   .setConstraints(constraints)
   .build()
```

Work Constraints

```
val constraints = Constraints.Builder()
   .setRequiresCharging(true)
   .setRequiredNetworkType(NetworkType.CONNECTED) // NetworkType.NOT_ROAMING
   .build()

val workRequest = OneTimeWorkRequestBuilder<MyWorker>()
   .setConstraints(constraints)
   .build()
```

Work Constraints

```
val constraints = Constraints.Builder()
  .setRequiresCharging(true)
  .setRequiredNetworkType(NetworkType.CONNECTED) // NetworkType.NOT_ROAMING
  .build()

val workRequest = OneTimeWorkRequestBuilder<MyWorker>()
  .setConstraints(constraints)
  .build()
```

Work Constraints

```
val constraints = Constraints.Builder()
   .setRequiresCharging(true)
   .setRequiredNetworkType(NetworkType.CONNECTED) // NetworkType.NOT_ROAMING
   .build()

val workRequest = OneTimeWorkRequestBuilder<MyWorker>()
   .setConstraints(constraints)
   .build()
```

Summary

WorkManager is an Android service to schedule background tasks

- Provides a consistent API that is backward compatible

Describe work in Worker classes

Work requests describe conditions for work to be done

Work can be scheduled once, or repeated

Work can accept inputs and return outputs

Work can depend on the state of the device

Kotlin extensions