Androidskolen Sesjon 5

Soon™

bouvet

Agenda

- Android-teori
- Grunnleggende trådhåndtering
- Praktisk oppgave # 1
- AsyncTask
- Praktisk oppgave # 2
- ExecutorService
- Praktisk oppgave # 3
- (Job)IntentService
- Praktisk oppgave # 4
- Annet
- Oppsummering

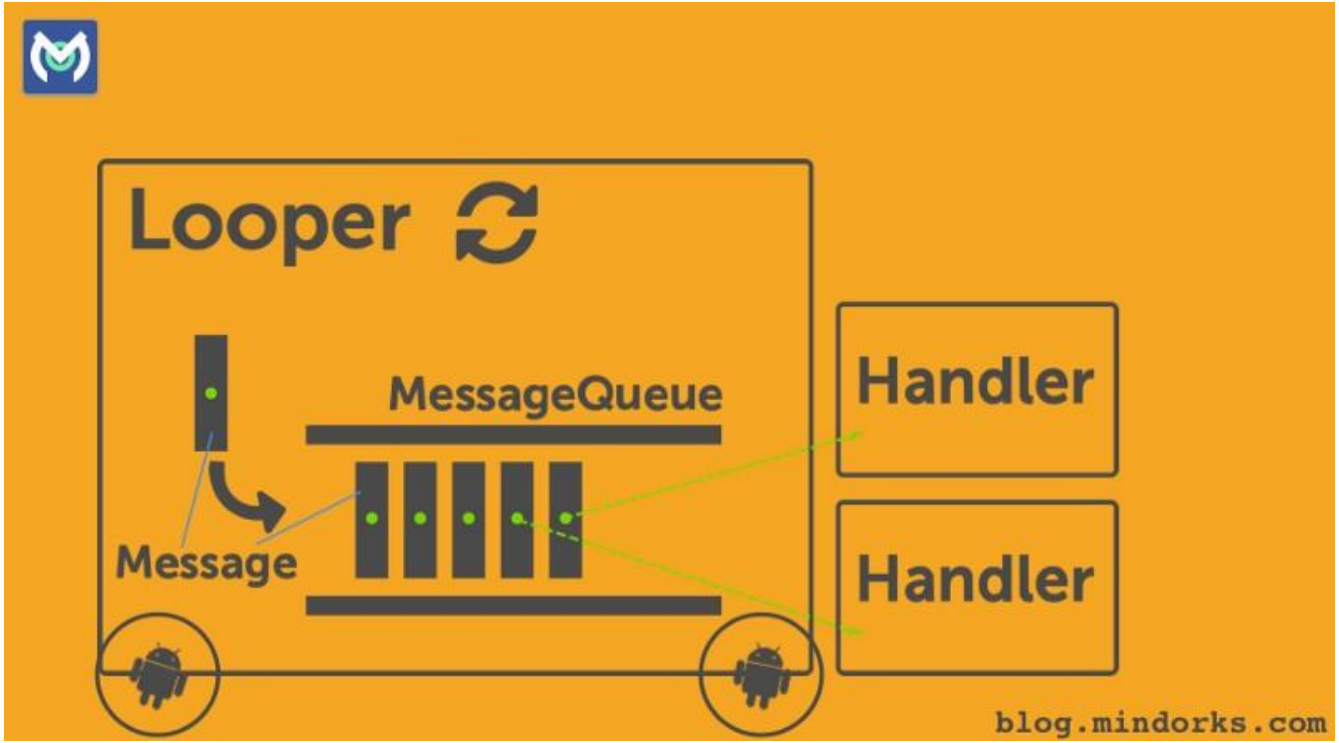
bouvet

Dagens målsetning

Siden Android-plattformen baserer seg på en En-tråds GUI modell, tvinges utviklere på plattformen til å forholde seg til tråd-håndtering uavhengig av kompleksiteten på applikasjonen. Målet blir dermed generell kjennskap til de ulike mekanismene for å avlaste GUI-tråden på tyngre operasjoner, og et vurderingsgrunnlag for hvilken fremgangsmåte som passer i ulike situasjoner.

bouvet

Android-teori



bouvet

Grunnleggende trådhåndtering

```
thread {
      // Do the thing!
}
```

- Vanlige tråder oppfører seg som forventet.
- GUI-tråden (“main”) er er spesiell, og nåes f.eks. via:

```
activity.runOnUiThread {
      // Do the thing, to views!
}
```

bouvet

Handler og Looper

- Handler - send meldinger til en dedikert tråd
- Looper - spinner tråden og håndterer meldingskøen

```
val handlerThread = HandlerThread(“Trådnavn for debugging”)
handlerThread.start() // Forebereder looperen, ellers fungerer ingenting
Handler(handlerThread.looper).post { work }
```

Praktisk oppgave #1

- Bruk Handlere istedenfor vanlige Threads til å gjøre AsyncWork
- Se HandlerAsyncWorker

bouvet

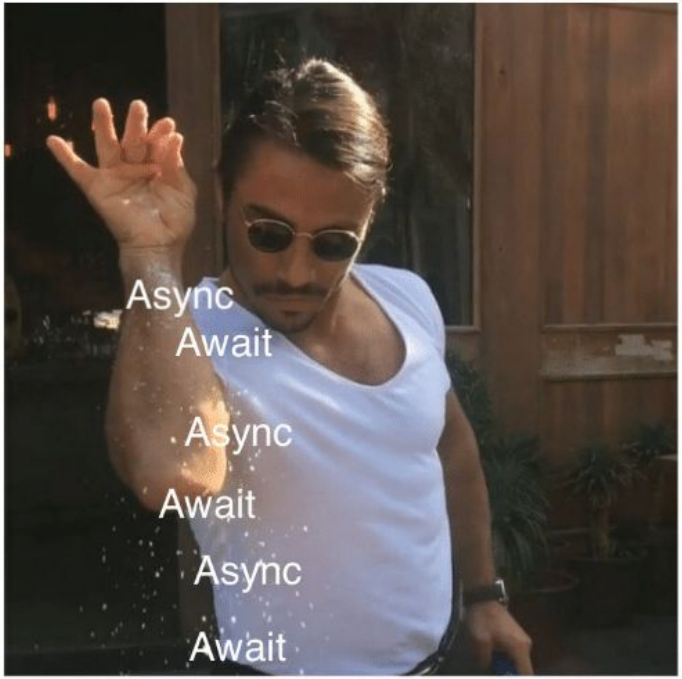
AsyncTask

- Innebygd fremgangsmåte for
  “gjør ting i bakgrunnen, oppdater GUI”

```
object : AsyncTask<Params, Progress, Result>() {
      override fun doInBackground(vararg params: Params?): Result {

      }
       override fun onPostExecute(result: Result) {

       }
}
```



Praktisk oppgave #2

- Samme som i oppgave #1, men med AsyncTask
- Se TaskAsyncWorker

bouvet

ExecutorService

- Innebygd Thread-pooling støtte. Håndterer arbeidskø internt, konfigureres med antall tåder som maksimalt skal brukes

```
val executor = Executors.newFixedThreadPool( thread count )
executor.submit { work }
```

bouvet

Praktisk oppgave #3

- Samme som i oppgave #2, men med en av ExecutorService-ene
- Se ExecutorAsyncWorker

bouvet

(Job)IntentService

- Send “arbeid” til en bakgrunnsservice.
- Kan også kjøres i egen prosess

```
enqueueWork(context, WorkService::class.java, 1000, work)

override fun onHandleWork(intent: Intent) {

}
```

bouvet

Praktisk oppgave #4

- Samme som i oppgave #3, men med JobIntentService
- Se ServiceAsyncWorker og WorkService

bouvet

BroadcastReceiver

- Tilsvarende mekanismer som JobIntentService
- Kjører basert på events fra andre apper eller systemet

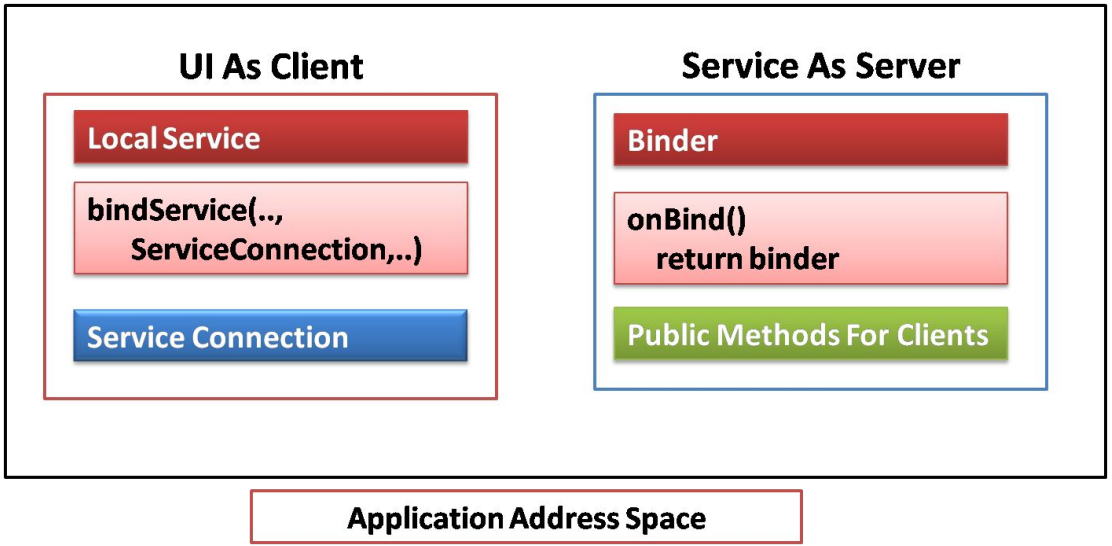
```
override fun onReceive(context: Context, intent: Intent) {

}
```

bouvet

Bound Services

- Oppfører seg mer som en “MicroService” i web-verdenen
- Kjører så lenge noen er tilkoblet



bouvet

Hva med Coroutines?

- Mer en Kotlin-detalj enn en Android-løsning
- Større endringer så sent som i Kotlin 1.3 (release i oktober)
- Eksamensrelevans?

bouvet

Oppsummering

- Hold arbeid unna “main”-tråden så langt det lar seg gjøre
- Fremgangsmåten er mindre viktig

bouvet