NTNU
Kunnskap for en bedre verden

Internasjonalisering og Android Services

Tomas Holt, Institutt for datateknologi og informatikk ved NTNU
20.09.2018
Lærestoffet er utviklet for faget IFUD1042 Applikasjonsutvikling for Android

Resymé: Denne leksjonen tar for seg tjenester (Android Service) som er et alternativ til aktiviteter i en del tilfeller. Hovedfokus ligger imidlertid på internasjonalisering.

Contents

9 Internasjonalisering og Android Services

9.1 Om leksjonen

Leksjonen omhandler hvordan man lager applikasjoner tilpasset internasjonale brukere. Dette er ikke tatt opp i boka, men ansees av oss faglærere som så viktig at vi likevel tar det med. I tillegg blir pensum kapittel 24 i boka om tjenester i Android (Android Services). Dette er en naturlig fortsettelse på det vi allerede har lært om tråder og nettverksprogrammering. F.eks. ville det vært mer naturlig å lage tjeneren i socketøvingen som en tjeneste i stedet for en normal applikasjon (da denne ikke har behov for noen kjørende GUI). Når det er sagt så vil tjenerene i de fleste tilfeller kjøre på egne PC'er.

9.2 Tjenester

I Android har man klassen Service og flere sub-klasser av denne. Dette er en klasser som gjerne brukes for å kjøre jobber i bakgrunnen, uten noen form for UI. La oss starte med å se på hva som står på [Android Service]. Følgende tekst er også hentet derfra:

”A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service might handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.
A service can essentially take two forms:
Started
A service is "started" when an application component (such as an activity) starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller. For example, it might download or upload a file over the network. When the operation is done, the service should stop itself.
Bound
A service is "bound" when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.
...your service can work both ways—it can be started (to run indefinitely) and also allow binding. It’s simply a matter of whether you implement a couple callback methods: onStartCommand() to allow components to start it and onBind() to allow binding.
Regardless of whether your application is started, bound, or both, any application component can use the service (even from a separate application), in the same way that any component can use an activity—by starting it with an Intent. However, you can declare the service as private, in the manifest file, and block access from other applications. ”

Du bør også merke deg følgende hentet fra samme kilde som over:

*"**Should you use a service or a thread?***
A service is simply a component that can run in the background even when the user is not interacting with your application. Thus, you should create a service only if that is what you need.
If you need to perform work outside your main thread, but only while the user is interacting with your application, then you should probably instead create a new thread and not a service. For example, if you want to play some music, but only while your activity is running, you might create a thread in onCreate(), start running it in onStart(), then stop it in onStop(). Also consider using AsyncTask or HandlerThread, instead of the traditional Thread class. See the Processes and Threading document for more information about threads.
Remember that if you do use a service, it still runs in your application's main thread by default, so you should still create a new thread within the service if it performs intensive or blocking operations."

Som du kan se så er det ikke klart definert hvor man bør bruke tjenester kontra vanlige aktiviteter. Her må man faktisk gjøre et valg selv. I det hele tatt finner du mye informasjon både i boka og på [Android Service] så temaet omtales ikke mer her.

9.3 Internasjonalisering

En god del av applikasjonene som lages for Android-plattformen lages for en internasjonal brukermasse. Det er derfor lagt opp til at det skal være enkelt å tilpasse språk og annen informasjon til brukeren. Dette omtales ikke i boka, men er noe man absolutt bør vite om når man lager løsninger for Android-plattformen. Vi skal her se spesielt på hvordan man håndterer strenger, tall, datoer og beløp.

9.3.1 Strenger

Når det gjelder tekst kan den enkelt tilpasses flere språk/regioner ved å lage flere varianter av strings.xml. Hver versjon lagres i en egen katalog under /res/values-xx-rYY. xx skiftes ut med språkkode (no, en, fr, de osv.), mens YY skiftes ut med region/land. Når det gjelder region så kan dette f.eks. være US eller AU for engelsk, avhengig av om man befinner seg i USA eller Australia.

Merk

- at for norsk så er bokmål og nynorsk definert som ulike språk (xx). Bokmål er nb, nynorsk nn, og norsk er no.

- at man ikke er nødt til å angi region. Det er mulig kun å forholde seg til språk og man bruker da kun /res/values-xx/strings.xml, hvor xx endres til språkkode (f.eks. no for norsk).
- at man bestandig bør sørge for å ha en katalog som bare heter /res/values/. Denne vil da brukes om applikasjonen ikke støtter ønsket språk/region.

La oss ta et lite eksempel der vi bruker engelsk og norsk (vi lager altså bare to strings.xml filer).

Vi kikker først på /res/values-nb-rNO/strings.xml som altså blir den norske (bokmål) utgaven.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="navn">Vennligst oppgi ditt navn</string>
</resorces>
```

I koden kan vi nå bruke strengen navn slik:

```
String n  = getString(R.string.navn);
```

Resultatet er at variabelen n blir ”Vennligst oppgi ditt navn” så lenge innstillingene på enheten (se neste kapittel) er bokmål (nb) og region er Norge (NO). I alle andre tilfeller vil strengen angitt i fila /res/values/strings.xml brukes. Om innholdet i denne fila er

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="navn">Please submit your name</string>
</resorces>
```

så vil n være lik ”Please submit your name”.

Denne oppbygningen gjør det meget enkelt å tilpasse tekst til ulike språk og regioner.

Språkkodene for xx finner du spesifisert i [ISO 639-1], mens kodene for region (YY) er beskrevet i [ISO 3166].

9.3.2 Locale

Locale (engelsk) beskriver gjerne både språk og region. Man kan finne ut hvilken locale systemet bruker ved å gjøre følgende:

```
Locale locale = getResources().getConfiguration().locale;
```

For fullstendig oversikt over klassen Locale se API-dokumentasjonen. Jeg kan her nevne metoden getDisplayCountry() som viser hvilket land som representeres og getDisplay-Language() som inneholder valgt språk.

Merk at det er mulig å skifte locale i systemet. Bruker man emulator kan dette gjøres via adb eller direkte i emulatoren (som forøvrig blir det samme for ordentlige enheter). Slik gjøres det i eumlatoren:

1. Trykk på knappen for å vise installerte applikasjoner.
2. Velg Custom Locale.
3. Bla fram til den innstillingen du vil ha (om den ikke finnes se under) og gjør et langt trykk på ønsket valg.
4. Velg så Apply.

Om ønsket locale ikke eksisterer trykk på Add New Locale. Du kan da legge til ditt eget ønske. Formen blir xx_YY hvor xx og YY skiftes ut som beskrevet i forrige delkapittel.

9.3.3 Tall

Når det gjelder tall så må vi også her gjøre tilpassinger til språk/region.

F.eks. om vi skal skrive ut et tall kan dette gjøres slik:

```
int tall = 20000;
String strTall = NumberFormat.getInstance().format(tall);
```

Innholdet i strTall vil nå avhenge av hvilken locale man bruker. I engelsktalende land er det vanlig å bruke komma for å angi tusen og strTall vil da være 20,000. I eksemplet vil NumberFormat.getInstance() hente et NumberFormat-objekt for gjeldene locale. Man kan imidlertid også selv bestemme hvilken locale man vil bruke slik:

```
NumberFormat nf = NumberFormat.getInstance(Locale.FRENCH);
```

Her vil nf være satt opp for fransk språk, og kan brukes på samme måte som beskrevet over.

9.3.4 Datoer

Datoer håndteres veldig likt tall. Her er et eksempel:

```
Date dato = new Date();//dagens dato brukes
String strDato = DateFormat.getDateInstance().format(dato);
```

Nå vil strDato inneholde en streng som passer til datoutskrift for gjeldende locale. Man kan også angi locale eksplisitt:

```
DateFormat df = DateFormat.getDateInstance(DateFormat.LONG, Locale.FRANCE);
```

Her tilpasses df til Frankrike og DateFormat.LONG. Her er en beskrivelse av DateFormat.LONG og de andre alternativene (hentet fra API-dokumentasjonen).

- SHORT is completely numeric, such as 12.13.52 or 3:30pm
- MEDIUM is longer, such as Jan 12, 1952
- LONG is longer, such as January 12, 1952 or 3:30:32pm
- FULL is pretty completely specified, such as Tuesday, April 12, 1952 AD or 3:30:42pm PST.

Jeg legger også merke til at utskrift av dato ikke nødvendigvis er riktig i forhold til språk. For norsk blir det i hvert fall ikke som det skal. Det kan derfor være en fordel å vurdere f.eks. engelsk foreløpig.

9.3.5 Beløp/valuta

For å håndtere beløp kan du bruke NumberFormat sammen med Currency-klassen. NumberFormat vil da brukes for å formatere tallet, mens Currency vil gi riktig bokstavkombinasjon (f.eks. nkr om vi snakker om norske kroner). Klassen Currency er temmelig enkel. Her er noen av metodene (se forøvrig API-dokumentasjonen):

```
public static Currency getInstance(Locale locale);
public String getSymbol();
public String getSymbol(Locale locale);
```

Her er beskrivelsen for den siste metoden:

Returns the localized currency symbol for this currency in locale. That is, given "USD" and Locale.US, you'd get "$", but given "USD" and a non-US locale, you'd get "US$". If the locale only specifies a language rather than a language and a country (such as Locale.JAPANESE or {new Locale("en", "")} rather than Locale.JAPAN or {new Locale("en", "US")}), the ISO 4217 currency code is returned.
If there is no locale-specific currency symbol, the ISO 4217 currency code is returned.

For mer informasjon om ISO 4217 se [ISO 4217].

Av informasjonen over kan man skjønne at en typisk bruk vil være slik:

```
Locale locale = //hent locale
String strCurrency = Currency.getInstance(locale).getSymbol();
```

strCurrency vil nå inneholde symbol/bokstaver for aktuell region. **Vær klar over at om systeminnstillingene kun er for språk (xx), og region (YY) ikke er satt, vil man få IllegalArgumentException ved bruk av koden over.**

9.3.6 Bilder

Det er også i mange sammenhenger ønskelig å kunne endre bilde(r) etter hvilket land brukeren er fra. Det kan f.eks. være aktuelt å vise et flagg som angir hvilket land det er snakk om. Måten dette håndteres på er lik det man gjør for strenger. Man lager en katalog som beskriver hvilken språk/region det er snakk om, så velges dette automatisk avhengig av systeminnstillingene. Standardkatalogen for bilder er som kjent /res/drawable/ (og denne vil velges om det ikke er noen andre kataloger som samsvarerer med locale i systemet).

Formen på navngivingen blir /res/drawable-xx/ hvor xx skiftes ut med språkkode (f.eks. no for norsk). Om man vil bruke region så blir navngivingen /res/drawable-xx-rYY/ hvor YY fortsatt er koden for regionen det er snakk om.

9.3.7 Endring av locale i kode

I enkelte tilfeller kan det være greit å kunne endre språk/region i kode. I en del tilfeller vil det være greit å spesifisere locale på metodenivå. F.eks. kan vi gjøre slik (litt endring av koden i 1.3.5):

```
String strCurrency =
Currency.getInstance(new Locale(”no”,”NO”)).getSymbol();
```

Her spesifiserer vi eksplisitt at vi vil ha norske kroner (vi angir norsk språk og norsk region). Etter at dette er gjort kan strCurrency vises for brukeren med norsk drakt.

I andre tilfeller vil det imidlertid være aktuelt å endre defaultverdien for locale, slik at den gjelder alt vi gjør i applikasjonen. Dette kan f.eks. påvirke hvilke bilder som blir lastet osv. Bilder lastes som kjent med kode ala

```
ImageView image = (ImageView)findViewById(R.id.image);
```

Her vil bilde lastes fra /res/drawable-xx, hvor xx er bestemt av språkkode. Ved å endre defaultverdien for locale før vi bruker findViewById() kan vi også endre hvilken katalog bildet lastes fra. La oss si at vi har en applikasjon hvor vi skal vise et norsk flagg for alle norske brukere. Problemet i norge er at vi har flere språk. Vi har bokmål (nb), nynorsk (nn) og norsk (no). Må vi da lage /res/drawable-nb, /res/drawable-nn og /res/drawable-no for å dekke alle disse mulighetene. Nei, det må vi ikke (men det kan godt hende det er lurt...). Uansett, vi kan bruke kode ala dette:

```
Locale locale = getResources().getConfiguration().locale;
if (locale.getLanguage().equals("no") ||
locale.getLanguage().equals("nb") ||
locale.getLanguage().equals("nn")){

locale = new Locale("no","NO");
//let’s update system configuration
```

```
Configuration config = new Configuration();
config.locale = locale;
Resources res = getBaseContext().getResources();
res.updateConfiguration(config, res.getDisplayMetrics());
}
```

Vi sjekker om språk er no, nb eller nn og i alle disse tilfellene oppretter vi en locale med norsk språk og region, som vi så bruker for å oppdatere konfigurasjonen av enheten vår. **NB!** For vi skal kunne endre default locale på denne måten så må vi sørge for å spesifisere dette i AndroidManifest.xml. Det kan der se slik ut:

```
<activity android:name=".MainActivity"
android:label="@string/app_name"
android:configChanges="locale">
```

Se forøvrig eksempelapplikasjon hvor du finner mye av det som er beskrevet over.

9.3.8 Andre muligheter

Man har også andre muligheter for håndtering av ressurser avhengig av språk, region og faktisk også nettverk. Om man f.eks. ønsker å ha bilder i ulike oppløsninger for ulike språk så er dette fullt mulig.

Du finner en generell beskrivelse av internasjonalisering på [Android Localization], mens du finner en detaljert beskrivelse av navngiving og hvordan Android prioriterer hvilke ressurser som skal velges på [Android Resources]. Det finnes også en liten tutorial på [Android Localization Tutorial].

9.3.9 Eksempelapplikasjon

I forbindelse med denne leksjonen vil det ikke være noen øving. Under finner du imidlertid en frivillig oppgave som er utgangspunkt for eksempelapplikasjonen til leksjonen. Du kan derfor godt prøve å løse oppgaven før du tyr til eksempelet. Da vil du også få en pekepinne på om stoffet sitter :-)

Oppgave:

Lage en applikasjon som tar hensyn til språk/region-innstillinger på enheten programmet kjører.

Lag en applikasjon som skriver ut

- en velkomstmelding tilsvarende: «Velkommen til norsk versjon av locale app», tilpasset språket det er snakk om.
- hvilken locale (språk og region) som er satt i systemet (forkortelser holder her f.eks. no og NO).

- hvilken dato det er
- hvilken valuta som brukes (forkortelse holder – som f.eks. kr eller Nkr)
- I tillegg så skal du også vise et flagg for hvilket land innstillingene gjelder for.

Flagg finner vi på http://www.flags.net/. De språk/regioner du skal støtte er norsk (bokmål og nynorsk) og engelsk. For sistnevnte holder det med Storbritannia og USA. Om locale er noe annet enn disse beskrevet foran skal engelsk (storbritannias-) flagg vises - eller om du klarer det - ikke noe flagg i det hele tatt. Tekst skal da være på engelsk, men valuta osv. skal skrives ut for gjeldende locale...

9.4 Referanser

[Android Localization] - http://developer.android.com/guide/topics/resources/localization.html

[Android Resources] -

http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources

[Android Service] - http://developer.android.com/guide/components/services.html

[Android Localization Tutorial] -

http://developer.android.com/resources/tutorials/localization/index.html

[ISO 3166] -

http://www.iso.org/iso/country_codes.htm

[ISO 4217] - http://no.wikipedia.org/wiki/ISO_4217

[ISO 639-1] - http://www.loc.gov/standards/iso639-2/php/code_list.php