Intenti
Broadcast Receiver-i

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Razvoj mobilnih aplikacija i servisa
Računarstvo i informatika

2023.

Intent

- Intent je objekat kojim se može zahtevati akcija neke druge komponente aplikacije, ili neke sasvim druge aplikacije
- Komunikacija između delova aplikacije postiže se pomoću Intent-a
- Eng: *intent* = namera

Intent

- Intent-i se koriste za:
  - Startovanje aktivnosti
  - Startovanje servisa
  - Vršenje broadcast-a nekih informacija
  - Startovanje drugih aplikacija
- Mogu se koristiti i prilikom zaustavljanja servisa

Intent klasa

Intent : *Parcelable, Cloneable*	
action	Akcija koju nosi intent
data	Podaci koje nosi intent, izraženi preko URI-ja
category	Kategorija akcije koju pokreće intent
type	Eksplicitni tip podataka koje nosi intent
component	Komponenta koja handle-uje intent
extras	Bundle koji predstavlja dodatne informacije

Primeri

- Action: ACTION_MAIN, ACTION_VIEW, ACTION_EDIT, ACTION_IMAGE_CAPTURE, ...
- Data: content://contacts/people/1, tel:123, ...
- Category: CATEGORY_HOME, CATEGORY_OPENABLE, ...
- Puna lista na: https://developer.android.com/reference/kotlin/android/content/Intent#constants

Vrste intenta

1. Eksplicitni:
   - Definisana je tačna komponenta koja treba da obradi Intent
   - Postavljanje komponente se vrši pomoću setComponent() funkcije, a postavljanje klase pomoću setClass()
   - Potrebno je proslediti ili ime paketa aplikacije koja rešava Intent, ili ime klase komponente iz aplikacije koja šalje Intent
   - Osim imena komponente, ovakav Intent najčešće ne sadrži druge informacije

Vrste intenta

1. Implicitni:
   - Ne definiše tačnu komponentu koja obrađuje Intent
   - Služe za deklarisanje akcije koju treba izvršiti
   - Android sistem treba sam da ispita tip Intent-a na osnovu njegovih atributa i da odredi koja aplikacija treba da ga obradi

Vrste intenta

- Primer eksplicitnog Intent-a:

Ovo nije akcija, već klasa!

```
val downloadIntent = Intent(this, DownloadService::class.java).apply {
    data = Uri.parse(fileUrl)
}
startService(downloadIntent)
```

```
public Intent (Context packageContext, Class<?> cls)
```

Vrste intenta

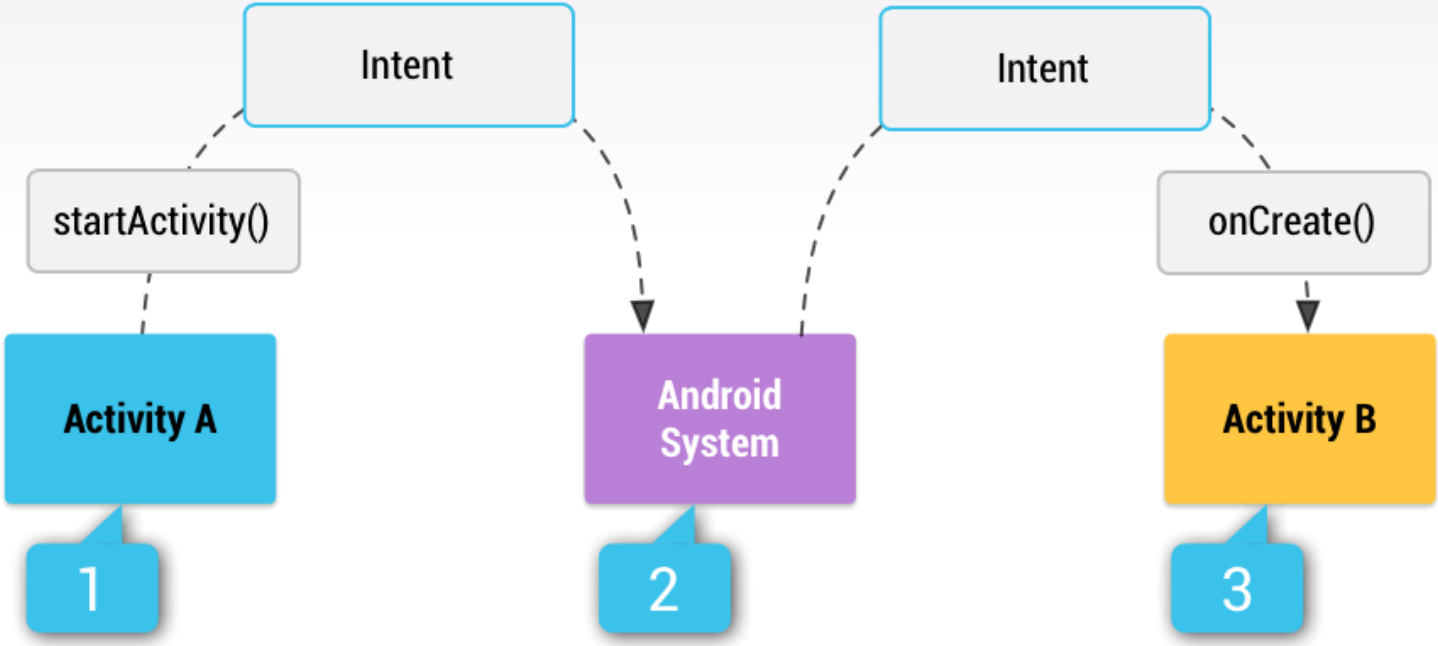
- Primer implicitnog Intent-a:

Ovo jeste akcija!

```
val cameraIntent: Intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
resultLauncher.launch(cameraIntent)
```

```
public Intent (String action)
```

Obrada implicitnog intenta



prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Obrada implicitnog intenta

- Aplikacija kreira Intent i pokreće activity
- Android sistem preuzima kreirani Intent i ispituje njegove atribute
- Sistem pretražuje sve aplikacije koje imaju definisani *intent filter* koji odgovara dobijenom intentu
- Ako se pronađe aktivnost koja može da obradi Intent, ona se startuje metodom onCreate() i prosleđuje joj se dati Intent

Obrada implicitnog intenta

- Ako sistem pronađe više aplikacija koje mogu da reše dati Intent, onda je dobra praksa eksplicitno pozvati *App Chooser* kako bi korisnik izabrao aplikaciju koju želi da pozove
- Dobar primer: klik na Share dugme treba da ponudi listu aplikacija koje mogu da share-uju izabrani resurs (npr. sliku)

Obrada implicitnog intenta

- Primer:

```
val sendIntent = Intent(Intent.ACTION_SEND)

// kreiranje intenta koji prikazuje App Chooser
val title: String = resources.getString(R.string.chooser_title)
val chooser: Intent = Intent.createChooser(sendIntent, title)

// Provera da li će intent biti obrađen od strane makar jedne aktivnosti
if (sendIntent.resolveActivity(packageManager) != null) {
    startActivity(chooser)
}
```

Intent Filter

- Intent filter je način da se aplikacija podesi tako da može da prima specifične intente
- Deklarišu se u AndroidManifest.xml fajlu
- Kada se definiše Intent Filter za neki Activity, drugim aplikacijama je omogućeno da pozivaju taj Activity korišćenjem implicitnih Intent-a

Intent Filter

- Primer Intent Filter-a:

```
<activity
    android:name=".activites.MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Akcije u intentima

- Neke od čestih akcija koje Intent podržava su:
  - ACTION_VIEW – ova akcija se koristi za intent koji treba da pokrene aktivnost koja prikazuje nešto korisniku
  - ACTION_SEND – ova akcija se koristi za tzv. share intent, koji ima cilj da aktivira neku komponentu koja može da podeli podatke (npr. slanje podataka za deljenje na društvenim mrežama)

Extras u intentima

- Intent može da nosi podatke – to se podešava funkcijom setData()
- Pored ovih podataka, Intent može da nosi i podatke koji nemaju URI, i to se postiže pomoću Bundle-a
- Bundle je objekat koji sadrži osnovne tipove podataka i služi da ih „uskladišti“ u Intent
- Bundle se stavlja u Intent pomoću funkcije putExtras()

Extras u intentima

- Kreiranje Bundle-a:

```
val bundle = Bundle()

// skladišti se string koji se mapira na key
bundle.putString("key1", "Neka string vrednost")

// pravi se intent i prosledjuje mu se bundle
intent = Intent(this, SecondActivity::class.java)
intent.putExtras(bundle)

// startovanje aktivnosti
startActivity(intent)
```

Extras u intentima

- Korišćenje Bundle-a:

```
// preuzimanje bundle-a
val bundle = getIntent().getExtras()

// preuzimanje stringa
val title = bundle.getString("key1", "Default")
```

PendingIntent

- PendingIntent je posebna klasa koja *wrappuje* Intent
- Svrha PendingIntent objekta je da pruži mogućnost ispunjavanja Intent-a čak i kad je aplikacija iz koje on potiče ugašena
- *Pending* = nešto što još nije ispunjeno
- Intent u okviru PendingIntent-a se može posmatrati kao buduća instanca koja će se iskoristiti nekada
- Primer primene: slanje notifikacija

PendingIntent

- PendingIntent treba u sebi da sadrži eksplicitni Intent iz sigurnosnih razloga
- Ukoliko PendingIntent sadrži implicitni Intent, nema garancije koja će aplikacija da ga obradi, što može dovesti do problema u sistemu
- Sa eksplicitnim Intent-om, osigurano je da enkapsulirani Intent izvršava tačno ona komponenta kojoj je namenjen
  - Obično je to deo same aplikacije koja kreira PendingIntent

PendingIntent

- PendingIntent može da se koristi za:
  - Pokretanje Activity-ja – kada ga obrađuje Activity
  - Pokretanje Service-a – kada ga obrađuje Service
  - Vršenje broadcast-a – kada ga obrađuje BroadcastReceiver

PendingIntent

- PendingIntent se obično kreira na jedan od tri načina:
  - ***PendingIntent.getActivity()*** – PendingIntent kome se prosleđuje Intent koji startuje Activity
  - ***PendingIntent.getService()*** – prosleđuje mu se Intent koji startuje Service
  - ***PendingIntent.getBroadcast()*** – PendingIntent koji se koristi za broadcast-ovanje

PendingIntent

PendingIntent : *Parcelable, Cloneable*

```
getActivity(Context context, int requestCode, Intent intent, int flags)

getService(Context context, int requestCode, Intent intent, int flags)

getBroadcast(Context context, int requestCode, Intent intent, int flags)
```

PendingIntent za notifikacije

```
val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com/"))
val pendingIntent = PendingIntent.getActivity(this, 0, intent, 0)

val builder = new NotificationCompat.Builder(this)
builder.setSmallIcon(android.R.drawable.ic_dialog_alert)
    .setContentIntent(pendingIntent)
    .setContentTitle("Notification")
    .setContentText("This notification uses pending intents.")

val notificationManager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE)
notificationManager.notify(1, builder.build())
```

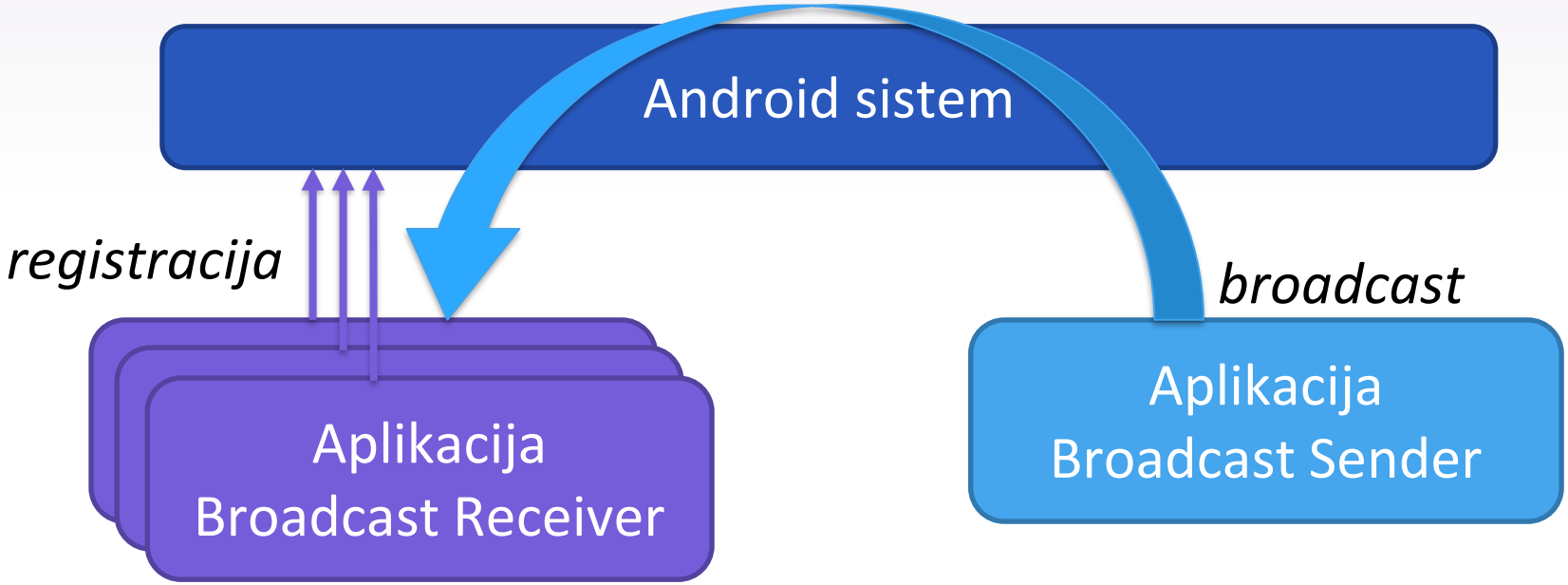
Broadcast u Android-u

- Android sistem podržava koncept broadcast-a
- Svaka aplikacija može da pošalje ili primi broadcast
- Implementacija publish-subscribe pattern-a
- Sam Android šalje broadcast kada se desi neki sistemski događaj (boot-ovanje sistema, početak punjenja uređaja)
- Aplikacije takođe mogu slati svoje custom broadcast-e

Broadcast u Android-u

- Aplikacija može da se registruje da prima određene broadcast-e
- Kada se nešto broadcast-uje, Android sistem pronalazi sve aplikacije koje su registrovane da žele da prime taj broadcast i prosleđuje im ga
- U srcu broadcast-a je Intent

Broadcast u Android-u



Kreiranje broadcast-a

```
val broadcastIntent = Intent("SOME_ACTION")
broadcastIntent.putExtra(
    "message",
    "Broadcast from App1")
sendBroadcast(broadcastIntent)
```

BroadcastReceiver

- Predstavlja base klasu za deo aplikacije koji je zadužen za obradu broadcast-a
- Instanca klase koja nasleđuje BroadcastReceiver treba da se registruje u Android sistemu kako bi aplikacija mogla da obrađuje broadcast
- Registracija BroadcastReceiver-a može se izvršiti dinamički ili statički

Definisanje BroadcastReceiver-a

- Bitno je da klasa nasleđuje BroadcastReceiver
- Bitno je da se predefiniše metoda onReceive()

```
class MyBroadCastReceiver: BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        val msg = intent?.getStringExtra("message")
        Log.i("RECEIVER", msg)
    }
}
```

Registracija BroadcastReceiver-a

- Dinamički (iz koda):

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    ...
    val filter = IntentFilter()
    filter.addAction("SOME_ACTION")
    filter.addAction("SOME_OTHER_ACTION")
    val receiver = MyBroadcastReceiver()
    registerReceiver(receiver, filter)
}
```

Registracija BroadcastReceiver-a

- Statički (AndroidManifest.xml, unutar <application> taga):

```
<receiver android:name="MyBroadcastReceiver">
    <intent-filter>
        <action android:name="SOME_ACTION"/>
    </intent-filter>
</receiver>
```

Registracija BroadcastReceiver-a

- Ako se BroadcastReceiver registruje statički, u trenutku kada se okine odgovarajući broadcast Android sistem kreira nov proces koji će obraditi taj broadcast
- Ako se BroadcastReceiver registruje dinamički, on živi dokle god je komponenta iz koje se registruje aktivna
- Kako bi se izbeglo nepredviđeno ponašanje aplikacije, potrebno je u onPause() metodi izvršiti unregisterReceiver(receiver)

Rezime - Intents

- Intent je objekat koji označava neku akciju
- Koristi se kada treba pozvati neku komponentu unutar aplikacije (eksplicitni Intent) ili kada treba zahtevati akciju od neke druge aplikacije (implicitni Intent)
- Eksplicitni Intent aktivira tačnu klasu (Activity, Service, BroadcastReceiver) koja mu je prosleđena
- Implicitni Intent aktivira neku od aplikacija koja zna kako da obradi akciju koja je prosleđena kroz Intent

Rezime - Intents

- IntentFilter je mehanizam koji omogućava povezivanje komponente aplikacije (Activity ili Service) sa nekim tipom Intent-a
- Specificiraju se atributi Intent-a koji komponenta može da obradi
- PendingIntent je način da se Intent aktivira u budućnosti, kada se desi neki događaj (npr. slanje notifikacija)

Rezime - BroadcastReceivers

- Broadcast je implementacija publish-subscribe pattern-a u Androidu korišćenjem Intent-a
- Aplikacija definiše BroadcastReceiver tako što se kreira klasa koja nasleđuje BroadcastReceiver i predefiniše se metoda onReceive()
- BroadcastReceiver se registruje u sistemu (statički ili dinamički)
- Uz registraciju se prosleđuje i IntentFilter kako bi se receiver podesio da prima samo specifične Intente

Rezime - BroadcastReceivers

- U trenutku slanja nekog broadcast-a, aplikacije koje su se registrovale za taj tip broadcast-a dobijaju njegov Intent od Android sistema
- Slanje broadcast-a se vrši kreiranjem Intent-a za taj broadcast, i zatim pozivanjem funkcije sendBroadcast(intent)

Literatura

- Intents - Android Docs
- Intent Filters - Android Docs
- Broadcast Receivers - Android Docs
- System Broadcasts - Android Docs

Hvala na pažnji!