Servisi i prijemnici poruka

Fakultet tehničkih nauka, Novi Sad

Pregled sadržaja

1 Procesi i niti

2 Rukovaoci

3 Asinhroni zadaci

4 Servisi

5 Prijemnici poruka

Prijemnici poruka

- Prijemnici poruka (`BroadcastReceiver`) obrađuju događaje (opisane objektima klase Intent)
- Te događaje može da izazove Android platforma ili druga komponenta (koja može da se nalazi u drugoj aplikaciji)

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest ... >
3    <application ... >
4      <receiver android:name=".SMSReceiver">
5        <intent-filter>
6          <action name="android.provider.Telephony.SMS_RECEIVED" />
7        </intent-filter>
8      </receiver>
9    </application>
10 </manifest>
```

SMSReceiver.java

```
1  public class SMSReceiver extends BroadcastReceiver {
2    @Override
3      public void onReceive(Context context, Intent intent) {
4        // ...
5    }
6  }
```

Prijemnici poruka

Parametri `onReceive` metode su:

- `context` (kontekst u kome se izvršava onReceive metoda)
- `intent` (namera koja opisuje događaj koji treba obraditi)

Namera prosleđena `startActivity` ili `startService` metodi neće prouzrokovati događaj koji će obraditi `onReceive` metoda (važi i obrnuto)

Prijemnici poruka

Konstanta	Značenje
`ACTION_BATTERY_LOW`	A warning that the battery is low.
`ACTION_HEADSET_PLUG`	A headset has been plugged into the device, or unplugged from it.
`ACTION_SCREEN_ON`	The screen has been turned on.
`ACTION_SHUTDOWN`	Device is shutting down.

Tabela: Neke od akcija (događaja) koje se mogu obraditi.

Prijemnici poruka

- Metoda `onReceive` se poziva iz glavne niti (to znači da "dugačke" operacije treba izvršavati u posebnom servisu koji startuje posebnu nit)
- Prijemnik poruka postoji samo u toku izvršavanja `onReceive` metode (zato se u ovoj metodi ne mogu izvršiti asinhrone operacije kao što su prikazivanje dijaloga ili vezivanje za servis)
- Proces u kome se izvršava `onReceive` metoda ima foreground prioritet (nakon toga, prioritet procesa određuju ostale komponente koje se u njemu nalaze)

Prijemnici poruka

Postoje dve vrste događaja koje prijemnici poruka mogu da obrade:

- normalni događaji (asinhroni su, prijemnici ih obrađuju nedefinisanim redosledom i njigova obrada je efikasnija)
- uređeni događaji (obrađuje ih više prijemnika redom, a svaki prijemnik može da prosledi događaj sledećem prijemniku ili da potpuno obustavi njegovu obradu)

ExampleActivity.java

```
1  public class ExampleActivity extends Activity {
2    @Override
3    public void onCreate(Bundle bundle) {
4      Intent intent = new Intent();
5      intent.setAction(SMS_RECEIVED);
6      sendBroadcast(intent);
7    }
8  }
```

SMSReceiver.java

```
1  public class SMSReceiver extends BroadcastReceiver {
2    @Override
3    public void onReceive(Context context, Intent intent) {
4      // ...
5    }
6  }
```

ExampleActivity.java

```
1  public class ExampleActivity extends Activity {
2    @Override
3    public void onCreate(Bundle bundle) {
4      Intent intent = new Intent();
5      intent.setAction(SMS_RECEIVED);
6      sendOrderedBroadcast(intent, null);
7    }
8  }
```

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest ... >
3    <application ... >
4      <receiver android:name=".SMSReceiver">
5        <intent-filter android:priority="100">
6          <action name="android.provider.Telephony.SMS_RECEIVED" />
7        </intent-filter>
8      </receiver>
9    </application>
10 </manifest>
```

SMSReceiver.java

```
1  public class SMSReceiver extends BroadcastReceiver {
2    @Override
3    public void onReceive(Context context, Intent intent) {
4      if (condition) abortBroadcast() {
5        // ...
6      }
7    }
8  }
```

Prijemnici poruka

Metoda	Značenje
`abortBroadcast()`	Sets the flag indicating that this receiver should abort the current broadcast.
`getResultCode()`	Retrieve the current result code, as set by the previous receiver.
`getResultData()`	Retrieve the current result data, as set by the previous receiver.
`setResultCode(int code)`	Change the current result code of this broadcast.
`setResultData(String data)`	Change the current result data of this broadcast.

Tabela: Metode koje se koriste za spregu između prijemnika poruka.

Prijemnici poruka

- Prijemnici poruka omogućavaju razmenu poruka između komponenti različitih aplikacija
- To znači da treba obratiti pažnju na moguće zloupotrebe
- Moguće je primeniti prava pristupa prilikom slanja poruke (prosleđujući permission parametar `sendBroadcast` metodi) ili prilikom prijema poruke (postavljajući permission atribut receiver elementa)
- Komponenta koja salje/prima poruku mora imati odgovarajuće pravo pristupa (zatraženo `<uses-permission>` elementom u `AndroidManifest.xml`)

Agenda

1 Procesi i niti

2 Rukovaoci

3 Asinhroni zadaci

4 Servisi

5 Prijemnici poruka

6 Zakazivanje zadataka

Zakazivanje

- Tajmer (`Timer`) zakazuje izvršavaje jednokratnih zadataka (u apsolutnom trenutku ili posle relativnog kašnjenja) ili zadataka koji se ponavljaju (sa fiksim periodom ili fiksnom frekvencijom)
- Svaki tajmer ima jednu nit koja zadatke izvršava sekvencijalno (to znači da može da dođe do kašnjenja u izvršavanju zadatka ukoliko je ta nit zauzeta)
- Komponenta koja je zakazala izvršavanje zadatka ne mora biti aktivna u trenutku u kome zadatak treba da se izvrši (to znači da zadatak može da se ne izvrši)

Zakazivanje

Metoda	Značenje
`schedule(TimerTask task, Date when)`	Schedule a task for repeated fixed-delay execution after a specific time has been reached.
`schedule(TimerTask task, long delay)`	Schedule a task for single execution after a specified delay.
`cancel()`	Cancels the Timer and all scheduled tasks.

Tabela: Metode klase `Timer`.

Zakazivanje

Metoda	Značenje
`schedule(TimerTask task, Date when, long period)`	Schedule a task for repeated fixed-delay execution after a specific time has been reached.
`schedule(TimerTask task, long delay, long period)`	Schedule a task for repeated fixed-delay execution after a specific delay.
`scheduleAtFixedRate (TimerTask task, long delay, long period)`	Schedule a task for repeated fixed-rate execution after a specific delay has passed.
`scheduleAtFixedRate (TimerTask task, Date when, long period)`	Schedule a task for repeated fixed-rate execution after a specific time has been reached.

Tabela: Metode klase texttttTimer.

SplashScreen.java

```
1  public class SplashScreen extends Activity {
2
3    public static final int SPLASH_TIMEOUT = 1500;
4
5    @Override
6    protected void onCreate(Bundle bundle) {
7      super.onCreate(bundle);
8      setContentView(R.layout.splash_screen);
9      new Timer().schedule(new TimerTask() {
10         @Override
11         public void run() {
12           startActivity(new Intent(SplashScreen.this, MyMovies.class));
13           finish();
14         }
15       }, SPLASH_TIMEOUT);
16   }
17 }
```

Zakazivanje

- Klasa `AlarmManager` omogućuje pristup sistemskom alarmu i startovanje aplikacije u nekom trenutku u budućnosti
- Kada se alarm aktivira, sistem emituje objekat klase `Intent`, što kao posledicu ima automatsko startovanje aplikacije (ukoliko već nije startovana)

PortfolioStartupReceiver.java

```
1  public class PortfolioStartupReceiver extends BroadcastReceiver {
2
3    private static final int FIFTEEN_MINUTES = 15*60*1000;
4
5    @Override
6    public void onReceive(Context context, Intent intent) {
7      AlarmManager mgr =
8        (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
9      Intent i = new Intent(context, AlarmReceiver.class);
10     PendingIntent sender = PendingIntent.getBroadcast(
11       context, 0, i, PendingIntent.FLAG_CANCEL_CURRENT);
12     Calendar now = Calendar.getInstance();
13     now.add(Calendar.MINUTE, 2);
14     mgr.setRepeating(AlarmManager.RTC_WAKEUP,
15     now.getTimeInMillis(),FIFTEEN_MINUTES, sender);
16   }
17 }
```

AlarmReceiver.java

```
1  public class AlarmReceiver extends BroadcastReceiver {
2    @Override
3    public void onReceive(Context context, Intent intent) {
4      Intent stockService = new Intent(context, PortfolioManagerService.class);
5      context.startService(stockService);
6    }
7  }
```

PortfolioManagerService.java

```
1  public class PortfolioManagerService extends Service {
2    @Override
3    public int onStartCommand(Intent intent, int flags, int startId) {
4      updateStockData();
5      return Service.START_NOT_STICKY;
6    }
7  }
```