Servisi i prijemnici poruka

Fakultet tehničkih nauka, Novi Sad

Pregled sadržaja

1. Procesi i niti
2. Rukovaoci
3. Asinhroni zadaci

Proces

- aktivnost (angažovanje procesora na izvršavanju programa)
- slika (deo operativne memorije koji sadrži naredbe u mašinskom jeziku i podatke na stack-u i heap-u)
- atributi (stanje, prioritet, itd.)

Nit

- Nit je redosled izvršavaja naredbi u procesu
- Jedan proces može da sadrži više niti
- Onda svaka nit sadrži stack, stanje i prioritet i izvršava relativno nezavisnu sekvencu naredbi

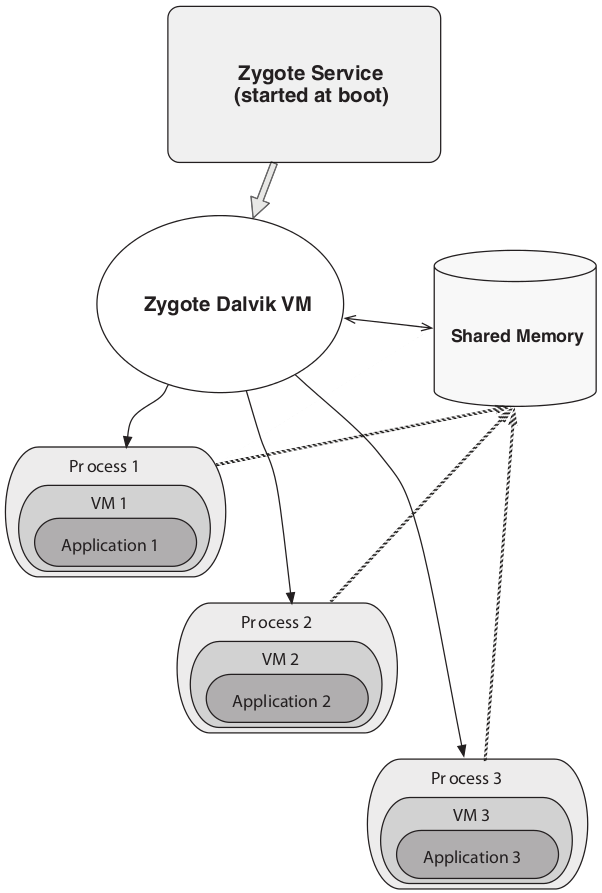
Razlika između procesa i niti

- Niti se koriste za "male" zadatke, a procesi za "velike" zadatke (izvršavanje aplikacije)
- Niti koje pripadaju istom procesu dele isti adresni prostor (to znači da mogu da komuniciraju direktno preko operativne memorije)
- Procesi ne dele isti adresni prostor (to znači da je komunikacija između procesa složenija i sporija od komunikacije između niti)

Raspoređivanje niti

- Različite niti mogu da se izvršavaju na jednom procesoru (konkurentno) ili na više procesora (paralelno)
- Kako jedan procesor ne može istovremeno da izvršava više niti, one se moraju izvršavati naizmenično
- Potrebno je voditi računa o sinhronizaciji niti

Android i procesi



Slika: Android i procesi.

- Kada Android startuje prvu komponentu neke aplikacije, startuje je u novom procesu sa jednom niti
- Svaka sledeća komponenta iste aplikacije startuje se u istom procesu i u istoj niti kao i prva komponenta
- Moguće je startovati različite komponente iste aplikacije u različitim procesima ili različite komponente različitih aplikacija u istom procesu (mada to nije preporučljivo)

Android i procesi

- Android zadržava procese u operativnoj memoriji što je duže moguće
- Da bi se slobodila memorija za procese višeg prioriteta, nekada je potrebno "ubiti" proces nižeg prioriteta
- Prioritet procesa se određuje na osnovu vrste i stanja komponenti koje sadrži kao i prioriteta drugih procesa koji od njega zavise
- Zato bi aktivnosti i prijemnici poruka koji izvršavaju dugačke operacije trebalo da startuju servis umesto niti

Android i procesi

Procesi mogu imati jedan od pet prioriteta (ukoliko proces zadovoljava više uslova ima najviši prioritet)

- `foreground` (proces sadrži aktivnost koja se nalazi u prvom planu, servis koji je vezan za aktivnost koja se nalazi u prvom planu, servis koji se izvršava u prvom planu, servis koji izvršava jednu od metoda životnog ciklusa ili prijemnik poruka koji izvršava onReceive metodu)
- `visible` (proces sadrži vidljivu aktivnost ili servis koji je vezan za vidljivu aktivnost)
- `service` (proces sadrži servis)
- `background` (proces sadrži aktivnost koja se nalazi u pozadini)
- `empty` (proces ne sadrži komponente)

Android i niti

- Android izvršava aplikaciju (tj. njene komponente) u glavnoj niti
- Ova nit je, između ostalog, zadužena za slanje i primanje poruka od komponenti korisničkog interfejsa (zato se zove i UI nit)
- Stoga nije preporučljivo blokirati UI nit ("application isn't responding" dijalog) i pristupati komponentama korisničkog interfejsa iz drugih niti (nisu thread-safe)
- Metode životnog ciklusa servisa i dobavljača sadržaja moraju biti thread-safe

Android i niti

```
1  // Pogresno (blokiranje UI niti)
2  public void onClick(View v) {
3    Bitmap b = loadImageFromNetwork(imageUrl);
4    imageView.setImageBitmap(b);
5  }
```

Android i niti

```
1  // Pogresno (GUI komponente nisu thread-safe)
2  public void onClick(View v) {
3    new Thread(new Runnable() {
4      public void run() {
5        Bitmap b = loadImageFromNetwork(imageUrl);
6        imageView.setImageBitmap(b);
7      }
8      }).start();
9  }
```

Android i niti

```
1  // Ispravo
2  public void onClick(View v) {
3    new Thread(new Runnable() {
4      public void run() {
5        Bitmap b = loadImageFromNetwork(imageUrl);
6        imageView.post(new Runnable() {
7          public void run() {
8            imageView.setImageBitmap(b);
9          }
10       });
11     }
12   }).start();
13 }
```

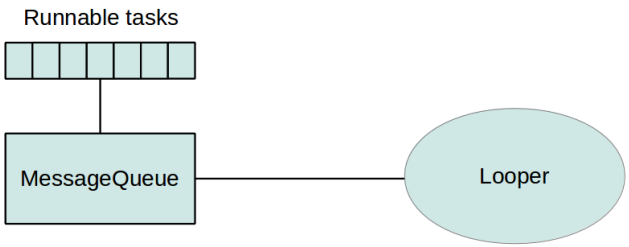
Pregled sadržaja

1 Procesi i niti

2 Rukovaoci

3 Asinhroni zadaci

Rukovaoci



Slika: Obrada poruka.

- Rukovaoci (`Handlers`) omogućavaju obradu poruka (instanci Message klase) i izvršavanje proizvoljnog programskog koda (implementacija Runnable interfejsa) u određenoj niti
- Svaki rukovalac je pridružen niti koja ga je napravila i njenom redu za poruke)

Rukovaoci

- Za obradu poruka potrebno je implementirati void handleMessage(Message msg) i pozvati
  - `boolean sendEmptyMessage(int)`
  - `boolean sendMessage(Message)`
  - `boolean sendMessageAtTime(Message, long)`
  - `boolean sendMessageDelayed(Message, long)`
- Za izvršavanje proizvoljnog koda pozvati
  - `boolean post(Runnable)`
  - `boolean postAtTime(Runnable, long)`
  - `boolean postDelayed(Runnable, long)`

Rukovaoci

- Moguće je iskoristiti postojeći rukovaoc pridružen glavnoj niti

```
1  // Reuses existing handler
2  handler = getWindow().getDecorView().getHandler();
3
4  // Uses UI component's post() method
5  ImageView imageView = (ImageView) findViewById(R.id.image_view);
6  imageView.post(new Runnable() {
7    @Override
8    public void run() {
9      imageView.setImageBitmap(bitmap);
10   }
11 });
```

Pregled sadržaja

1 Procesi i niti

2 Rukovaoci

3 Asinhroni zadaci

Asinhroni zadatak

- Asinhrnoni zadatak (`AsyncTask`) olakšava asinhrono izvršavanje operacija
- Automatski izvršava blokirajuću operaciju u pozadinskoj niti, vraća rezultat UI niti i nudi dodatne funkcije (kao što je obaveštavanje o progresu operacije)
- Svi asinhroni zadaci jedne aplikacije izvršavaju se u jednoj niti (oni se serijalizuju)

Asinhroni zadatak

```
1  public void onClick(View v) {
2    new DownloadImageTask().execute(imageUrl);
3  }
```

Asinhroni zadatak

```
1  private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
2    // The system calls this to perform work in a worker
3    // thread and delivers it the parameters given to
4    // execute()
5    protected Bitmap doInBackground(String... urls) {
6      return loadImageFromNetwork(urls[0]);
7    }
8
9    // The system calls this to perform work in the UI
10   // thread and delivers the result from doInBackground()
11   protected void onPostExecute(Bitmap result) {
12     imageView.setImageBitmap(result);
13   }
14 }
```

Asinhroni zadatak

AsyncTask je generička klasa koja koristi tri tipa:

- `params` (tip parametara koji se prosleđuju pozadinskoj niti)
- `progress` (tip jedinice u kojoj se meri progres operacije)
- `result` (tip povratne vrednosti koju vraća pozadinska nit)

Asinhroni zadatak

i sadrži četiri metode:

- `void onPreExecute()` - poziva se u UI niti pre izvršavanja zadatka
- `Result doInBackground(Params... params)` - poziva se u pozadinskoj niti odmah posle onPreExecute
- `void onProgressUpdate(Progress... values)` - poziva se u UI niti posle poziva publishProgress u pozadinskoj niti
- `void onPostExecute(Result result)` - poziva se u UI niti posle izvršavanja zadatka