Introdução aos dispositivos móveis
Services & Comunicação entre processos

Marcelo Zamith

e-mail: mzamith@ufrrj.br
https://www.dcc.ufrrj.br/~marcelo/

Universidade Federal Rural do Rio de Janeiro - DCC

Introdução

Introdução

Service

Um *Service* é um processo. É um componente do aplicativo empregado para processamento de longa duração e não possui interface com o usuário. É executado em segundo plano.

Introdução

- Um service por ser executado:
  - Não vinculado :
    - Primeiro plano (**`Service`**): executa de acordo com o aplicativo, mesmo sem estar em primeiro plano. Exemplo: a faixa de uma música
    - Segundo plano (**`IntentService`**): é chamado do aplicativo e utilizado para realizar tarefas de comunicação;
  - Vinculado (**`Service`**): executa do aplicativo e cria uma estrutura cliente servidor, utilizando para grande volumes de processamento
- Estrutura de comunicação entre os processos ←

Não vinculado **`Service`**

Não vinculado **`Service`**

Execução em primeiro plano

- Service iniciado pelo método: `onStartCommand`
- Parâmetros:
  - Intent intent: intent que chamou o serviço, utilizado para obter os parâmetros;
  - int flags: flags de chamada (START_FLAG_REDELIVERY e START_FLAG_RETRY)
  - int startId: inteiro único que representa a requisição do serviço;
- retorna um inteiro:
  - `START_STICKY`: Indica que o serviço foi inicializado e pode terminar em um outro tempo, por ação ou não do usuário;
  - `START_NOT_STICKY`: O serviço começa e pode ser interrompido, podendo retornar de onde parou;
  - `START_REDELIVER_INTENT`: Caso o serviço seja interrompido, será reinicializado, utilizando por base o intent.

Não vinculado **Service**

Execução em primeiro plano

- Chamada do `service`

```
1     ...
2   startService(new Intent(this, PlayFG.class));
3     ...
4   stopService(new Intent(this, PlayFG.class));
5     ...
```

- Métodos do `service`

```
1   @Override
2   public int onStartCommand(Intent intent, int flags, int startId) {
3
4       player = MediaPlayer.create( this, Settings.System.DEFAULT_RINGTONE_URI );
5       player.setLooping( true );
6       player.start();
7
8       return START_STICKY;
9   }
10  @Override
11  public void onDestroy() {
12      super.onDestroy();
13      player.stop();
14  }
```

Não vinculado **`IntentService`**

Não vinculado **IntentService**

Execução em segundo plano

- Executa conforme demanda

```
1       ...
2   Intent intent = new Intent(this, DownloadBG.class);
3   File dirname = Environment.
4       getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
5   intent.putExtra(DownloadBG.EXTRA_PARAM_DIR, dirname.getPath());
6   intent.putExtra(DownloadBG.EXTRA_PARAM_FILE, "aulas.zip");
7   startService(intent);
8       ...
```

- A classe `DownloadBG` implementa o método:

```
1   @Override
2   protected void onHandleIntent(Intent intent) {
3       if (intent != null) {
4           mDirName = intent.getStringExtra(EXTRA_PARAM_DIR);
5           mFileName = intent.getStringExtra(EXTRA_PARAM_FILE);
6           download();
7       }else{
8           sendMessage("error");
9       }
10  }
```

Vinculado **`Service`**

Vinculado **`Service`**

- Utiliza uma estrutura cliente/servidor;
- Instancia o service através da função `bindService`
- Necessidade de trabalhar com uma classe que implemente `ServiceConnection`
  - `onServiceConnected`: conectar ao service
  - `onServiceDisconnected`: desconectar do service, libera os recursos

Vinculado **Service**

- Chamada

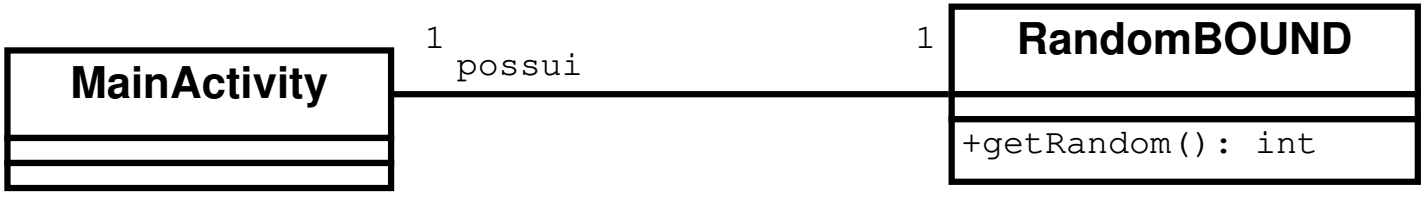
```
1       ...
2   Intent intent = new Intent(this, RandomBOUND.class);
3   mServiceConn = new MyServiceConnection();
4   bindService(intent, mServiceConn, Context.BIND_AUTO_CREATE);
5       ...
```

- Classe `RandomBOUND`:

```
1   @Override
2   public IBinder onBind(Intent intent) { return mIBinder; }
3
4   public class LocalBinder extends Binder{
5      RandomBOUND getService(){return RandomBOUND.this; }
6   }
```

- A função - ideia de RPC

```
1       ...
2   public int getRandom(){ return mGen.nextInt(100);}
3       ...
```



UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
UFRRJ

Comunicação

Comunicação

Intent

- `Intent`:
  - permite chamar componentes da aplicação
  - passar parâmetros:
    - tipos básicos: string, int, double
    - tipo classe - `implements Parcelable`

Comunicação

BroadcastReceiver

- `BroadcastReceiver`:
  - trata eventos e mensagens recebidas de outros componentes;
  - abre um canal de comunicação - apenas recebe
  - executa tarefas de curta duração (≈ 10 segundos)
  - método `onReceive`
- componente implementa - recebe a mensagem;
- comunicação em um sentido;
- registrar canal e liberar recurso;

Comunicação

BroadcastReceiver

- Tratamento da mensagem - evento

```
1  private class MyBroadcastReceive extends BroadcastReceiver{
2      @Override
3      public void onReceive(Context context, Intent intent) {
4          String message = intent.getStringExtra("message");
5          ...
6      }
7  }//private class MyBroadcastReceive extends BroadcastReceiver{
```

- Registro do objeto BroadcastReceiver "`From-Download-To-MainActivity`"

```
1  ...
2  mMessageReceiver = new MyBroadcastReceive();
3  LocalBroadcastManager.getInstance(this).
4      registerReceiver(mMessageReceiver,
5      new IntentFilter("From-Download-To-MainActivity"));
6  ...
```

- Liberação do recurso

```
1      ...
2  LocalBroadcastManager.getInstance(this).unregisterReceiver(mMessageReceiver);
3      ...
```

Comunicação

BroadcastReceiver

- Componente cliente;
- LocalBroadcastManager;
- Utilização de Intent - comunicação;

```
1       ...
2   Intent myintent = new Intent("From-Download-To-MainActivity");
3   myintent.putExtra("message", msg);
4   LocalBroadcastManager.getInstance(this).sendBroadcast(myintent);
5       ...
```

Considerações finais

- Componente service
- LocalBroadcastManager e BroadcastReceiver
- Acesso a internet, bluetooth e tarefas assíncronas
- Mostrar código e comentar permissões

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
UFRRJ