


Android

Processamento em Segundo Plano com Services




Tópicos Abordados



- Introdução
- A classe *Service*
- Configurando services
- Executando e parando services
- Implementando services
- Conectando-se a services existentes
- IntentServices

Introdução



- Services são componentes do Android que permitem executar processamento em segundo plano
 - Não estão ligados a uma interface gráfica
- Os services em execução são considerados de alta prioridade pelo Android
 - Só perdem para a activity visível
 - O Android destrói outros componentes antes de destruir os services

A Classe Service



- Para criar um service, é preciso estender a classe **Service**

```
public class CounterService extends Service {  
  
    public void onCreate() {  
    }  
  
    public int onStartCommand(Intent intent, int flags, int startId) {  
    }  
  
    public void onDestroy() {  
    }  
  
    public IBinder onBind(Intent intent) {  
        return null;  
    }  
}
```

Configurando os Services



- Os services são configurados no arquivo *AndroidManifest.xml*

```
<service android:name=".CounterService">  
    <intent-filter>  
        <action android:name="START_COUNT" />  
    </intent-filter>  
</service>
```

Services são iniciados
através de intents

Executando um Service



- Um service pode ser iniciado por uma activity, broadcast receiver ou outro service
- Uma das formas de iniciá-lo é através do método **startService()**

```
MyActivity.java  
  
Intent intent = new Intent(this, CountService.class);  
startService(intent);  
  
Intent intent = new Intent("START_COUNTER");  
startService(intent);
```

Executando um Service



- Depois que o service é iniciado com `startService()`, ele fica desvinculado de quem o criou
 - O service executa de forma independente
- Services não são processos ou threads
 - Eles executam na UI thread
 - É recomendado que o código do service seja executado em uma thread separada, que deve ser criada pelo programador

Parando um Service



- Para parar a execução de um service iniciado com `startService()`, existem duas formas
 - Invocar o método `stopService()`, fornecendo a intent que corresponde ao service

```
Intent intent = new Intent(this, CounterService.class);  
stopService(intent);
```

- O próprio service invocar o método `stopSelf()`

```
stopSelf();
```

Escrevendo um Service



- Um service deve estender da classe `Service`
 - O método `IBinder onBind(Intent)` deve ser implementado
- Métodos de callback normalmente utilizados
 - `onCreate()`
 - Invocado quando o service é criado
 - `onStartCommand(Intent, int, int)`
 - Invocado toda vez que `startService()` é chamado
 - `onDestroy()`
 - Invocado quando o service é destruído

Métodos de Callback



- Algumas considerações importantes sobre os métodos de callback quando o service é iniciado com `startService()`
 - `onCreate()`
 - É executado apenas uma vez para o service
 - `onStartCommand(Intent, int, int)`
 - Chamado toda vez que o método `startService()` é invocado
 - O Android associa um `startId` diferente cada vez que `startService()` é invocado
 - `onDestroy()`
 - Executado apenas uma vez, quando o service é destruído

Exemplo de Service



```
public class CounterService extends Service {  
    private CounterWorker worker;  
  
    public void onCreate() {  
        worker = new CounterWorker(this);  
    }  
  
    public int onStartCommand(Intent intent, int flags, int startId) {  
        new Thread(worker).start();  
        return START_STICKY;  
    }  
  
    public void onDestroy() {  
        worker.stop();  
    }  
  
    public IBinder onBind(Intent intent) {  
        return null;  
    }  
}
```

Exemplo de Service



```
public class CounterWorker implements Runnable {  
    private CounterService service;  
    boolean active = true;  
  
    public CounterWorker(CounterService service) {  
        this.service = service;  
    }  
  
    public void run() {  
        for (int i = 0; i < 100 && active; i++) {  
            counter++;  
            Log.d("App", "Counter => " + counter);  
        }  
        service.stopSelf();  
    }  
  
    public void stop() {  
        active = false;  
    }  
}
```

Conectando-se a um Service



- O Android proporciona uma forma para que seu código possa se conectar a um serviço em execução
 - Obter uma referência do serviço para invocar métodos
- Isto é feito através dos métodos **bindService()** e **unbindService()**

Binder



- Um objeto chamado binder é responsável por fazer a "ponte" entre o seu código e o service

```
public class CounterServiceBinder extends Binder {  
    private CounterService service;  
  
    public CounterServiceBinder(CounterService service) {  
        this.service = service;  
    }  
  
    public CounterService getService() {  
        return service;  
    }  
}
```

Estende de
Binder

Retorna
referência para
o service

Binder



```
public class CounterService extends Service {  
    private IBinder binder = new CounterServiceBinder(this);  
  
    public IBinder onBind(Intent intent) {  
        return binder;  
    }  
}
```

O método **onBind()** do
service deve retornar o
binder criado

Service Connection



- Para que possa haver a conexão ao serviço, é preciso uma referência a um objeto do tipo **ServiceConnection**
- Os métodos dessa classe são invocados pelo Android
 - *onServiceConnected()*
 - Quando a conexão com o serviço foi feita com sucesso
 - *onServiceDisconnected()*
 - Quando a conexão com o serviço foi terminada

Service Connection



```
public class CounterServiceConnection implements ServiceConnection {  
    private CounterService counter;  
  
    public void onServiceConnected(ComponentName component,  
        IBinder binder) {  
  
        CounterServiceBinder counterBinder =  
            (CounterServiceBinder) binder;  
        this.counter = counterBinder.getService();  
    }  
  
    public void onServiceDisconnected(ComponentName component) {  
    }  
}
```

Recebe a referência
do service

```
CounterServiceConnection conn = new CounterServiceConnection();  
Intent intent = new Intent(this, CounterService.class);  
bindService(intent, conn, Context.BIND_AUTO_CREATE);
```

```
unbindService(conn);
```

Cria o service caso
ele não exista

Mais Sobre Conexão a Services



- Um service iniciado com *bindService()* (através da flag *BIND_AUTO_CREATE*)
 - Fica atrelado ao componente que o criou
 - Por exemplo, se a activity que criou o service for destruída, o service também é destruído
 - Invoca o método *onCreate()* no service
 - Nunca invoca o método *onStartCommand()*
- O mais comum é usar *startService()* primeiro e *bindService()* depois

A Classe *IntentService*



- Um service que herda de **IntentService** executa suas tarefas em uma thread a parte
 - IntentService herda de *Service*
 - É preciso implementar o método **onHandleIntent()**
 - Não é preciso se preocupar em criar uma thread manualmente
 - É criada uma thread a parte (worker) que executa as tarefas sequencialmente
 - Também não é preciso se preocupar com a parada do service
 - Ele termina automaticamente ao final da execução da tarefa

A Classe *IntentService*



```
public class MyService extends IntentService {  
    public MyService() {  
        super("MyServiceWorker");  
    }  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        // Tarefa a ser executada  
    }  
}
```

Herda de *IntentService*

Fornece um nome para a thread no construtor

Método que executa a tarefa em uma thread a parte

```
Intent intent = new Intent(this, MyService.class);  
startService(intent);
```

A inicialização é igual a qualquer service