


Android

Recebendo Eventos com Broadcast Receivers




Softblue
cursos online

Tópicos Abordados



- O que são e como funcionam os broadcast receivers
- A classe *BroadcastReceiver*
- Configuração
 - Estática e dinâmica
- Enviando mensagens e mensagens ordenadas
- Ciclo de vida
- Recomendações
- Eventos de broadcast nativos do Android
- Broadcasts locais

Broadcast Receivers

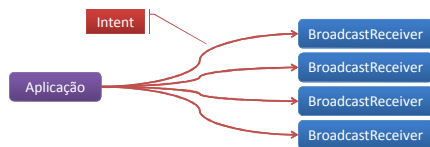


- Um componente broadcast receiver funciona de forma análoga a uma activity
 - Iniciados através de intents
 - Usam intent filters para decidir quais intents devem receber
- O broadcast receiver não trabalha atrelado à interface gráfica
 - Ele executa em segundo plano, sem o conhecimento do usuário

Broadcast Receivers



- Os broadcast receivers reagem a eventos disparados em forma de intents
 - Estes eventos podem ser das suas próprias aplicações ou de aplicações nativas do Android



A Classe BroadcastReceiver



- Um broadcast receiver é representado pela classe **BroadcastReceiver**
- Basta estender esta classe e implementar o método **onReceive()**

```
public class MyBroadcastReceiver extends BroadcastReceiver {  
    public void onReceive(Context context, Intent intent) {  
        //...  
    }  
}
```

Contexto de invocação

Intent utilizada

Configurando um BroadcastReceiver



- A configuração pode ser feita de duas formas
 - Estática (no arquivo *AndroidManifest.xml*)
 - Dinâmica (no código)
- Na configuração estática, o broadcast receiver pode receber a intent mesmo que a aplicação não esteja sendo executada
- Na configuração dinâmica, a aplicação deve estar executando

Configuração Estática



- Broadcast receivers são configurados no arquivo *AndroidManifest.xml*

```
<receiver android:name=".MyReceiver">  
  <intent-filter>  
    <action android:name="company.android.action.COMPLETED" />  
  </intent-filter>  
</receiver>
```

Classe do broadcast receiver

Intent filters que determinam quais intents podem ser recebidas

Configuração Dinâmica



- Broadcast receivers são configurados pelo código
- São usados dois métodos
 - **registerReceiver()**
 - **unregisterReceiver()**

Configuração Dinâmica



```
public class MyActivity extends Activity {  
  private MyReceiver receiver;  
  private IntentFilter filter;  
  
  public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    receiver = new MyBroadcastReceiver();  
    filter = new IntentFilter();  
    filter.addAction("company.android.action.COMPLETED");  
  }  
  
  protected void onResume() {  
    super.onResume();  
    registerReceiver(receiver, filter);  
  }  
  
  protected void onPause() {  
    super.onPause();  
    unregisterReceiver(receiver);  
  }  
}
```

Enviando Mensagens



- Para que os broadcast receivers registrados na plataforma recebam as intents, elas devem ser disparadas
- Isto é feito através do método **sendBroadcast()**

```
Intent i = new Intent("company.android.action.COMPLETED");  
sendBroadcast(i);
```

Enviando Mensagens




- O método **sendBroadcast()** é assíncrono
- A intent é enviada a todos os broadcast receivers ao mesmo tempo
- Não existe garantia de ordem na entrega das intents

Broadcasts Ordenados



- Para estabelecer uma ordem de envio de broadcast, é possível enviar broadcasts ordenados
- Um broadcast é enviado de cada vez, de acordo com uma prioridade
- Quem recebe o broadcast pode alterar dados ou até cancelar a propagação
- O método **sendOrderedBroadcast()** é utilizado

Configurando as Prioridades



AndroidManifest.xml


```

<receiver android:name=".Receiver1">
  <intent-filter android:priority="30">
    <action android:name="BROADCAST_EVENT"></action>
  </intent-filter>
</receiver>
<receiver android:name=".Receiver2">
  <intent-filter android:priority="40">
    <action android:name="BROADCAST_EVENT"></action>
  </intent-filter>
</receiver>
<receiver android:name=".Receiver3">
  <intent-filter android:priority="50">
    <action android:name="BROADCAST_EVENT"></action>
  </intent-filter>
</receiver>

```


Quanto maior o número, maior a prioridade

Interferindo na Propagação



- Se um broadcast receiver desejar cancelar a propagação da intent, basta chamar o método **abortBroadcast()**
- O método **getResultExtras()** retorna o objeto *Bundle* com informações extras
 - Quando um broadcast receiver coloca ou altera uma informação, isto é refletido para os próximos broadcast receivers que serão executados

Ciclo de Vida



- O objeto que representa o broadcast receiver fica ativo apenas durante a execução do método *onReceive()*
 - Depois disso o Android pode destruí-lo
- Devido a isso, operações assíncronas não são permitidas
- O método *onReceive()* deve completar em até 10 segundos
 - Não é possível realizar processamento "pesado"

Recomendações



- Broadcast receivers não devem interferir no que o usuário está fazendo
 - Exibir caixas de diálogo, abrir activities, etc.
 - A forma correta é utilizar o serviço de notificação para avisar o usuário
- Apesar de executar em segundo plano, os broadcast receivers não devem executar código que demanda tempo
 - Limite de 10 segundos para execução
 - Services são recomendados nestes casos

Eventos de Broadcast Nativos



- O Android faz o broadcast de intents para avisar sobre eventos internos do sistema
- As aplicações podem registrar broadcast receivers para serem notificadas a respeito destes eventos
 - `ACTION_POWER_CONNECTED`
 - `ACTION_POWER_DISCONNECTED`
 - `ACTION_BOOT_COMPLETED`
 - `ACTION_BATTERY_CHANGED`
 - etc.

Eventos de Broadcast Nativos



- A partir do Android 8 (API Level 26), broadcasts nativos não podem ser interceptados por broadcast receivers configurados de forma estática
 - A configuração precisa ser dinâmica

Broadcasts Locais



- Um objeto **LocalBroadcastManager** permite enviar broadcasts locais
 - Apenas a sua própria aplicação recebe os broadcasts
 - Outras aplicações externas não podem enviar o broadcast para a sua aplicação
- Esta classe pertence à API de compatibilidade

Trabalhando com Broadcasts Locais



- Obter uma instância da classe

```
LocalBroadcastManager lbm = LocalBroadcastManager  
.getInstance(context);
```

- Enviar broadcasts

```
lbm.sendBroadcast(intent);
```

- Registro de broadcast receivers

```
lbm.registerReceiver(receiver, intentFilter);
```

```
lbm.unregisterReceiver(receiver);
```

Exportando Broadcast Receivers



- O atributo **exported** pode ser utilizado para indicar se um broadcast receiver é exportado ou não

```
<receiver ...  
  exported = "true" />
```

Um broadcast receiver exportado pode receber broadcasts de outros aplicativos

```
<receiver ...  
  exported = "false" />
```

Um broadcast receiver não exportado pode receber apenas broadcasts do próprio aplicativo

- Se o atributo for omitido
 - Assume *false* se não houver um intent filter
 - Assume *true* se houver um intent filter