Dispositivos Móveis

Aula 06 - Activities II

Apresentação

Vídeo 01 - Apresentação

Nos encontramos novamente! Dessa vez, para continuarmos nossa aula sobre Activities. Na primeira aula, vimos um pouco sobre como implementar uma Activity, seu ciclo de vida e seus callbacks. Agora, continuaremos nossos estudos com a parte relacionada à comunicação entre Activities. Conheceremos os Intents mais a fundo e como podemos utilizá-los para criar um fluxo de Activities em nossa aplicação. Além de conhecer os Intents implícitos e explícitos, iremos estudar os Intent Filters que podem ser utilizados no AndroidManifest para que sua Activity responda corretamente às chamadas que a dizem respeito. Veremos também como passar dados entre Activities utilizando os Intents e como isso pode facilitar a utilização da aplicação. Outro ponto importante para a comunicação entre Activities é como utilizá-la para obter um resultado para outra Activity. Esse processo também será visto nesta aula. Para finalizar, veremos brevemente o que é o Back Stack e por que ele é tão importante para as aplicações Android.

Objetivos

Ao final desta aula, você será capaz de:

- Criar uma aplicação na qual diversas Activities se comunicam;
- Inicializar recursos padrões do sistema a partir de sua aplicação;
- Entender o funcionamento da comunicação de Activities e do Back Stack;

O que são Intents?

No Android, para fazer a comunicação entre componentes (Activities, Services e Broadcast Receivers) utilizamos um objeto chamado Intent. Esse objeto carrega dentro dele informações relacionadas à ação que deve ser executada, bem como dados relacionados à execução dessa aplicação. A partir dessas informações, o sistema pode detectar quem deve assumir um Intent que tenha sido enviado e inicializar esse componente para tomar as ações necessárias. Mesmo componentes que façam parte do sistema, como o gerenciador de mensagens ou chamadas, podem ser inicializados através de Intents específicos, que podem ser gerados por qualquer que seja a aplicação.

Apesar do objeto Intent ser o mesmo para todos os componentes que o utilizam, as mensagens enviadas possuem descritores do tipo de componente que aquele Intent busca, passando apenas pelos componentes daquele tipo, em busca de tratamento. Um Intent criado e lançado por um comando *startActivity(Intent)* nunca vai chegar a um Broadcast Receiver nem a um Service, assim como um Intent lançado por um *startService(Intent)* nunca vai chegar a uma Activity ou a um Broadcast Receiver e assim por diante. Isso evita que o sistema perca tempo testando Intents contra componentes que não deveriam recebê-los, além de evitar um componente receber um Intent que não deveria ser dele, pois dois componentes diferentes podem ter o mesmo filtro, diferenciado apenas pelo tipo do componente.

O que há Dentro de um Intent?

O Intent, por ser um objeto compartilhado por diversos componentes do sistema, possui uma série de elementos padrões dentro dele, os quais indicam ao sistema o que fazer com cada um desses Intents. Esses elementos também indicam aos componentes que recebem o Intent informações dos dados a serem tratados. Vamos ver então quais são os campos que devem ser preenchidos nos Intents para que funcionem corretamente.

O primeiro campo de um Intent que iremos estudar é o campo **action**, que normalmente é configurado já na inicialização do Intent, e é o campo responsável por dizer ao Intent qual é o componente que ele deverá inicializar. É possível inicializar o componente através de uma String indicando o caminho até aquele componente, ou através de uma ação padrão do Android.

Existe várias Actions padrões no Android. Dentre elas, podemos citar alguns exemplos:

- ACTION_VIEW - iniciar uma nova Activity
- ACTION_CALL - fazer uma chamada telefônica
- ACTION_SEND - enviar informações, como uma mensagem de texto por exemplo
- ACTION_WEB_SEARCH - realizar uma pesquisa web

Outro campo de igual importância aos Intents é o campo *data*. Esse campo passa os dados necessários a uma Intent. Um exemplo disso seria criar um link para enviar e-mails no perfil de uma pessoa em uma aplicação de uma rede social. Quando o usuário tocar nesse link, o aplicativo no dispositivo responsável pelos e-mails será inicializado, sendo preenchido o campo e-mail com o e-mail do usuário passado no campo data. Vejamos, na **Listagem 1**, um exemplo com valores que podemos utilizar em um Intent de nossa aplicação.

```
1 Intent intent = new Intent(Intent, ACTION_DIAL);
2 intent.setData (Uri.parse("tel:12345678"));
3 startActivity(intent);
```

Listagem 1 - Intent para fazer uma chamada telefônica

Um Intent criado utilizando a ação *ACTION_DIAL* e os dados *tel:12345678*, será lançado para a aplicação responsável pelas chamadas do aparelho, já preenchendo o número a ser discado com o número passado nos dados, no caso 12345678. Esse é um exemplo de como qualquer aplicação pode lançar serviços padrões do sistema, como o serviço de chamadas.

Outro campo existente é o *category*. Como veremos adiante, quando estudarmos os Intent Filters, os componentes, quando definidos, definem também uma categoria ao qual eles vão pertencer. Essas categorias fornecem informações

extras em relação ao componente que deve responder ao Intent. Um exemplo disso seria utilizar a categoria CATEGORY_PREFERENCE. Essa categoria indica que a Activity alvo é um painel de preferências. Existem diversas outras categorias de Intent. Essas categorias podem ser vistas no autocompletar do Android Studio, em um campo relacionado, ou na página do Android Developers, citada nas referências.

Na sequência, temos o campo *component*. Esse campo é o responsável por indicar qual o nome do componente que o Intent é destinado. Funciona como um "endereço", indicando exatamente qual o componente que irá receber o Intent. Normalmente, os Intents são entregues ao componente que tem os requisitos descritos nos campos action, data e category, e não diretamente endereçados. Mesmo assim, é importante saber que é possível direcionar o chamado a um componente específico através desse campo.

Por fim, o último campo que compõe um Intent é o campo extras. Esse é o campo responsável por carregar valores extras que devem ser passados junto ao Intent. As informações são colocadas dentro de um Bundle e enviadas. Para exemplificar, imagine que temos uma lista de mensagens e, quando clicamos em alguma delas, abrimos uma Activity para exibir e editar essa mensagem. Para que essa segunda Activity possa exibir a mensagem que havia sido selecionada, ela deve receber as informações da mensagem, como título, corpo, etc., sendo passadas dentro dos extras do Intent.

Agora que já conhecemos como é composto um Intent, podemos passar a estudar os dois tipos de Intent que existem.

Intents Implícitos e Explícitos

Os Intents podem ser divididos em duas categorias, os Intents Implícitos e os Intents Explícitos. São chamados de Intent Implícitos aqueles que não possuem o campo component configurado, ou seja, não possuem um endereço exato. Com isso, esses Intents vão ser respondidos por um componente qualquer que se encaixe nos campos que foram configurados, normalmente action, data e category, como citado anteriormente. Esse tipo de Intent é mais utilizado quando o Intent é destinado a uma aplicação genérica, ou a um componente que não faz parte da aplicação.

No caso dos Intents Implícitos, o Android deve testar o Intent contra diferentes componentes até encontrar o que mais se adéqua a responder aquela chamada. Esses testes são feitos utilizando os campos citados acima contra os valores declarados nos Intent Filters de cada componente. Mais adiante, estudaremos os Intent Filters mais detalhadamente. Uma vez encontrado um componente que seja capaz de responder, o Android passa o Intent a esse componente, ou, caso sejam encontrados mais de um componente, o Android deixa que o usuário escolha qual componente deve responder a aquele Intent. Um exemplo que podemos citar, e que os usuários de dispositivos Android certamente estarão familiarizados, é quando desejamos compartilhar/enviar uma foto e o sistema irá mostrar uma lista com aplicativos que podem fazer isso: seja o aplicativo de e-mail, seja uma mensagem multimídia, seja por alguma rede social, etc.

Outro detalhe importante é que nunca se deve utilizar um Intent implícito para inicializar um serviço, uma vez que não há exatamente controle sobre qual serviço será iniciado e o usuário também não terá conhecimento sobre qual serviço foi inicializado, uma vez que os Services são executados em background. Na verdade, em versões posteriores do Android (API 21 em diante) esse comportamento passou a gerar uma exceção, impossibilitando o seu uso.

Já no caso dos Intents Explícitos, é declarado explicitamente a qual componente aquele Intent se destina. Através do campo component, visto anteriormente, o sistema sabe exatamente a qual componente o Intent deve ser entregue. Normalmente, utilizamos esse tipo de Intent quando ele se destina a um componente que faz parte da mesma aplicação. Assim, é mais fácil dizer a qual componente ele se destina e as chances de algo sair errado durante a passagem do Intent são extremamente baixas. Um Intent, quando utilizado dessa maneira, faz com que o Android ignore qualquer outro campo na hora de entregá-lo ao componente responsável. Apenas o componente descrito é levado em conta na busca.

Agora que já sabemos como os Intents são passados, vamos ver como os componentes devem ser preparados para recebê-los.

Atividade 01

1. Cite os cinco principais campos de um Intent e dê um exemplo de valor que pode ser declarado em cada um deles, explicando-o.

Intent Filters

Os Intent Filters são a maneira que um componente tem de dizer ao Android quais são os Intents que ele é capaz de responder. Esses filtros definem quais os valores que o Android deve testar quando está procurando um componente para entregar um Intent implícito que tenha sido lançado por algum outro componente. É importante lembrar que os Intent Filters são apenas responsáveis por filtrar Intents que não tenham um alvo definido, ou seja, Intents que não sejam explícitos. Caso um Intent seja destinado a um componente específico, explicitado no campo component, o Intent será entregue a esse componente, não importa o que esteja definido nos filtros dele.

No caso de um Intent não ter um destino certo, os Intent Filters são a maneira que um componente de um aplicativo tem de informar ao sistema quais os tipos de ações ele é capaz de tratar. Um componente pode ter diversos filtros, indicando as suas diversas funções, ou até mesmo nenhum filtro, caso receba apenas Intents explícitos. Esses filtros são normalmente declarados junto à declaração do componente, no AndroidManifest. Através da declaração no XML, fica mais fácil para o Android testar os campos do Intent. A exceção a essa regra acontece quando criamos filtros para um Broadcast Receiver, mas veremos isso mais para o fim do curso, quando formos estudar esses componentes.

Um filtro declarado no AndroidManifest possui campos que se equivalem aos campos de um Intent, que são utilizados para a comparação entre o Intent e seu receptor. Ao declarar um filtro, deve-se declarar um ao mais dos seguintes campos: action, category e data. Um Intent Implícito, que vai ser testado contra esse componente, deve passar em todos os testes para ser entregue. Se pelo menos um

dos filtros falhar, no caso do componente utilizar mais de um filtro, o componente não será considerado capaz de responder a esse Intent e será descartado. Vejamos agora, em detalhes, como funciona o teste de cada um desses campos dos filtros:

- *Action:* Nesse primeiro teste, o importante é testar se a action descrita pelo Intent está entre as actions listadas pelo componente. Mesmo que um Intent tenha apenas uma action, um componente pode ter mais de uma action na sua lista de filtros, sendo capaz de responder a qualquer uma delas. Caso dentre essas actions listadas encontre-se a que o Intent procura, o componente passa nesse teste. Caso contrário, o componente é descartado imediatamente. Perceba que um componente que não tenha nenhuma action declarada entre seus filtros não irá receber Intents implícitos nunca, pois sempre falhará nesse teste. Até mesmo quando o Intent não declarar uma *action*, um componente sem filtros para esse campo será rejeitado como possível responsável.

- *Data:* No teste de *data*, o tipo dos dados que está sendo requerido no Intent é testado contra o tipo aceitável declarado no componente. Esse teste é realizado da mesma maneira que o anterior, necessitando o casamento de pelo menos um dos elementos, declarados pelo componente, com o elemento passado pelo Intent.

- *Category*: Nesse teste, a categoria que o Intent está requerendo é comparada à categoria que o componente é capaz de responder. Para um Intent ser bem sucedido num teste da category, o componente deve declarar todas as categorias pedidas pelo Intent. Se pelo menos uma não passar no teste, o Intent é rejeitado pelo componente. Caso o Intent não tenha nenhuma categoria, ele deveria passar nesse teste, porém, por restrição do Android, o filtro deve incluir a categoria android.intent.category.DEFAULT para ser capaz de receber Intents implícitos.

Agora que já vimos exemplos de todos os filtros que podem ser utilizados, vejamos, na **Listagem 2**, um trecho do código de um dos exemplos do Android, que demonstra alguns desses filtros implementados.

```xml
<activity
   android:name="NotesList"
   android:label="@string/title_notes_list">

   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
   <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <action android:name="android.intent.action.EDIT" />
      <action android:name="android.intent.action.PICK" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
   </intent-filter>
   <intent-filter>
      <action android:name="android.intent.action.GET_CONTENT" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
   </intent-filter>

</activity>
```

Listagem 2 - Trecho do AndroidManifest.xml do sample NotePad.

Nesse código, vemos uma Activity com três Intent Filters diferentes declarados para ela. Isso indica que essa Activity pode servir a mais de um propósito e é capaz de responder a mais de um tipo de chamado dos Intents. O primeiro Intent Filter declara que essa é a Activity principal da aplicação e que deve ser demonstrada no Launcher. Essa é outra função que os Intent Filters têm em uma aplicação. O Android, para montar a tela inicial de uma aplicação corretamente, utiliza-se de Intent Filters para descobrir qual Activity deve ser mostrada ao inicializar um aplicativo. Para entender o significado exato de cada constante utilizada nos campos dos filtros, consulte a referência ao Android Developers.

Agora que entendemos bem como funcionam os Intents, que são os principais responsáveis pela comunicação entre componentes diversos, vamos estudar um pouco como funciona essa comunicação entre Activities.

Atividade 02

---

1. Descreva, em poucas palavras, a importância dos Intent Filters para Intents implícitos e explícitos.

2. Cite os três campos de comparação que um Intent Filter pode conter.

Comunicação entre Activities

A comunicação entre Activities é uma parte bastante importante em qualquer aplicação Android. É através dessa comunicação que é possível criar aplicações com telas diversas, capazes de realizar tarefas com grande precisão. Para iniciar uma nova Activity, utilizamos o comando *startActivity(),* passando como parâmetro um Intent que descreva a Activity a ser inicializada.

Frequentemente estaremos utilizando esse método para inicializar novas Activities dentro do contexto da nossa aplicação. Sendo assim, a maneira mais fácil e eficiente de se iniciar essas Activities é através de Intents explícitos, indicando exatamente a qual classe aquele Intent se destina. Uma vez criada uma nova Activity e declarada no Manifest, basta dizer que a classe a ser inicializada por Intent, deve ser a que foi previamente declarada.

Vejamos na **Listagem 3** como criar um Intent destinado a uma Activity da nossa aplicação.

```
1 Intent intent = new Intent(this, HelloActivity.class);
2 startActivity(intent);
```

Listagem 3 - Inicializando uma Activity explicitamente.

Na **Listagem 3**, vemos o exemplo da criação de um Intent responsável por inicializar a Activity HelloActivity, que já deve ter sido criada e declarada no Manifest. Esse construtor de Intents recebe dois parâmetros: o contexto e o componente alvo. O contexto da aplicação pode ser passado com o parâmetro “this” caso este represente a Activity atual. Já o component deve ser preenchido com uma referência à classe que desejamos acessar. Utilizando essa forma de trabalhar com Intents, é possível criar um fluxo de telas entre as Activities do seu programa. Para ver o funcionamento disso, vamos criar essa nova Activity dentro do projeto da tela de login, utilizado na aula anterior. Essa nova Activity pode ser a padrão, apenas com o texto de “Hello Activity”. Nomeie essa Activity HelloActivity e declare-a no Manifest. A **Listagem 4** mostra a alteração que deve ser feita no código da Activity principal, criada na aula anterior, para que possamos iniciar a Activity HelloActivity ao realizarmos o login.

```
package br.ufrn.imd.aula05f;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

public class Aula05Activity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button confirmar = (Button) findViewById (R.id.confirmar);
        Button limpar = (Button) findViewById (R.id.limpar);

        final EditText login = (EditText) findViewById(R.id.login_text);
        final EditText senha = (EditText) findViewById(R.id.pass_text);
        final ImageView imgView = (ImageView) findViewById(R.id.imageView1);

        confirmar.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                String loginStr = login.getText().toString();
                String senhaStr = senha.getText().toString();
                if (loginStr != null && loginStr.equals("aluno")) {
                    if (senhaStr != null && senhaStr.equals("aluno")) {
                        // Sucesso
                        Intent intent = new Intent(getApplicationContext(), HelloActivity.class);
                        startActivity(intent);
                    } else {
                        //Falha senha
                        Toast.makeText(getApplicationContext(), "Senha inválida", Toast.LENGTH_SHORT).sho
                    }
                } else {
                    //Falha login
                    Toast.makeText(getApplicationContext(), "Login inválido", Toast.LENGTH_LONG).show();
                }
            }
        });

        limpar.setOnClickListener(new View.OnClickListener() {
            @Override

            public void onClick(View v) {
                login.setText("");
```

```
52              senha.setText("");
53              imgView.setImageResource(android.R.drawable.presence_offline);
54          }
55      });
56  }
57 }
58
```

Listagem 4 - Modificando o código da aula anterior para iniciar uma nova Activity.

Agora, ao invés de modificar um ImageView no caso de um login bem sucedido, iniciamos uma nova Activity, que, após algum desenvolvimento, pode ser a tela inicial para aquele usuário que acaba de logar, com as informações que a aplicação se propõe a fornecer. O que é realmente importante nesse exemplo é entender como é simples criar uma nova Activity utilizando um Intent e o comando startActivity(), passando esse Intent.

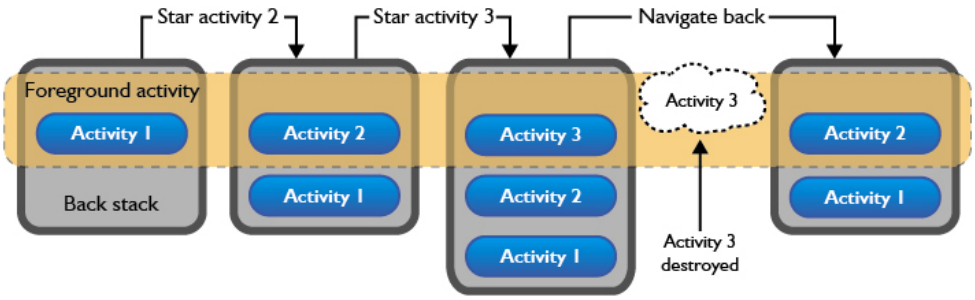
Agora uma pergunta: o que será que acontece uma vez que apertarmos a tecla voltar do dispositivo, tendo essa Activity nova à frente? Se você tiver executando o exemplo e tiver tido essa curiosidade, verá que a tela se fechou e voltamos a nossa tela de login, do jeito que a havíamos deixado. Esse é um bom momento para falarmos um pouco sobre o Back Stack.

Vídeo 02 - Invocando Activities

Back Stack

O Back Stack, ou, numa tradução literal, pilha de volta, é uma pilha fictícia no qual o Android armazena Activities a medida que estas são lançadas e tomam a frente de uma aplicação. Cada aplicação tem seu próprio Back Stack, em que suas Activities são empilhadas, gerando o senso de navegação por parte do usuário. A **Figura 1** mostra como funcionaria o Back Stack de uma aplicação com três Activities, uma lançada após a outra.

Figura 01 - Back Stack de uma aplicação.



Fonte: http://developer.android.com/index.html. Acesso em: 04 dez. 2015

Como vemos na **Figura 1**, à medida que novas Activities são lançadas dentro de uma aplicação, elas vão tomando o foco e sendo colocadas no topo da pilha. Uma vez destruída, normalmente através de um comando de voltar gerado pelo usuário, a Activity é encerrada e deixa o topo da pilha, passando o foco para a Activity que estava na segunda posição e agora está no topo. Existem vários detalhes que podem modificar o componente do Back Stack e é possível, até mesmo, a criação de novos Back Stacks para que uma Activity se torne a nova base de uma aplicação. Esses detalhes, entretanto, são um conteúdo bastante avançado e fica como curiosidade buscar mais informações sobre eles. Uma dica: o Android Developers tem um bom texto sobre isso!

Inicializando Activities que não são Parte da Aplicação

Outro ponto importante na navegação entre Activities é explorar a utilização de Activities que não são parte de sua aplicação, mas são padrão no sistema Android. Um exemplo seria utilizar a Activity de e-mails para enviar um e-mail a partir de sua aplicação. Agora, veremos como isso pode ser feito.

Da mesma maneira que necessitamos criar um Intent para navegar entre Activities de nossa aplicação, precisamos criar um para inicializar Activities que sejam padrão do Android, mas que não fazem parte diretamente de sua aplicação. Vejamos na **Listagem 5** como seria a implementação do exemplo de e-mail que citamos.

```
1 Intent intent = new Intent(Intent,ACTION_SEND);
2 startActivity(intent);
```

Listagem 5 – Mandando um e-mail a partir de sua aplicação.

O exemplo da **Listagem 5** mostra a criação de um Intent utilizando o Intent padrão do Android (lembre-se de utilizar o recurso autocompletar do Android Studio para conhecer mais sobre essas constantes) voltado ao envio de e-mails. Após definir a ação a ser executada, basta lançar o Intent através do startActivity(), como é feito com qualquer outra Activity, e o Android se encarregará de encontrar a aplicação de e-mail e iniciá-la para responder ao seu chamado. Agora que já sabemos como iniciar novas Activities dentro de nossa aplicação, o próximo passo é conseguir extrair dessas Activities alguma informação, quando necessário.

Vídeo 03 - Invocando Activities pt.2

Passando Dados entre Activities

É de grande importância para uma aplicação Android ser capaz de transferir dados entre suas Activities. Esses dados podem conter informações de usuário logado, de dados que são importantes para a criação da nova Activity ou mesmo dados que foram gerados numa Activity filha e estão sendo retornados ao pai. Vejamos na **Listagem 6** como adicionar dados a um Intent, passando o texto digitado no campo *login* para a próxima tela.

```
package br.ufrn.imd.aula05f;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

public class Aula05Activity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button confirmar = (Button) findViewById (R.id.confirmar);
        Button limpar = (Button) findViewById (R.id.limpar);

        final EditText login = (EditText) findViewById(R.id.login_text);
        final EditText senha = (EditText) findViewById(R.id.pass_text);
        final ImageView imgView = (ImageView) findViewById(R.id.imageView1);

        confirmar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String loginStr = login.getText().toString();
                String senhaStr = senha.getText().toString();
                if (loginStr != null && loginStr.equals("aluno")) {
                    if (senhaStr != null && senhaStr.equals("aluno")) {
                        // Sucesso
                        imgView.setImageResource(android.R.drawable.presence_online);

                        Intent intent = new Intent(getApplicationContext(), HelloActivity.class);
                        intent.putExtra("login", loginStr);
                        startActivity(intent);

                    } else {
                        // Falha senha
                        Toast.makeText(getApplicationContext(), "Senha inválida", Toast.LENGTH_SHORT).sho
                    }
                } else {
                    // Falha login
                    Toast.makeText(getApplicationContext(), "login inválido", Toast.LENGTH_LONG).show();
                }
            }
        });

        limpar.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
52
53          public void onClick(View v) {
54             login.setText("");
55             senha.setText("");
56             imgView.setImageResource(android.R.drawable.presence_offline);
57          }
58       });
59    }
60 }
```

Listagem 6 – Colocando dados em um Intent.

Como estudamos na primeira parte dessa aula, os Intents possuem um campo chamado extra, cuja funcionalidade é justamente armazenar informações extras. É através desse campo que passamos dados entre Activities. Para colocar esses dados, precisamos identificá-los com uma String, para que possam ser recuperados do outro lado da comunicação, e então colocarmos os dados. No exemplo, identificamos o campo com a String “login”, e o dado a ser passado é o valor contido na variável loginStr. O Android aceita diversos tipos de dados dentro dos Intents, como inteiros, shorts, longs e até mesmo serializables em geral. Colocando esses valores dentro de um Intent com Strings de identificação diferentes, é possível passar adiante várias informações. A **Listagem 7** mostra como recebemos essa informação na HelloActivity iniciada.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/RelativeLayout1"
5     android:layout_width="fill_parent"
6     android:layout_height="fill_parent"
7     android:background="#007">
8
9     <TextView
10       android:id="@+id/textView_login"
11       android:layout_width="wrap_content"
12       android:layout_height="wrap_content"
13       android:layout_alignParentTop="true"
14       android:layout_centerHorizontal="true"
15       android:layout_marginTop="15dip"
16       android:gravity="center"
17       android:text="Olá!"
18       android:textSize="20sp"
19       android:textStyle="bold"
20       android:textColor="#FFF" />
21
22 </RelativeLayout>
```

Listagem 7a - Arquivo activity_hello.xml, representando o XML do Layout da HelloActivity.

```
1  package br.ufrn.imd.aula05f;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.TextView;
6
7  public class HelloActivity extends Activity {
8     protected void onCreate(Bundle savedInstanceState) {
9        super.onCreate(savedInstanceState);
10       setContentView(R.layout.activity_hello);
11
12       String dados = getIntent().getStringExtra("login");
13
14       TextView texto = (TextView) findViewById(R.id.textView_login);
15       texto.setText("Olá " + dados);
16    }
17 }
```

Listagem 7b – Recebendo dados de um Intent.

Como vemos na **Listagem 7**, o método getIntent() é o responsável por nos retornar o Intent que chegou até essa Activity. O método getStringExtra(), chamado nesse Intent, se encarrega de extrair uma String que tenha sido colocada entre os extras. Esse método recebe como parâmetro o identificador usado anteriormente para o dado gravado nos extras. O método procura um dado do tipo String com o identificador “login” existente nos extras da Intent. No nosso exemplo, o dado encontrado é armazenado na variável dados, e depois é atribuído para exibição ao TextView da HelloActivity, através do ID que definimos para ele. Tenha cuidado ao copiar os exemplos se você está de fato utilizando os mesmos nomes para as variáveis! Nesse exemplo obtemos um extra do tipo String, mas podemos também obter outros tipos de dados através de métodos como getIntExtra(), getFloatExtra(), getBooleanExtra(), getSerializableExtra(), etc.

Também podemos retornar informações para uma Activity anterior, como veremos a seguir.

Vídeo 04 - Activity para Share

Iniciando uma Activity para um Resultado

Imagine que, além de iniciar uma nova Activity, seu programa queira receber alguma informação processada por essa Activity. Por exemplo, imagine que você tenha uma lista de nomes cadastrados e queira editar algum desses nomes. Imagine que para fazer isso, você vai tocar no nome e uma nova Activity será exibida para editar esse texto. Uma vez concluída essa edição, essas informações devem ser salvas e exibidas pela Activity inicial.

Para fazer isso, precisamos receber da Activity lançada informações sobre os dados que foram manipulados nela. É aí que entra o comando startActivityForResult(). Ao iniciar uma Activity utilizando esse comando, espera-se que a nova Activity, ao finalizar, retorne para a Activity anterior alguma informação que vai ser útil a ela. A **Listagem 8** mostra um exemplo de uma Activity inicializando outra a fim de obter um resultado dela.

```
1 Intent intent = new Intent(getApplicationContext(), HelloActivity.class);
2 startActivityForResult(intent, 1);
```

Listagem 8 – Iniciando uma Activity para um Resultado.

Uma vez iniciada utilizando esse comando, uma Activity deve responder com um resultado, utilizando um resultCode, configurado através do método setResult(), e um Intent, criado durante a execução da Activity. Esses dois valores são os responsáveis por retornar os resultados requeridos pela Activity inicial. Assim que a Activity for finalizada, esses valores serão enviados ao método onActivityResult() da Activity pai. A **Listagem 9** demonstra uma possível implementação desse método.

```
@Override
   protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
     super.onActivityResult(requestCode, resultCode, intent);
     if (requestCode == 1 && resultCode != 0) {
        String dados = (String) intent.getStringExtra("dados");

        switch(resultCode) {

            case 1:
               // sucesso(dados);
               break;
            case 2:
               //erro();
               break;
            case 3:
               // naoAlterado();
               break;
            }
     }
}
```

Listagem 9 – Método onActivityResult.

Veja que esse método tem três parâmetros. O primeiro deles, o requestCode, é o parâmetro responsável por dizer a Activity qual foi o código que ela enviou na sua criação. O valor deste parâmetro deve ser o mesmo indicado no segundo parâmetro na Activity pai, na chamada do método startActivityForResult(intent,1) (no caso, o valor 1). Esse parâmetro é necessário, pois é possível iniciar o tratamento do resultado por mais de uma Activity, e ele ajuda a saber qual Activity filha está retornando para esse método. O parâmetro resultCode é utilizado pela Activity filha para sinalizar qual foi o resultado da execução. É através dessa variável que a Activity filha poderá indicar se teve sucesso ou não em realizar o que lhe foi pedido, por exemplo. Por fim, o Intent que é devolvido a esse método é utilizado para transportar dados entre as duas Activities, tornando possível ao pai utilizar dados que foram gerados no filho para se atualizar, ou completar os seus objetivos. Essa transferência de dados entre Activities é outro ponto muito importante na comunicação destas.

Para retornar essas informações em um Intent, quando a Activity for iniciada por um resultado, o procedimento é bem parecido como o visto anteriormente para passar um extra de uma Activity para outra. A única diferença é que o Intent não será pego através do getIntent() na Activity pai e sim será passado como parâmetro para o método onActivityResult().

Com esses conhecimentos sobre como utilizar os Intents para passar valores entre Activities, encerramos nossa segunda aula sobre Activities, que falou sobre a comunicação entre esses componentes. Até a próxima!

Atividade 03

---

1. Qual a diferença entre os métodos startActivity() e startActivityForResult()? No caso do método startActivityForResult(), qual o método responsável por obter o resultado requerido?

Leitura Complementar

INTENTS ACTIONS. Disponível em: <http://developer.android.com/reference/android/content/Intent.html>. Acesso em: 04 dez. 2015.

INTENTS and Intent Filters. Disponível em: <http://developer.android.com/guide/topics/intents/intents-filters.html>. Acesso em: 04 dez. 2012.

Resumo

Na aula de hoje, vimos o que é um Intent e como ele funciona internamente. Vimos ainda a divisão desses em Intents implícitos e explícitos. Também vimos como o Android é capaz de passar esse Intent para diferentes componentes e como os componentes respondem se são capazes de receber esses Intents através da utilização de Intent Filters. Por fim, estudamos como funciona a comunicação entre Activities, utilizando Intents para popular o Back Stack de uma aplicação e criar uma sequência entre as telas. Vimos também como obter resultados de uma Activity filha que tenha sido iniciada com esse fim e como passar dados entre essas Activities. Essa é outra aula de importância fundamental para qualquer aplicação que venha a ser desenvolvida.

Autoavaliação

1. Qual a diferença de um Intent explícito para um implícito? Quando cada um deles é mais utilizado?
2. O que acontece com um componente que não tem nenhum Intent Filter? E caso o Intent Filter não possua nenhuma action listada?
3. Qual o método utilizado para adicionar dados a um Intent? Em qual campo esses dados são adicionados?

4. É possível iniciar um componente padrão do sistema utilizando Intents? Como isso deve ser feito?

Referências

ANDROID Developers. 2015. Disponível em: <http://developer.android.com>. Acesso em: 12 mai. 2015.

DIMARZIO, J. **Android**: A Programmer's Guide. *New York:* McGraw-Hill, 2008. ISBN 9780071599887. Disponível em: <http://books.google.com.br/books?id=hoFI5pxjGesC>. Acesso em: 3 ago. 2012.

LECHETA, Ricardo R. **Google Android**: aprenda a criar aplicaçoes. 2. ed. São Paulo: Novatec, 2010.

_____. **Google android para tablets**. São Paulo: Novatec, 2012.

MEIER, R. **Professional Android 2 Application Development**. New Jersey: John Wiley & Sons, 2010. ISBN 9780470874516. Disponível em: <http://books.google.com.br/books?id=ZthJlG4o-2wC>. Acesso em: 3 ago. 2012.