Minicurso Spring e Android

Criado por Gabriel Schaidhauer - Aula 5

Activities

Activity

Activity é a principal classe dentro de um aplicativo Android. É em uma activity que são definidos diversos comportamentos de tela, realizadas solicitações de permissão e criadas intenções para acessar recursos do sistema operacional ou navegar para outras telas.

Activity

Além da classe java, uma activity também é composta de um arquivo xml o qual define como será a tela correspondente a uma atividade.

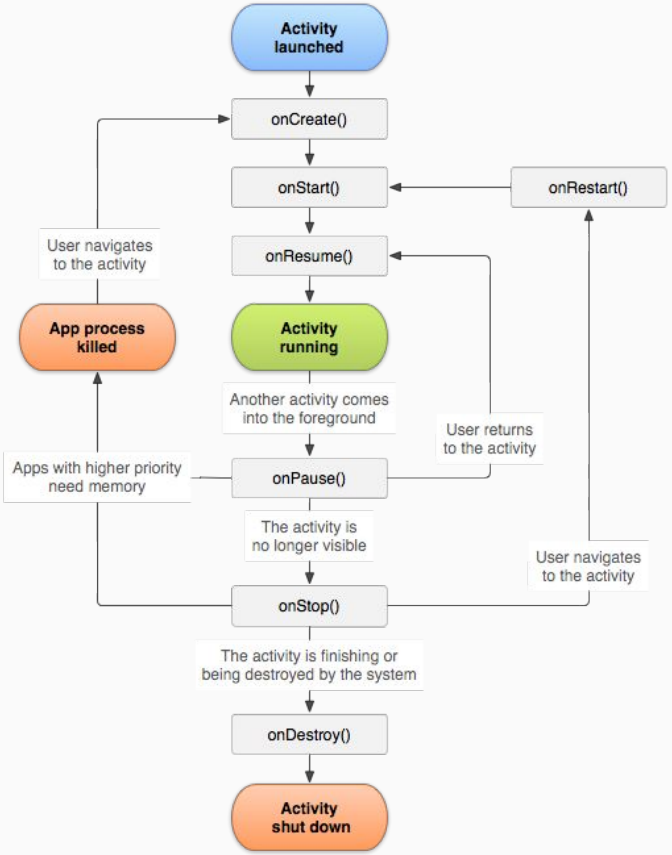
Clique AQUI Para entender melhor, como estilizar o seu App.

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.gabriel.aula5.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Activity

Uma activity possui um ciclo de vida fixo, o qual é gerenciado pelo sistema operacional. É importante que entendamos este ciclo de vida, uma vez que o framework android nos fornece métodos da classe activity, onde podemos realizar interações no momento em que certo ponto do ciclo de vida é alcançado.

Ciclo de vida de uma Activity



onCreate

Ao ser criada uma activity, é imediatamente invocado o método onCreate, o qual deve obrigatoriamente ser implementado pela activity, podendo agir como se fosse um construtor para o objeto.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.d( tag: "teste",  msg: "onCreate");
}
```

onStart

Este método é invocado no momento em que a activity passa a ser exibida ao usuário, caso seja utilizado deve ser aqui que serão inicializados BroadcastReceivers.

```
@Override
protected void onStart() {
    super.onStart();
    Log.d( tag: "teste",  msg: "onStart");
}
```

onPause

Este método é chamado quando o aplicativo perde o foco, pausando a execução da activity. sendo este um bom local para liberar alguns recursos que só precisem estar em execução quando o app está focado.

```
@Override
protected void onPause() {
    super.onPause();
    Log.d( tag: "teste",  msg: "onpause");
}
```

onResume

Este método é chamado quando uma Activity volta a ter foco após ter perdido o foco e ter sido invocado o método onPause. aqui podem ser solicitados novamente recursos que tenham sido liberados no onPause.

```
@Override
protected void onResume() {
    super.onResume();
    Log.d( tag: "teste",  msg: "onResume");
}
```

onStop

Este método é invocado no momento em que a activity deixa de ser vista pelo usuário. Devendo ser liberados recursos como Broadcast Receivers que interajam com a UI.

A partir deste método, podem surgir tanto onRestart quanto onDestroy

```
@Override
protected void onStop() {
    super.onStop();
    Log.d( tag: "teste",  msg: "onstop");
}
```

onRestart

Este método é invocado quando uma activity que havia sumido da tela (onStop) é novamente trazida para a frente. Neste momento não podemos ocupar os recursos que haviam sido liberados pois será invocado onStart.

onDestroy

Este método é chamado quando a activity é destruída, ou seja, ou aplicativo foi fechado, ou o sistema Android precisou liberar memória para outros recursos. Este é o seguimento de onStop caso não seja reaberta a activity.

Beleza, mas como faz? XML? Java?

Activity

Todos os elementos de tela declarados no arquivo XML da activity podem ser acessados na porção Java. Isto pode ser feito de diversas formas. A forma mais comum de acessar elementos de tela e aplicar comportamentos a estes é através de um identificador textual, o qual é utilizado para encontrar o elemento. Ainda, alguns elementos do XML possuem eventos como o onClick dos botões, onde é possível colocar o nome de um método da classe java, o qual será invocado ao clicar no botão.

Interagindo com Activity

Ao clicar no botão, o método 'changeTextClickHandler' é invocado, e dentro do método, é recuperada a instância de do texto exibido da tela, e seu conteúdo é alterado.

```
public void changeTextClickHandler(View view) {
    TextView helloText = findViewById(R.id.hello_text);
    helloText.setText("Botão foi clicado");
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.gabriel.aula5.MainActivity">

    <TextView
        android:id="@+id/hello_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/example_button"
        android:layout_centerInParent="true"
        android:onClick="changeTextClickHandler" />

</RelativeLayout>
```

Activities

Quando construímos um App, ele não possui somente uma tela, existem uma série de telas, e podemos ir de uma para outra, e assim por diante. No Android, cada uma destas telas pode ser uma activity, e para navegar entre uma e outra devemos solicitar ao sistema operacional que realize a navegação. Esta solicitação é uma intenção (Intent).

Intent

Intent

No desenvolvimento de aplicativos Android, quase tudo o que fazemos deve ser feito através de uma intent (intenção), a qual funciona como uma solicitação para para que o android realize uma a ação que queremos. Estas ações pode ser as mais variadas, como navegar para uma nova Activity ou abrir a câmera para ler um código de barras.

Navegando para outra tela

Para navegar para outra activity, criamos uma Intent passando como parâmetros a activity em que estamos e a activity para a qual queremos ir. usamos a Intent também para adicionar a mensagem exibida na tela para a próxima activity, e por fim invocamos o método startActivity.

```
public void navigateToNext(View view) {
    TextView helloText = findViewById(R.id.hello_text);

    Intent intent = new Intent( packageContext: this, SecondActivity.class);
    intent.putExtra( name: "TextoHello", helloText.getText());
    startActivity(intent);
}
```

Intent

Ao instânciar uma nova activity, ela seguirá o lifecycle que vimos anteriormente, podendo captar dados da activity anterior, como a mensagem que passamos através da intent.

Fragment

Fragment

Um Fragment é muito similar a uma activity. Ele representa uma parte da tela do usuário, possui um ciclo de vida próprio. É como se um Fragment fosse um trecho de uma activity podendo integrar o todo da interface do usuário.

Activity x Fragment

- Activity pode existir de forma independente;
- Fluxo de vida da activity é independente;
- Pode conter um ou mais Fragments;

- Fragment existe dentro de uma activity;
- Fragment possui alguns pontos do fluxo de vida que dependem da activity;
- É contido por uma activity;

1. Fragments na UI

Uma das formas de adicionar um Fragment na UI do aplicativo é utilizando-o diretamente no XML do nosso app.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.gabriel.aula5.MyFragmentActivity">

    <fragment
        android:id="@+id/frag1"
        android:name="com.example.gabriel.aula5.TextFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/frag1"
        android:text="Criar Fragment"
        android:onClick="createFragment"/>

    <RelativeLayout
        android:layout_below="@id/btn"
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</RelativeLayout>
```

Fragments de forma programática

É possível inserir um fragment dentro de uma activity ou outro layout em uma activity utilizando um FragmentManager.

```
public void createFragment(View view) {
    FragmentManager fm = getFragmentManager();
    FragmentTransaction ft = fm.beginTransaction();
    android.app.Fragment f = new TextFragment();
    ft.add(R.id.fragment_container, f);
    ft.commit();
}
```

Dúvidas?