

# ANDROID

Ciclo de Vida de uma Activity

Intents



## Activity

- Mostra a tela para o usuário
- Classe que herda de Activity
- Possui um ciclo de vida definido
  - <https://developer.android.com/guide/components/activities/activity-lifecycle.html>
  - Métodos que são executados sempre que a Activity muda de estado
  - Podem ser sobreescritos pelo programador
  - Deve-se sempre chamar a implementação do método pai



# Activity

- Estados de uma Activity
  - **Criada** : quando ela é inicializada e configurada (ex, layout)
  - **Iniciada** : quando ela vai ser tornar visível
  - **Retomada** : visível
  - **Pausada** : parcialmente visível, ex, atrás de uma dialog
  - **Interrompida**: quando ela está escondida, não visível
  - **Destruída**: liberando os recursos



# Activity

- **onCreate ()**:
  - Chamado quando a Activity é criada
  - Responsável por carregar layouts XML e outras operações.
  - Executada somente uma vez no ciclo de vida
- **onStart ()**:
  - Chamado após **onCreate**, antes de se tornar visível
  - Chamada também quando a Activity estava no fundo (Parada) e volta a ser mostrada
- **onResume ()**:
  - Chamado após **onStart**, quando se torna visível
  - Sempre invocado quando a Activity volta a ter o foco

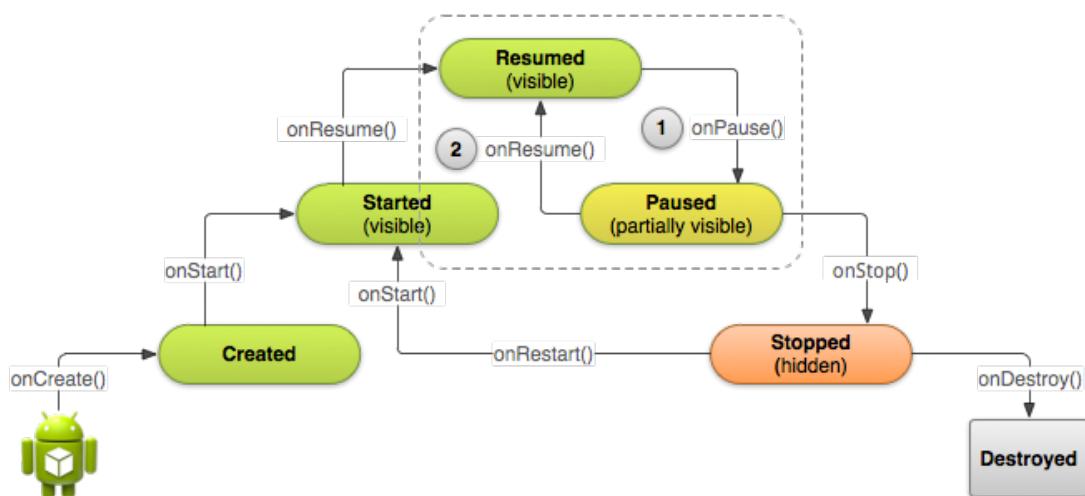


# Activity

- **onPause()**:
  - Primeiro método a ser invocado quando vai perder o foco (outra Activity na frente)
- **onStop()**:
  - Análogo a **onPause**, mas invocado quando a Activity fica completamente encoberta por outra (não é mais visível)
- **onRestart()**:
  - Chamado antes de **onStart()**, quando a Activity estava Parada e voltará a ter o foco
- **onDestroy()**:
  - Último método invocado, quando está prestes a ser destruída



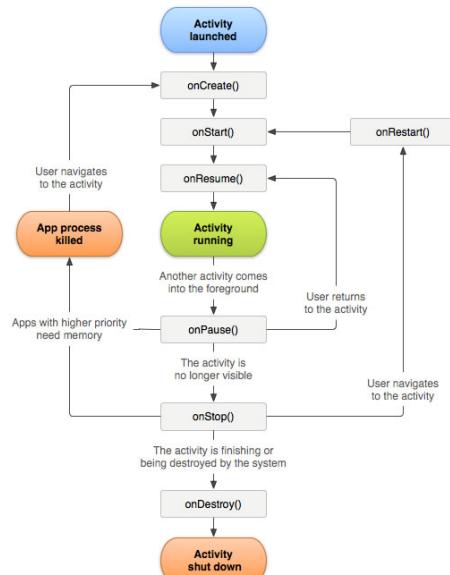
## Ciclo de Vida – Visão dos Estados



Fonte: android.developer.com



# Ciclo de Vida – Visão dos Métodos



Fonte: android.developer.com



# Navegação entre Activities

## Iniciar outra Activity

- Usa-se um objeto Intent para iniciar outra Activity
- Chamada:
  - `startActivity()` : inicia outra Activity e não recebe qualquer informação de retorno

```
Intent intent = new Intent(this, NovaActivity.class);
startActivity(intent);
```



## Passagem de Parâmetros entre Activities

- Passa-se via Intent
 

```
Intent intent = new Intent(this, NovaActivity.class);
intent.putExtra("nome", "Razer");
intent.putExtra("local", "UFPR");
intent.putExtra("idade", 22);
startActivity(intent);
```
- Para se obter os dados passados, faz-se no `onCreate()`

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.meu_layout);

    Bundle extras = getIntent().getExtras();
    String nome = extras.getString("nome");
    String local = extras.getString("local");
    int idade = extras.getInt("idade");
    ...
}
```



## Iniciar outra Activity – Com Retorno

- Algumas alterações devem ser implementadas
  - Usa-se um objeto Intent para iniciar outra Activity
  - Chama-se o método `startActivityForResult()`, com um código de requisição
  - A segunda Activity deve gerar o resultado ou cancelá-lo
  - A primeira Activity deve implementar `onActivityResult()` para obter o resultado
- Chamada na PrimeiraActivity:
  - `startActivityForResult()` : inicia outra Activity, mas deve-se receber um retorno desta

```
Intent intent = new Intent(this, SegundaActivity.class);
startActivityForResult(intent, 1);
```

- Onde o segundo parâmetro é um ID, que será usado para obter o resultado



## Iniciar outra Activity – Com Retorno

- A SegundaActivity deve gerar um resultado ou cancelar o resultado
- Para Gerar um Resultado:
 

```
Intent returnIntent = getIntent();
returnIntent.putExtra("resultado", oResultadoDesejado);
setResult(Activity.RESULT_OK, returnIntent);
finish();
```
- Para Cancelar o Resultado (não retornar nada):
 

```
Intent returnIntent = getIntent();
setResult(Activity.RESULT_CANCELED, returnIntent);
finish();
```



## Iniciar outra Activity – Com Retorno

- A PrimeiraActivity deve obter o resultado gerado na SegundaActivity
- Sobrescreve o método `onActivityResult()`

```
protected void onActivityResult(int requestCode, int resultCode,
                               Intent data) {
    if (requestCode == 1) {
        if(resultCode == Activity.RESULT_OK) {
            String resultado = data.getStringExtra("resultado");
        }
        if (resultCode == Activity.RESULT_CANCELED) {
            // Código se não tiver resultado
        }
    }
}
```



## Iniciar Activities de Outros Aplicativos

- Invocar telas de outros aplicativos
  - Câmera



## Tirando Fotos

- Permissão no aplicativo para usar a câmera no manifesto

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera"
                  android:required="true" />
    ...
</manifest>
```



## Tirando Fotos

- Usa-se um Intent

```
Intent fotoIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
if (fotoIntent.resolveActivity(getApplicationContext()) != null) {
    startActivityForResult(fotoIntent, 1);
}
```

- Onde:

- **resolveActivity()** : retorna a primeira activity que pode tratar esse Intent. Se não houver nenhuma e esta condição não for usada, a app para
- **getPackageManager()** : retorna um **PackageManager**, que contém informações sobre pacotes instalado no dispositivo



## Tirando Fotos

- Para obter a miniatura da foto, implementa-se `onActivityResult()`

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 1 && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        img.setImageBitmap(imageBitmap);
    }
}
```

- Onde:

- A minitura é retornada em `extras` com o nome "data" do tipo `Bitmap`
- O código acima coloca a minitura em um `ImageView`



## Tirando Fotos – Salvar a Foto Completa

- Deve-se ter permissão para leitura/escrita no armazenamento

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
■ Antes de invocar a câmera, cria-se um arquivo temporário da imagem, e passa-se como
parâmetro
File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
File imageFile = File.createTempFile("image1.jpg", storageDir);
Uri photoURI = FileProvider.getUriForFile(this,
        "com.razer.android.fileprovider", imageFile);
Intent fotoIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
fotoIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
if (fotoIntent.resolveActivity(getApplicationContext()) != null) {
    startActivityForResult(fotoIntent, 1);
}
```



## Tirando Fotos – Salvar a Foto Completa

- Configurar um FileProvider no manifesto aplicação

```
<application>
    ...
    <provider
        android:name="android.support.v4.content.FileProvider"
        android:authorities="com.razer.android.fileprovider"
        android:exported="false"
        android:grantUriPermissions="true">
        <meta-data
            android:name="android.support.FILE_PROVIDER_PATHS"
            android:resource="@xml/file_paths">
        </meta-data>
    </provider>
    ...
</application>
```



## Tirando Fotos – Salvar a Foto Completa

- Cria-se um recurso XML para assinalar o path, em `res/xml/file_paths.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path name="my_images"
        path="Android/data/com.razer.package.name/files/Pictures" />
</paths>
```

- O valor no atributo path é retornado pelo método `getExternalFilesDir()`, feito na chamada da Intent:

```
File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
```



## Tirando Fotos – Adicionar à Galeria

- Deve-se invocar o scanner de mídia
  - Deve-se ter o caminho absoluto da imagem

```
String pathAbsoluto = imageFile.getAbsolutePath();
Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
File f = new File(pathAbsoluto);
Uri contentUri = Uri.fromFile(f);
mediaScanIntent.setData(contentUri);
this.sendBroadcast(mediaScanIntent);
```

