

Unidade 1

Seção 3



Desenvolvimento Para Dispositivos Móveis

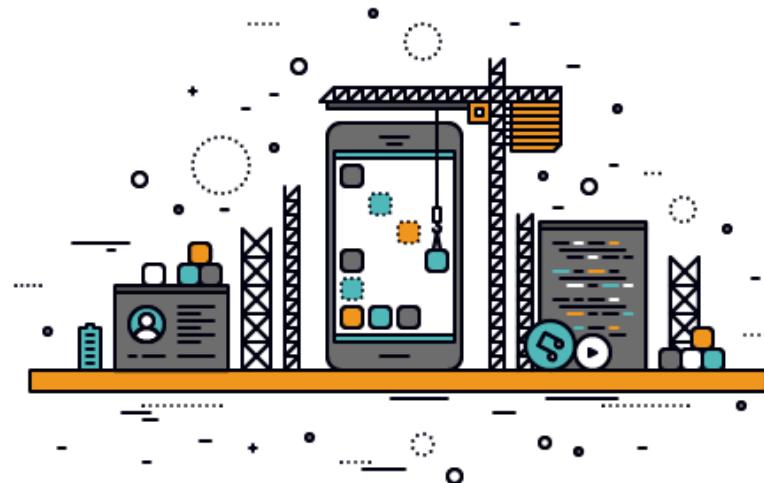




Weaula 3

Trabalhando Com *Views em Activities*

Nesta webaula nós estudaremos como acessar os elementos que estão disponíveis no *layout criado*. A partir da interação entre o usuário e esses elementos, o aplicativo apresentará comportamentos programados por nós. Todos os aplicativos que utilizamos, sejam eles aplicativos para trocas de mensagens, redes sociais, e-mail, agenda, câmera ou contatos de telefone, executam algum comportamento ao interagirmos com os elementos que dispõe o *layout* do aplicativo.



Fonte: Istock

Para que possamos compreender como acessar um elemento em um recurso de *layout*, vamos analisar a seguinte etapa do processo de compilação. Ao testar, implantar, assinar e distribuir o seu aplicativo, é necessário realizar o processo de compilação. Esse processo irá compilar o código fonte com os recursos disponíveis no aplicativo e os empacotará em um arquivo com extensão “.apk”.

Durante o processo de compilação, o **SDK** executa a ferramenta **Android Asset Packaging Tool - aapt**. Essa ferramenta se encarrega de procurar por arquivos e referências.

AAPT
Ferramenta de Empacotamento de Recursos do Android



Quando o usuário inicia o aplicativo, uma *Activity* é exibida no dispositivo móvel. Para que esse processo aconteça, essa atividade executa diversos métodos. Explore a galeria a seguir e veja que o primeiro método executado é o *onCreate()*, que:



Inicialmente é declarado na classe *AppCompatActivity*, porém, devemos sobrepor-lo em cada *Activity* do nosso aplicativo para que possamos configurá-lo de acordo com a necessidade dele.



Agora que nossa *MainActivity* inicializa com uma interface definida por nós, devemos acessar os elementos dessa interface. Primeiramente definimos qual elemento queremos acessar e criamos uma instância desse objeto no método *onCreate()* da *MainActivity*.

Clique na imagem para expandi-la.

```
1. package com.exemplo.primeiroapp;
2.
3. import android.os.Bundle;
4. import android.support.v7.app.AppCompatActivity;
5. import android.widget.EditText;
6.
7. public class MainActivity extends AppCompatActivity {
8.
9.     private EditText mEditText;
10.
11.    @Override
12.    protected void onCreate(Bundle savedInstanceState) {
13.        super.onCreate(savedInstanceState);
14.        setContentView(R.layout.activity_main);
15.
16.        mEditText = (EditText) findViewById(R.id.editText);
17.    }
18. }
```

Fonte: Elaborado pelo autor

Linha 9: declara o objeto que queremos acessar.

Linha 16: por meio do método `findViewById(int id)` é possível inicializar o objeto que desejamos acessar na interface. Observe que também estamos utilizando a Classe R para fazer referência a um elemento dentro do recurso de *layout*.

Segundo Developer (2018), houve uma mudança de assinatura para o método `findViewById(int id)`, partindo da versão 26.0.0 do SDK. De acordo com a documentação, a nova assinatura é:

`mEditText = findViewById(R.id.editText);`

Onde o tipo de retorno é `View`. O cast conforme exibido na linha 16 do código acima. A utilização do método `findViewById(int id)` poderá ser simplificada conforme **linha de exemplo**.

Linha de exemplo

```
mEditText = findViewById(R.id.editText);
```

Existem diversas maneiras para definir o comportamento do elemento *Button* ao ser pressionado pelo usuário. Seguiremos um passo-a-passo a fim de identificar a ação de toque no botão:

Acesse o arquivo “activity_main.xml” e acrescente o atributo “*android:onClick=“confirmar”*” no elemento *Button*.

38. <**Button**
39. ***android:id="@+id/button"***
40. ***android:layout_width="match_parent"***
41. ***android:layout_height="wrap_content"***
42. ***android:onClick="confirmar"***
43. ***android:text="Confirmar curso" />***

Fonte: Elaborado pelo autor

Acesse a classe `MainActivity.class` e crie um método seguindo as características abaixo:

- Deve ser público.
- Não pode possuir retorno;
- Deve ser criado com o mesmo nome definido no atributo `onClick` do elemento `Button`;
- Deve receber como parâmetro um objeto do tipo `View`.

```
1. package com.exemplo.primeiroapp;
2.
3. import android.os.Bundle;
4. import android.support.v7.app.AppCompatActivity;
5. import android.view.View;
6. import android.widget.EditText;
7.
8. public class MainActivity extends AppCompatActivity {
9.
10.     private EditText mEditText;
11.
12.     @Override
13.     protected void onCreate(Bundle savedInstanceState) {
14.         super.onCreate(savedInstanceState);
15.         setContentView(R.layout.activity_main);
16.
17.         mEditText = findViewById(R.id.editText);
18.     }
19.
20.     public void confirmar(View view) {
21.         // Acrescente a lógica aqui
22.     }
23. }
```

Fonte: elaborado pelo autor.

Clique na imagem para expandi-la.

8

Toda a lógica desenvolvida dentro do método `confirmar(View view)` será executada quando o usuário pressionar o botão disponível na interface da **MainActivity**.

O objeto “***mEditText***” faz referência ao elemento ***EditText*** na interface, portanto agora podemos recuperar o texto digitado pelo usuário através desse objeto.

Na linha 22, declaramos uma variável local String que recebe o valor digitado no *EditText* do *layout*.

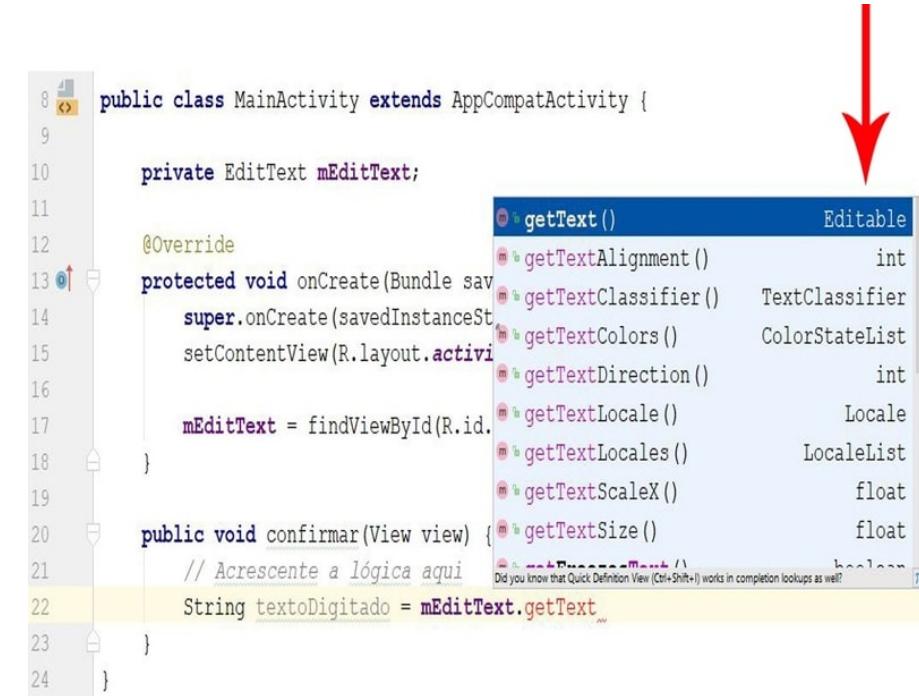
```
20. public void confirmar(View view) {  
21.     // Acrescente a lógica aqui  
22.     String textoDigitado =  
           mEditText.getText().toString();  
23. }
```

Fonte: elaborado pelo autor.

Vamos analisar o método `getText()` conforme figura.

Observe que ao digitar o nome do objeto `mEditText` e adicionar o símbolo “.” (ponto) para acessar os seus métodos é exibida uma caixa com todos os métodos disponíveis. Na coluna à direita, conforme destacado na figura, é exibido o tipo de retorno de cada método.

Clique na imagem para expandi-la.



```
8 public class MainActivity extends AppCompatActivity {
9
10    private EditText mEditText;
11
12    @Override
13    protected void onCreate(Bundle savedInstanceState) {
14        super.onCreate(savedInstanceState);
15        setContentView(R.layout.activity_main);
16
17        mEditText = findViewById(R.id.editText);
18    }
19
20    public void confirmar(View view) {
21        // Acrescente a lógica aqui
22        String textoDigitado = mEditText.getText();
23    }
24}
```

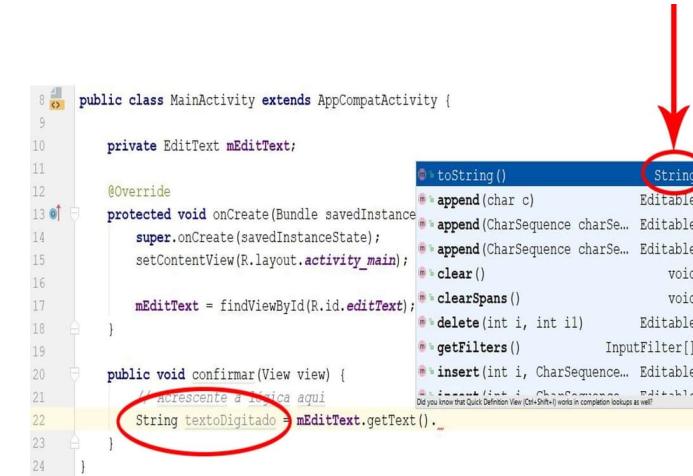
The screenshot shows the `MainActivity.java` file in an IDE. A red arrow points from the text above to the `mEditText.getText()` call at line 22. A tooltip for the `getText()` method is displayed, showing its return type as `Editable` and listing several other methods available on the `EditText` object.

| Método | Tipo de Retorno |
|----------------------------------|-----------------------------|
| <code>getText()</code> | <code>Editable</code> |
| <code>getTextAlignment()</code> | <code>int</code> |
| <code>getTextClassifier()</code> | <code>TextClassifier</code> |
| <code>getTextColors()</code> | <code>ColorStateList</code> |
| <code>getTextDirection()</code> | <code>int</code> |
| <code>getTextLocale()</code> | <code>Locale</code> |
| <code>getTextLocales()</code> | <code>LocaleList</code> |
| <code>getTextScaleX()</code> | <code>float</code> |
| <code>getTextSize()</code> | <code>float</code> |

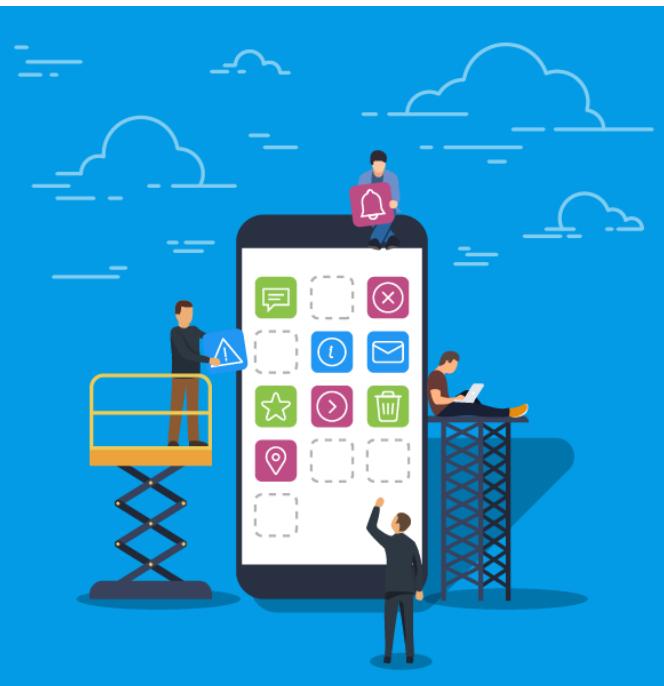
Fonte: elaborada pelo autor.

O objeto *EditText* possui o método *getText()* que permite recuperar o texto digitado e retorna um objeto do tipo *Editable*. Como desejamos trabalhar com uma *String*, devemos chamar ainda o método *toString()* para recuperar o texto digitado, conforme exibido na figura.

Clique na imagem para expandi-la.



Fonte: O autor



Fonte: iStock

Vimos durante esta webaula sobre como acessar os elementos que estão disponíveis no *layout* criado.

Vimos alguns exemplos para facilitar a compreensão dentre alguns outros elementos importantes para o trabalho com *Views* em *Activities*.

Vídeo de encerramento







Bons estudos!

