03

Melhorando a lista

Transcrição

A lista possui dois novos alunos, "Ronaldo" e "Daniel"! Para inseri-los fizemos alterações no `.xml`. Para seguir acrescentando nomes devemos retornar ao `.xml`. Mas, existe um caminho mais fácil, por exemplo, selecione com o mouse as linhas do `TextView` do "Ronaldo" até o fim:

```
<TextView android:text="Ronaldo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Para copiar e colar as linhas do código podemos seguir o caminho "Menu > Edit > Paste" ou utilizar os atalhos "Ctrl+C" e "Ctrl+V" (Windows) e "Comand C" e "Comand V" (Mac). Aproveitamos a linha repetida e apenas trocamos os nomes. Basta modificar "Ronaldo" para "Jeferson", por exemplo:

```
<TextView android:text="Jeferson"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Assim, "Jeferson" passa a ser o terceiro aluno!

Para acrescentar outros alunos o procedimento é o mesmo. Selecionamos as linhas do `TextView` e vamos em "Menu > Edit > Paste" ou usamos os atalhos. E após colar as linhas, basta trocar o nome do aluno, adicionaremos "Felipe":

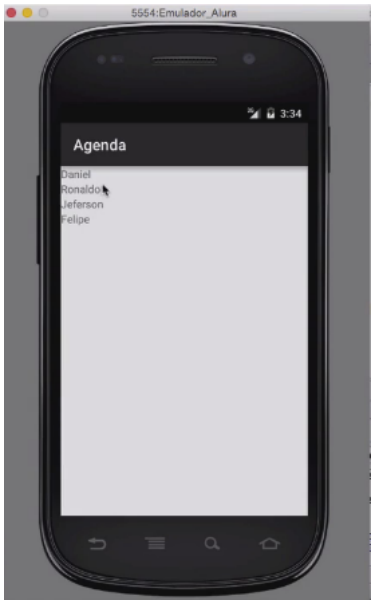
```
<TextView android:text="Felipe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Agora temos quatro novos alunos! Observe:

```
<TextView android:text="Daniel"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content" />

<TextView android:text="Ronaldo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView android:text="Jeferson"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView android:text="Felipe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Lembrando: Para visualizar a tela basta salvar, apertar o *play*, escolher o emulador que já está rodando e aguardar iniciar. Para trazer o emulador para frente existe o atalho "Alt + Tab".

Todos os alunos novos que inserimos vão aparecer na tela do celular. Observe a simulação:



Depois de aprender a inserir os nomes, vamos introduzir algumas novidades!

Vamos trazer esses mesmos alunos a partir ou de um arquivo ou de um banco de dados ou de um *webservice* ou de outro lugar que não o `.xml`. Vamos começar pelo arquivo! Essa alteração será feita no `Activity`. Afinal, é na `Activity` que fica o comportamento da tela e é onde escrevemos o código Java.

O próximo objetivo é colocar um arquivo do `.xml` na `Activity` e para isso vamos na aba `ListaALunosActivity.java`. Existe uma forma simples de fazer esse procedimento. O que faremos é dizer para o *Android* que queremos exibir na tela uma lista e no código adicionaremos os elementos dessa lista. Os componentes que desejamos mostrar na tela estão no `.xml` mas, de onde eles serão trazidos? A parte da lógica ficará na `Activity`.

Portanto, faz sentido que os alunos que estão no `.xml`, ou seja, os `Views` sejam trazidos de outra fonte, arquivo, webservice ou banco de dados na `Activity`. O bom disso é não precisar alterar o `.xml` e manter a separação de comportamento na `Activity` *vs.* conteúdo na `xml`.

Para fazer isso apagaremos todos os `TextView` da `.xml` Isto é, removeremos o seguinte:

```
<TextView android:text="Daniel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView android:text="Ronaldo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView android:text="Jeferson"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView android:text="Felipe"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Apagaremos os `View` para justamente não precisar recriá-los novamente. Ficaremos apenas com isso na `.xml` :

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

Depois de apagar os `TextView` na `.xml` vamos acrescentar um componente diferente do *Android*, a `ListView` . Esse componente representa uma lista. Damos um *Enter* na última linha da `LinearLayout` e ao digitar as primeiras letras da palavra `ListView` , ela já irá aparecer. Agora é só selecionar e dar *enter*.

Lembrando: Qualquer tipo de `View` criada na `.xml` deve ser seguida pela altura e largura. Ao em vez de inserir o `wrap_content` , que faz com que nossa lista acompanhe as medidas do conteúdo inserido nela, vamos pedir para que ela ocupe toda a tela inicial. Portanto, inserimos após o `layout_widht` e o `layout_height` o `match_parent` .

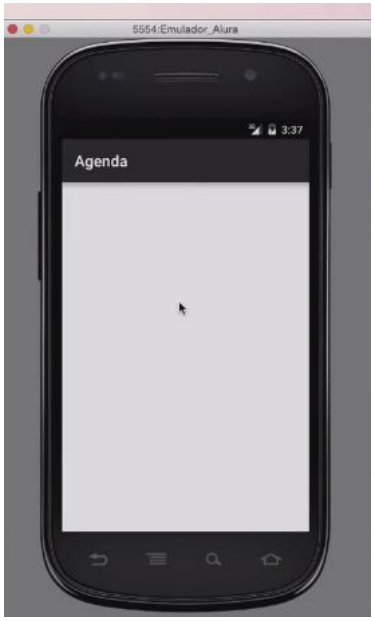
Vamos ter:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">


    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Relembrando: O `match_parent` indica que a largura e a altura serão iguais a do "pai", nesse caso, as medidas da `ListView` serão iguais a do `LinearLayout` , o pai. Esta última, é a classe raiz, que por sua vez não acompanha as medidas de ninguém, pois ela acompanha as medidas da tela inteira.

Para a lista é apenas isso que precisamos especificar. Não esqueça de fechar a *tag* com o `/>` !

Mas, ao rodar o emulador veremos que a lista estará vazia. Mesmo que tenhamos introduzido uma lista no formulário falta nela o seu conteúdo e, por isso, ela aparece vazia:



Para alterar o conteúdo da lista temos que voltar na `Activity` e escrever um código para populá-la. Vamos acessar a aba `ListaAlunosActivity.java` e nela encontramos o seguinte:

```
package br.com.alura.agenda;

import ...

public class ListaAlunosActivity extends AppCompatActivity {

    @0verride
    protected void onCreate(Bundle saverInstanceStatel) {
        super.onCreate (savedInstanceState);
        setContentView(R.layout.activity_lista_alunos);

        }
}
```

Poderíamos, primeiro, abrir um arquivo ou um banco de dados, mas, no momento, esse não é o o foco. Para introduzir um conteúdo na `Activity` , isto é, uma "lista de alunos", vamos declarar apenas uma *Array* simples, uma `String` e dentro dela inserimos os alunos. Então, temos a seguinte *Array*: `String[] = {}` . Dentro das chaves inserimos os nomes dos alunos. Não podemos esquecer de acrescentar também o nome da variável, no nosso caso, "alunos". Ficaremos com:

```
package br.com.alura.agenda;

import ...

public class ListaAlunosActivity extends AppCompatActivity {

    @0verride
    protected void onCreate(Bundle saverInstanceStatel) {
        super.onCreate (savedInstanceState);
        setContentView(R.layout.activity_lista_alunos);

        String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};
```

```
    }
}
```

Queremos trazer também a lista que está no `.xml` para a `Activity`. Abaixo do que acabamos de escrever na `Activity`, na próxima linha, acrescentamos `listadealuno`, um ponto e `add`. Isto é: `listaAluno.add()`. Ficamos com:

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

        String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};
        listaAluno.add()

    }
```

O *add*, de adicionar, vem justamente para acrescentar algo, no caso, cada aluno da *Array*. Precisamos, ainda, referenciar a `listaAlunos.add()`. Para isso usaremos um método simples da `AppCompatActivity`, apagaremos o `listaAlunos.add()` e escreveremos a palavra *find*. Logo abaixo completaremos com o `findViewById`.

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

        String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};
        findViewById

    }
```

O que estamos fazendo é localizando uma `View` através da especificação de um `Id`. Precisamos dizer para o *Android* qual `View` gostaríamos de localizar. Para encontrá-la é preciso voltar ao `.xml`, que está assim:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Vamos especificar na `.xml` uma identificação para encontrarmos a `View` que queremos. Para isso iremos acrescentar um atributo na `ListView`, o `id`.

O `id` é seguido de um `@` que serve para referenciar um recurso da pasta "res", acrescentamos na sequência o sinal de "+" que é usado para falar que estamos criando um recurso, ou seja, uma identificação para a lista. Também digitamos o tipo de recurso que estamos manipulando, nesse caso, um `id`. Depois do id colocamos uma barra para separar a chave

do `id` que especificamos na frente. Depois da barra inserimos o nome que queremos dar a `ListView`, no caso, "lista_alunos". O `id` é `id="@+id/lista_alunos"/>`. E o código ficará assim:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/lista_alunos" />


</LinearLayout>
```

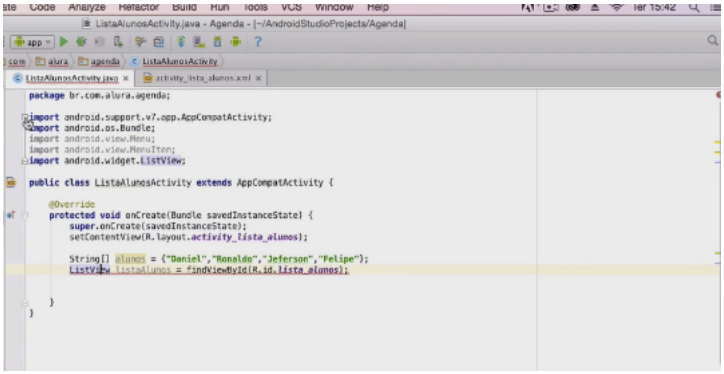
Voltando na `Activity` vamos, finalmente, referenciar a lista no `findViewById`. Ele vai pedir o `id` que definimos no `.xml`. Podemos utilizar a classe `R` e especificar apenas o `id` desejado, o `lista_alunos`. Agora a chamada do método está completa: `findViewById(R.id.lista_alunos)`. O todo ficará assim:

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

        String[] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};
        findViewById(R.id.lista_alunos);

    }
```

Quando inserimos o `findViewById` ele devolve uma instância da `View` gerada no `setContentView`. Agora, falta guardar o `findViewById`. Vamos inserir na `Activity` uma variável `ListView` e nomeá-la `ListaAluno`. Assim, `ListView ListaAluno` vai ficar da seguinte forma:

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

        String[] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};
        ListView listaAlunos = findViewById(R.id.lista_alunos);

    }
```

A `ListView` será sublinhada em vermelho pois, ainda necessitamos fazer um *import*. Aparecerá em cima da palavra `ListView` um balão em azul, basta pressionar "Alt+Enter" e ele fará o *import* automaticamente. Para ver o *import* basta clicar no símbolo de positivo (+), como mostra a figura:



Como é possível perceber pela imagem acima, a `ListView` ainda está em vermelho. Para resolver isso basta converter a referência de `View` para `ListView` através de um *casting*. Vamos introduzir entre parênteses que iremos converter essa referência, assim, teremos: `(ListView)` . Ficamos com:

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

String[] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"} ;
ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);

}
```

Agora temos,

- uma lista de alunos: `String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};`
- o componente ou a `View` de alunos: `ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);`

Falta inserir na lista os nomes dos alunos. Para fazer isso precisamos converter os alunos da `String` para `View` . A classe `ArrayAdapter` faz exatamente isso! Para instancia-la, pulamos para a próxima linha e inserimos `new ArrayAdapter <>` :

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

String[] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"} ;
ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);
new ArrayAdapter <>

}
```

Entre os argumentos `<>` do *generics* vamos preencher com a `String` , pois é isso que desejamos importar. Teremos: `new ArrayAdapter<String>` .

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

    String[] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"} ;
    ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);
```

```
        new ArrayAdapter <String>;

    }
```

Caso um balão azul apareça, isso significa que a classe ainda não foi importada, basta dar "Alt+Enter" que logo acima a classe `ArrayAdapter` será trazida. Falta especificar os parâmetros da `Array Adapter` , que pede o contexto e também o *layout*. O contexto serve para identificação e o que faremos é passar uma referência para a própria `Activity` . Para passar uma referência basta inserir o `this` . Então, acrescentamos ao `new ArrayAdapter<String>` , entre parênteses, o `this` . Ficaremos com `new ArrayAdapter<String>(this, )` .

O segundo parâmetro é o *Layout*. Como utilizaremos um *layout* padrão vamos utilizar a classe `R` do próprio *Android*. Escrevendo, `android.R.layout.simple_list_item_1` que é simplesmente um `TextView` que auxilia a mostrar o nome do aluno. Teremos:

```
new ArrayAdapter <String> (this, android.R.layout.simple_list_item_1)
```

Falta dizer quais são os dados que queremos converter em 'Views', que nesse caso são os alunos. Assim, só completamos depois da palavra "item", com uma vírgula, um espaço e a palavra "alunos", que é a fonte de dados. Teremos:

```
new ArrayAdapter <String> (this, android.R.layout.simple_list_item_1, alunos);
```

Agora a `ArrayAdapter` está instanciada e para que a referência não se perca vamos digitar na frente dela uma variável do tipo `ArrayAdapter` de `String` , que chamaremos de `adapter` . Com isso acrescentaremos o seguinte, `ArrayAdapter<String> adapter` e, por fim, teremos:

```
ArrayAdapter<String> adapter = new ArrayAdapter <String> (this, android.R.layout.simple_list_ite
```

Retomando, temos dois objetos separados, uma Lista de alunos, que é a `ListView` e também, um `adapter` que irá converter os alunos que são `String` em `View` para serem introduzidos na lista. Precisamos ,agora, juntar as duas coisas! Atenção para as cores, se só a variável está cinza é por que ainda não a utilizamos em nenhum lugar. Mas, agora, nós vamos utilizar!

Vamos dizer para a própria lista para utilizar o `adapter` como sendo seu `adapter` . Isso pode soar um pouco estranho, mas na verdade o que queremos dizer é que a lista deverá pedir ao `adapter` para converter os alunos em `View` . Assim, na linha de baixo do `ArrayAdapter` , digitamos `listaAlunos.setAdapter(adapter);` :

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

        String[] alunos ={"Daniel","Ronaldo","Jeferson","Felipe"};
        ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_l
        listaAlunos.setAdapter(adapter);
```

Repare que as palavras `adapter` mudarão de cor, elas estarão com a cor preta para demostrar que estão sendo utilizadas. Vamos rodar o emulador para ver o que acontece:

Temos uma lista bem mais completa, com todos os nomes dos alunos, com um comportamento diferente, espaçamento e até um animação.