

Programação Para Dispositivos Móveis I

ListView Adapters

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

ListView

É um grupo de **views** que apresenta uma **lista** através de uma **fonte de dados** (por ex. um array ou um objeto).

No seu funcionamento, apresenta um conjunto de items idênticos, que partilham o mesmo tipo de layout (e respetivas configurações).

<https://developer.android.com/reference/android/widget/ListView>

CountryList

Name:	Angola
Continentes:	Africa
Name:	Argentina
Continentes:	America
Name:	Brazil
Continentes:	America
Name:	Chile
Continentes:	America
Name:	Cuba
Continentes:	America
Name:	Egypt
Continentes:	Africa
Name:	Germany
Continentes:	Europe
Name:	Greece
Continentes:	Europe

ListView

main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:entries="@array/array_countries" />

</LinearLayout>
```

No ficheiro de layout onde desejarmos incluir a nossa list, deveremos inserir o elemento `ListView`.

Na propriedade `android:entries` é possível definir a fonte de dados a ser preenchida na lista. Esta fonte de dados está explícita no ficheiro `strings.xml`

ListView

strings.xml

```
<string-array name="array_countries">
  <item>Portugal</item>
  <item>United Kingdom</item>
  <item>Mexico</item>
  <item>Argentina</item>
  <item>Brazil</item>
  <item>Egypt</item>
  <item>Sidney</item>
  <item>Greece</item>
  <item>Japan</item>
  <item>Istanbul</item>
  <item>South Africa</item>
  <item>Thailand</item>
</string-array>
```

Exemplo de uma entrada no ficheiro strings.xml que contém um array de strings composto pelos nomes de diferentes países.

ListView

onItemClickListener()

```
private ListView listView;

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

    listView = (ListView) findViewById(R.id.list_view);
    listView.setOnItemClickListener(this);
}

@Override
public void onItemClick(AdapterView<?> parent,
                        View view,
                        int position,
                        long id) {

    Log.i("ITEM_CLICKED", listView.getItemAtPosition(position).toString());

}
```

Cada elemento da nossa lista (item) pode sofrer interações por parte do utilizador.

À semelhança do método `onClickListener()`, as `ListView`s tem uma propriedade para verificar que item sofreu um click, o `onItemClickListener`.

A activity deverá implementar a interface `AdapterView.OnItemClickListener` que lhe irá indicar para a implementação do método `onItemClick`

ListView

Conteúdo dinâmico

A inclusão da propriedade `android:entries` e respetiva caracterização no ficheiro `strings.xml` através de um `string-array`, é limitativa e apenas contempla **conteúdo estático**.

Se pretendermos popular as nossas listas com **conteúdo dinâmico**, é necessário a inclusão de um **adapter**.

Adapters

Os **adapters** são responsáveis pelo **tratamento e apresentação dos dados**.
Estabelecem uma ponte entre um AdapterView (ex. ListView, Spinner) e os dados que serão preenchidos nessa View.

Permitem a gestão de grandes conjuntos de dados de uma maneira **eficiente e escalável**.

Constroem o layout para cada um dos elementos da lista (requer a criação de um layout adicional que representa como cada elemento da lista será apresentado).

ListView (com adapter)

main_activity.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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"
    android:orientation="vertical"
    tools:context=".MainActivity">

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

</LinearLayout>
```

Com a utilização de adapters (e conteúdo dinâmico) removemos a propriedade `android:entries`. A gestão do conteúdo da lista será feita de forma dinâmica.

ListView (com adapter)

list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/txt_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textStyle="bold"
    android:textSize="45dp"
    android:layout_marginLeft="12dp"
    android:layout_marginTop="5dp">

</TextView>
```

Com a utilização de adapters (e conteúdo dinâmico) é necessário definir como cada item da lista será apresentado.

Isto é feito através da criação de um novo layout que contém as definições de um elemento da lista. Este layout será replicado por todos os elementos da lista.

ListView (com adapter)

Adapter

Lista que irá ser preenchida com o conteúdo do array de strings em `strings.xml`

```
private ListView listView;  
private String[] itemList;  
private ArrayAdapter<String> adapter;
```

Definição do adapter

`@Override`

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

Preencher o array com o conteúdo do array de strings em `strings.xml`

```
    listView = (ListView) findViewById(R.id.list_view);  
    listView.setOnItemClickListener(this);
```

```
    itemList = getResources().getStringArray(R.array.array_countries);
```

```
    adapter = new ArrayAdapter<String>  
        (this, R.layout.list_item, R.id.txt_item, itemList);
```

Inicialização do adapter, com o layout que deverá utilizar para cada item, e respetivos dados

```
    listView.setAdapter(adapter);
```

Associação do adapter à ListView

ListView (com adapter)

onItemClick()

@Override

```
public void onItemClick(AdapterView<?> parent,  
    View view,  
    int position,  
    long id) {
```

```
    adapter.getItem(position);  
    Log.i("ITEM_CLICKED", listView.getItemAtPosition(position).toString());  
}
```

Com a utilização do adapter podemos aceder
ao item que foi clicado através do método
getItem()

Spinner

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
```

```
// Criar um ArrayAdapter utilizando um array de strings e o layout predefinido da spinner
```

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.data, android.R.layout.simple_spinner_item);
```

```
// Especificar o layout a utilizar quando a lista aparece
```

```
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

```
spinner.setAdapter(adapter);
```

Custom ArrayAdapter

CustomAdapter.java

```
public class CustomAdapter extends ArrayAdapter<E>
```

```
public CountryListAdapter(Activity context, E[] itemList){  
    super(context, R.layout.list_item, itemList);  
    this.context = context;  
    this.itemList = itemList;  
}
```

- Quando pretendemos atualizar e apresentar uma **lista de objetos complexos com um layout personalizado** devemos utilizar um **ArrayAdapter**.
- O tipo <E> deverá ser substituído pelo tipo de dados que queremos utilizar (ex. uma classe);
- No construtor devemos especificar (para a classe superior) qual o layout a utilizar e passar os objetos (dados) a serem apresentados na interface.

Custom ArrayAdapter

CustomAdapter.java

```
public View getView(int position, View view, ViewGroup parent){  
    LayoutInflater inflater = context.getLayoutInflater();  
    View itemRowView = inflater.inflate(R.layout.list_item, null, true);  
  
    TextView txtView = (TextView) itemRowView.findViewById(R.id.txt_view);  
  
    txtView.setText(itemList[position].getName());  
  
    return itemRowView;  
}
```

- Sempre que a interface necessita de ser atualizada o método `getView()`, da classe do `ArrayAdapter`, é invocado para **cada item** da lista.

Gestão dos Adapter e ListView

- Se uma ListView consegue apresentar em simultâneo apenas 5 elementos, nunca deverá instanciar mais do que 5 views, mesmo que na estrutura de dados existam 100 elementos;
- As views utilizadas para cada item, serão sempre reaproveitadas quando é efetuado o scroll/drag da lista;

`getView (int position, View view, ViewGroup parent)`

- position: o índice do elemento da lista a apresentar
- view: se != null, contém um layout previamente instanciado, e que deve ser reutilizado
- parent: contentor das views correspondentes a elementos da lista

Programação Para Dispositivos Móveis I

ListView Adapters

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt