



# Programação para Android

Aula 05: Estilos e temas; galeria de imagens

# Objetivos

- ▶ Aplicar estilos e temas
- ▶ Criar uma galeria de imagens com o widget Gallery



# Parte 01: Estilos e temas

# Interface de Usuários - Estilos e Temas

- ▶ Um estilo é uma coleção de propriedades que especificam o visual e formato de uma View ou janela. Um estilo por especificar propriedades como altura, padding, cor de fonte, tamanho da fonte, cor de fundo e muito mais. Um estilo é definido em um recurso XML que é reparado do XML que especifica o layout.
- ▶ Estilos em Android compartilham uma filosofia similar ao que é o CSS para web design - eles permitem a você separar o design do conteúdo.

# Interface de Usuários - Estilos e Temas

- ▶ Por exemplo, usando um estilo, você pode pegar esse XML de layout...

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#00FF00"
    android:typeface="monospace"
    android:text="@string/hello"
/>
```

- ▶ ...e torná-lo em algo como isso:

```
<TextView
    style="@style/Brasil"
    android:text="@string/Pais" />
```

# Interface de Usuários - Estilos e Temas

- ▶ Todos os atributos relacionados a estilo foram removidos do layout XML e colocados em um estilo definido pelo nome CodeFont, que é então aplicado com o atributo style.
- ▶ Um tema é um estilo aplicado em toda uma atividade ou aplicação, ao invés de apenas em uma View individual (como no exemplo acima). Quando um estilo é aplicado como um tema, cada View na atividade vai ter o estilo aplicado onde for suportado. Por exemplo, você pode aplicar o mesmo estilo CodeFont como um tema para uma atividade e então todo o texto dentro da atividade vai ter uma fonte mono espaçada de cor verde.

# Definindo Estilos

- ▶ Para criar um conjunto de estilos, abra o arquivo **style.xml** na pasta **res/values** dentro do seu projeto.
- ▶ Ao abrir este arquivo você irá verificar o nó raiz do arquivo XML de estilo deverá ser **<resources>**.

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">
```

- ▶ Para cada estilo que você queira criar adicione um elemento **<style>** ao arquivo com um *name* que identifique unicamente o estilo. Então adicione elementos **<item>** para cada propriedade do estilo, com o *name* que declare a propriedade de estilo e o valor a ser usado.
- ▶ O valor para o **<item>** pode ser uma string, um código hexadecimal para cor, uma referência para outro recurso ou outro valor dependendo das propriedades do estilo.

# Definindo Estilos

- Aqui está um exemplo de um estilo:

```
<style name="Textos" parent="AppTheme">
    <item name="android:textColor">#0000FF</item>
    <item name="android:textSize">20px</item>
    <item name="android:typeface">serif</item>
</style>
```

```
<style name="Botao" parent="AppTheme">
    <item name="android:textColor">#00FF00</item>
    <item name="android:textSize">40px</item>
    <item name="android:textAllCaps">true</item>
    <item name="android:typeface">serif</item>
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
</style>
```



# Definindo Estilos

- ▶ O atributo ***parent*** dentro do elemento <style> é opcional e especifica o ID de recurso de um outro estilo a partir do qual esse estilo deverá herdar propriedades.
- ▶ Cada filho do elemento <resources> é convertido em um objeto de recurso da aplicação em tempo de compilação, que pode ser referenciada pelo valor do atributo ***name*** dentro do elemento <style>. Esse estilo de exemplo pode ser referenciado a partir de um XML de layout usando @style/**CodeFont**.
- ▶ Para utilizar um estilo criado em um arquivo de layout, usamos o atributo **style** conforme a seguir:

```
<TextView
    android:text="Endereço"
    style="@style/Textos" />
<Button
    android:text="Verificar"
    style="@style/Botao" />
```

# Definindo Estilos

## ► Resultado



The image shows a web form titled "EstilosTemas" with a black header bar containing an Android icon and the title. Below the header, there are three input fields with blue labels: "Nome:", "Passaporte", and "Endereço". Each field has a corresponding input line. At the bottom of the form is a large, light gray button with the word "VERIFICAR" in green capital letters.

# Definindo temas

- ▶ Um theme é um estilo aplicado a uma Activity ou aplicação inteira, ao invés de elemento por elemento. Quando um estilo é aplicado como um theme, todos os widgets na Activity ou aplicação irão usar todas as propriedades de estilo por ele definidas.
- ▶ Para aplicar uma definição de estilo como um tema, você deve aplicá-lo à atividade ou aplicação dentro do manifesto (AndroidManifest.xml) do Android.



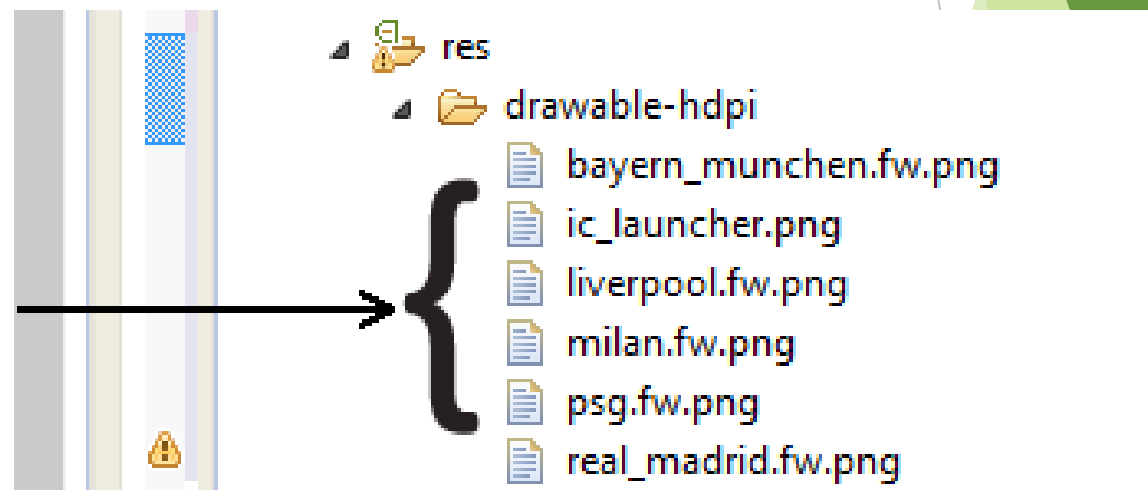
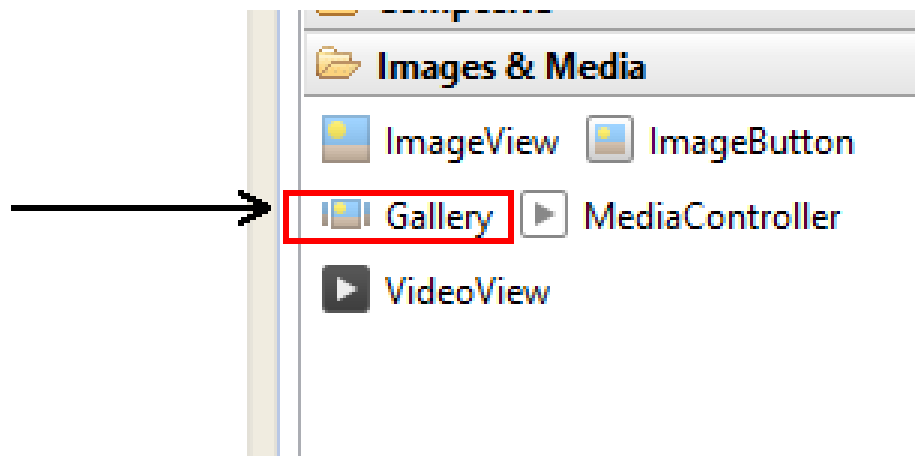
```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Textos" >
<activity
```

- ▶ Quando você faz dessa maneira, cada elemento dentro da atividade ou aplicação vai aplicar as propriedades que são suportadas. Quaisquer elementos que não suportem alguma das propriedades não a terão aplicada. Se um elemento suporta apenas uma das propriedades, apenas essa propriedade será aplicada.

## Parte 02: Gallery

# Galeria de imagens

- ▶ O Componente Gallery permite criar uma galeria de imagens em uma aplicação Android. Uma das formas mais comuns de apresentação é visualizar um conjunto de imagens presentes nas pastas res/drawable de nossa aplicação.



# Galeria de imagens

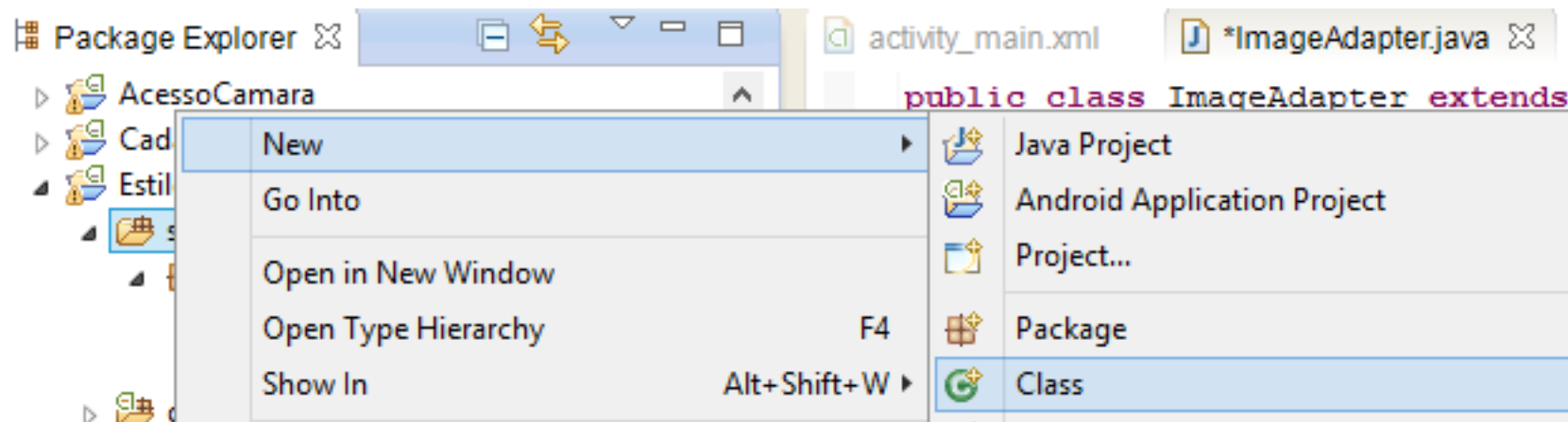
- ▶ O componente Gallery pode ser adicionado no layout.xml conforme a seguir:

```
<Gallery  
    android:id="@+id/galeriaImagem"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
/>
```

- ▶ Para popular o Gallery com um conjunto de imagens, precisamos programar uma classe java baseada na classe BaseAdapter. Esta classe funciona com um adaptador de dados que fará a ligação entre as imagens presentes na aplicação e componente Gallery.
- ▶ Felizmente esta classe já está criada por programadores ao redor do mundo, não sendo necessário programa-la. Neste exemplo iremos apenas configurar as imagens que serão utilizadas.

# Galeria de imagens

- Para criar uma classe nova, clique com o botão direito no diretório src e escolha a opção NEW -> CLASS.



# Galeria de imagens

- ▶ Em seguida dê um nome a esta classe. Iremos chama-la de ImageAdapter. Em seguida clique em FINISH.
- ▶ Depois de criada, copiaremos o código em java disponibilizado em anexo a esta apresentação para o arquivo com a classe recém criada.
- ▶ OBSERVAÇÃO: VERIFICAR SE O PACKAGE DA CLASSE TEM O MESMO NOME DO PROJETO.

`package com.example.estilostemas;`

The screenshot shows the 'New Java Class' dialog box. At the top, it says 'Java Class' and 'The use of the default package is discouraged.' Below this, there are fields for 'Source folder' (set to 'EstilosTemas/src'), 'Package' (set to '(default)'), and 'Enclosing type'. The 'Name' field is set to 'ImageAdapter'. Under 'Modifiers', 'public' is selected. Under 'Superclass', 'java.lang.Object' is selected. The 'Interfaces' section is empty. Under 'Which method stubs would you like to create?', 'Inherited abstract methods' is checked. At the bottom, there are 'Finish' and 'Cancel' buttons.



# Galeria de imagens

- No método onCreate da classe relacionada a Activity iremos relacionar o galeria de imagem criada no layout.xml.

```
galeria = (Gallery) findViewById(R.id.galeriaImagem);
```

- Em seguida será criado um vetor com as imagens presentes na pasta drawable.

```
//CRIAÇÃO DE UM VETOR DE INTEIRO COM AS IMAGENS ARMAZENADAS NA PASTA  
DRAWABLE  
int []imagens =new  
int[] {R.drawable.bayern_munchen,R.drawable.liverpool,R.drawable.milan,R.  
drawable.psg,R.drawable.real_madrid};
```

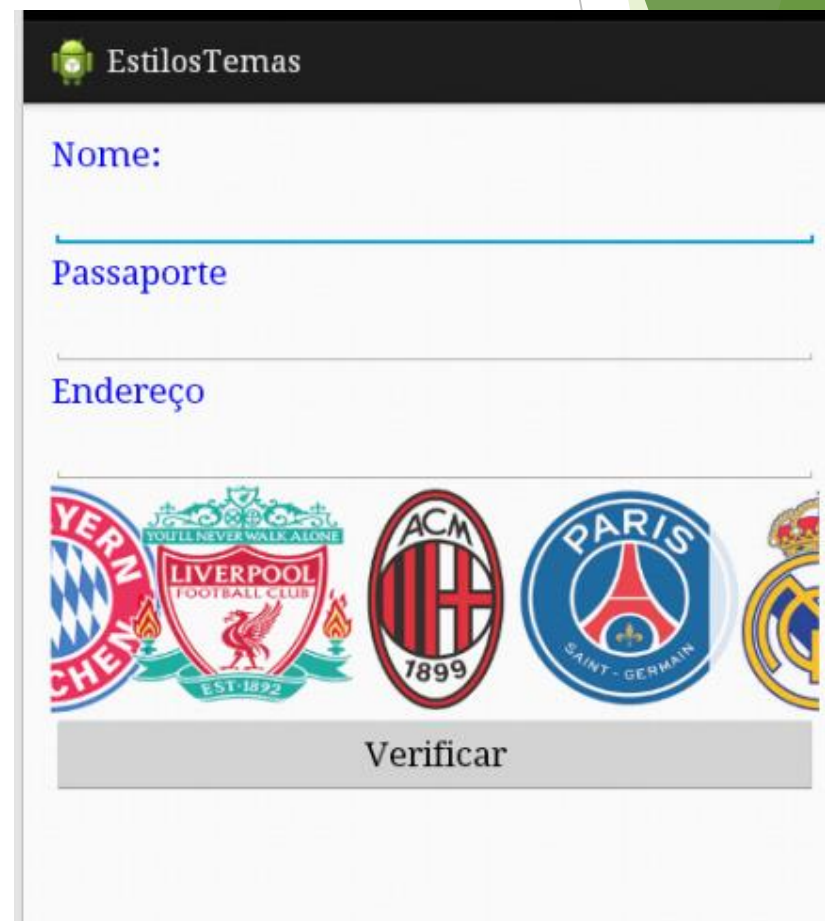
# Galeria de imagens

- Por fim criamos um objeto do tipo ImageAdapter (classe que foi criada para relacionar uma fonte de dados com um objeto Gallery) e em seguida alteramos a fonte de dados da Gallery para este objeto.

```
//CRIA UM OBJETO DO TIPO IMAGE ADAPTER ATRIBUINDO O VETOR DE  
IMAGENS  
ImageAdapter imgAdapter=new ImageAdapter(this, imagens);  
//ALTERA A FONTE DE DADOS DA GALERIA  
galeria.setAdapter(imgAdapter);
```

# Galeria de imagens

- ▶ O resultado pode ser verificado quando a aplicação for executada. O usuário ao arrastar o dedo na tela poderá verificar a alteração das imagens.
- ▶ Para verificar qual imagem foi selecionada podemos utilizar o evento `OnItemClickListener`.
- ▶ Para isto devemos programar o código relacionado ao clique da imagem e em seguida atribuí-lo ao objeto Gallery. Veja no slide a seguir este processo.



The screenshot shows an Android application interface with a black header bar containing an Android robot icon and the text "EstilosTemas". Below the header, there are three text input fields with blue labels: "Nome:", "Passaporte", and "Endereço". Each field has a blue underline. Below the input fields, there is a horizontal row of five football club logos: Chelsea, Liverpool, AC Milan, Paris Saint-Germain, and Real Madrid. Below the logos is a grey button with the text "Verificar".

# Galeria de imagens

- Evento relacionado ao clique da imagem

```
OnItemClickListener selecionar = new.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,  
        long arg3) {  
        // TODO Auto-generated method stub  
        Toast.makeText(getApplicationContext(), "Você selecionou a imagem de nº 1" +  
            (arg2 + 1), Toast.LENGTH_SHORT).show();  
    }  
};
```

- Referenciando o evento a uma galeria. O processo abaixo deve ser feito no método onCreate da Activity.

```
galeria.setOnItemClickListener(selecionar);
```

# Na próxima aula...

- ▶ Widgets: Spinner e ListView

