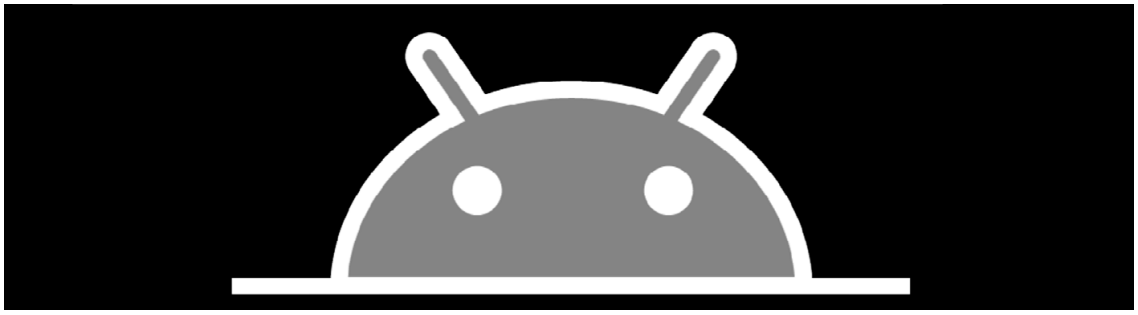


Capítulo VI - Trabalhando com internacionalização e literais na plataforma Android

Aprenda neste capítulo a tirar o máximo de vantagem do uso das literais e da internacionalização nos aplicativos Android

O processo desenvolvimento de aplicativos evoluiu muito nas últimas décadas. Foi-se o tempo quando os desenvolvedores conseguiam ganhar muito dinheiro desenvolvendo aplicações simples, que trabalhavam de forma isolada e que eram disponibilizadas totalmente no mercado nacional. Com as limitações da Internet há algumas décadas, a disponibilização dos aplicativos acontecia com a venda de CDs ou disquetes. Estes eram instalados em computadores desktop e podiam ser replicados para outros computadores sem muitas dificuldades. Neste mercado, os softwares eram muito valorizados, principalmente quando este resolvia um problema do dia a dia e os usuários não hesitavam em investir grandes quantias em softwares confiáveis.

Hoje os tempos mudaram, assim como o processo de desenvolvimento de software, as plataformas de execução e a forma de fazer dinheiro com eles. Temos acesso à Internet e inúmeros computadores pelos quais passamos ao longo do dia (isso mesmo, hoje, uma pessoa normal pode contar com smartphones, tablets, TV digital, canetas inteligentes, carros com computadores de bordo, caixas eletrônicos, relógios inteligentes e, claro, não podemos esquecer dos computadores desktops e laptops). Assim, esses dispositivos precisam executar programas e esses programas estão disponíveis na rede, de forma gratuita ou paga, ou ainda em portais, como, por exemplo, o Google Play.

Assim, os softwares desenvolvidos para os smartphones Android podem ser acessados e baixados por pessoas de qualquer parte do mundo, que pagarão por esses aplicativos em reais, dólares, euros, libras ou qualquer outra moeda. Entretanto, é necessário que esses softwares estejam acessíveis para todos os usuários.

O termo acessível não se refere apenas em deixar o apk, o instalador de um aplicativo Android, disponível para download na Internet. Acessível refere-se a tornar o aplicativo utilizado por diferentes usuários, que falam diferentes idiomas, possuem diferentes culturas, que trabalham com diferentes unidades de medidas.

Um aplicativo simples, como um que calcula o IMC (índice de massa corpórea) pode funcionar perfeitamente no Brasil. Se traduzirmos o aplicativo e colocarmos no mercado americano, a venda desse aplicativo poderá ser um desastre, pois além do idioma, os Estados Unidos

usam um sistema de medidas diferente do Brasil e lá não se costuma medir o peso em quilogramas ou a altura em centímetros. Desta forma, a maioria dos americanos não saberia sua altura em cm ou seu peso em kg, o que inviabilizaria a utilização do software.

Outra mudança está no formato de navegação e leitura. Os povos ocidentais costumam ler da esquerda para direita, de cima para baixo, porém, em alguns povos, a leitura acontece de forma invertida. Assim, uma propaganda de sabão em pó, como a apresentada na **Figura 6.1**, faz sentido para os povos ocidentais, indicando que a roupa lavada com esta marca de sabão fica limpa no final do processo.

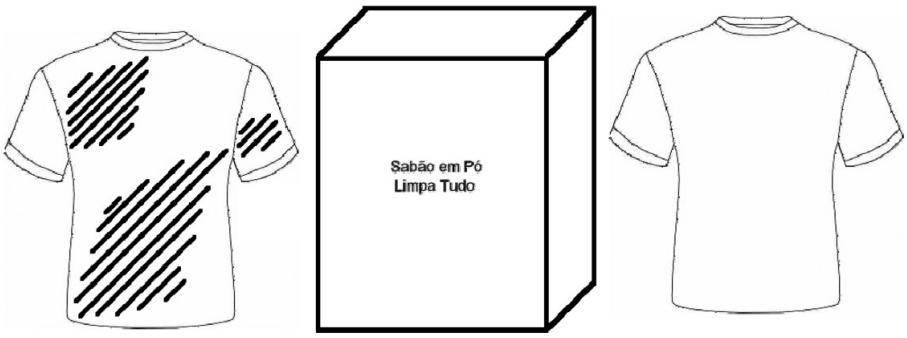


Figura 6.1. Exemplo típico de imagem que deve ser refeita dependendo da cultura onde será apresentada.

Agora, em outras culturas, onde se lê da direita para a esquerda, uma imagem como esta significa que o sabão em pó consegue deixar uma camiseta que está limpa toda suja. Assim, ao trabalhar com a internacionalização, deve-se preocupar não só com a tradução dos textos como muitos imaginam, mas com as imagens, o avanço das telas que normalmente acontece da direita para a esquerda, com o formato da data/hora, unidades de medidas e muito mais.

Assim, este capítulo apresentará algumas dicas de internacionalização na plataforma Android, focando principalmente no processo para traduzir o texto e personalizar alguns recursos dentro do aplicativo.

Como estudo de caso, será utilizado o aplicativo CalculaIMC, este já tratado também pelos capítulos anteriores. Quem já desenvolveu o aplicativo, pode pular para a seção “Tratando as literais de uma aplicação Android”.

Estudo de caso – Calcular o IMC

Para a utilização dos recursos de depuração, um projeto chamado CalculaIMC deve ser criado. Este deve possuir um arquivo de layout (activity_principal.xml), presente na pasta res, subpasta layout, e uma Activity principal (PrincipalActivity.java), presente no pacote br.com.livro.calculaimc. A estrutura do projeto, após criado, é apresentada na **Figura 6.2**.

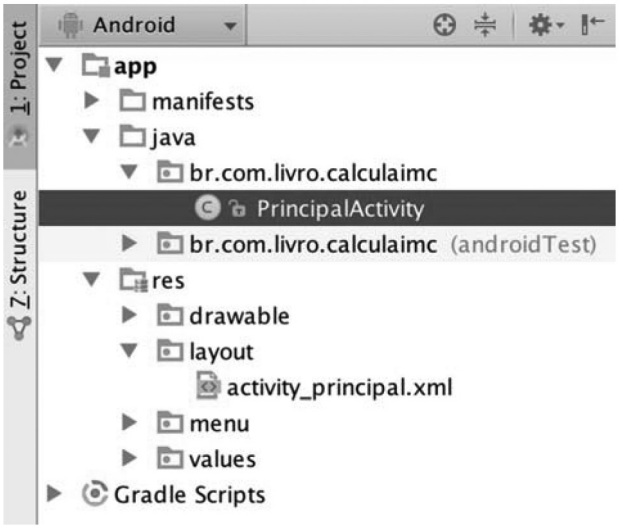


Figura 6.2. Estrutura do projeto no Android Studio.

O arquivo activity_principal.xml terá o layout apresentado na **Listagem 6.1**.

Listagem 6.1. activity_principal.xml – Interface gráfica principal do aplicativo.

```
01.        <LinearLayout xmlns:android=”http://schemas.android.com/
apk/res/android”
02.            xmlns:tools=”http://schemas.android.com/tools”
03.            android:layout_width=”match_parent”
004.            android:layout_height=”match_parent”
05.            android:orientation=”vertical”
06.            tools:context=”.PrincipalActivity” >
07.
08.            <TextView
09.                android:layout_width=”wrap_content”
10.                android:layout_height=”wrap_content”
```

```
11.                 android:text=”Peso:” />
12.
13.             <EditText
14.                 android:id=”@+id/etPeso”
15.                 android:layout_width=”wrap_content”
16.                 android:layout_height=”wrap_content”
17.                 android:inputType=”numberDecimal”
18.                 android:text=”” />
19.
20.             <TextView
21.                 android:layout_width=”wrap_content”
22.                 android:layout_height=”wrap_content”
23.                 android:text=”Altura:” />
24.
25.             <EditText
26.                 android:id=”@+id/etAltura”
27.                 android:layout_width=”wrap_content”
28.                 android:layout_height=”wrap_content”
29.                 android:inputType=”numberDecimal”
30.                 android:text=”” />
31.
32.             <TextView
33.                 android:layout_width=”wrap_content”
34.                 android:layout_height=”wrap_content”
35.                 android:text=”IMC:” />
36.
37.             <TextView
38.                 android:id=”@+id/tvResult”
39.                 android:layout_width=”wrap_content”
40.                 android:layout_height=”wrap_content”
41.                 android:text=”0,0” />
42.
43.              <Button
44.                 android:id=”@+id/btCalcular”
45.                 android:layout_width=”wrap_content”
46.                 android:layout_height=”wrap_content”
47.                 android:text=”Calcular”
48.                 android:onClick=”btCalcularOnClick”/>
49.
50.         <Button
51.                android:id=”@+id/btLimpar”
52.                android:layout_width=”wrap_content”
53.                android:layout_height=”wrap_content”
54.                android:text=”Limpar”
55.                  android:onClick=”btLimparOnClick”/>
56.
57.         </LinearLayout>
58.
```

Como pode ser observado, a interface gráfica não possui recursos avançados, simplesmente foram declarados dois componentes para a

entrada de dados (EditText), dois componentes para o processamento (Button) e um componente para a saída das informações (TextView), cada um com seu nome para a identificação via Activity utilizando a propriedade android:id. Também foram apresentadas na interface algumas literais (TextView).

A classe Java responsável pelo funcionamento do aplicativo é apresentada na **Listagem 6.2**.

Listagem 6.2. PrincipalActivity.java – Classe principal para o cálculo do IMC.

```
01.  package br.com.livro.calculaimc;
02.
03.  import android.app.Activity;
04.  import android.os.Bundle;
05.  import android.view.View;
06.  import android.widget.Button;
07.  import android.widget.EditText;
08.  import android.widget.TextView;
09.  import android.widget.Toast;
10.
11.  import java.text.DecimalFormat;
12.
13.  public class PrincipalActivity extends Activity {
14.
15.      private EditText etPeso;
16.      private EditText etAltura;
17.      private Button btCalcular;
18.      private Button btLimpar;
19.      private TextView tvResultado;
20.
21.      @Override
22.      protected void onCreate(Bundle savedInstanceState) {
23.          super.onCreate(savedInstanceState);
24.          setContentView(R.layout.activity_principal);
25.
26.          etPeso = (EditText) findViewById( R.id.etPeso );
27.          etAltura = (EditText) findViewById( R.id.etAltura );
28.          tvResultado = (TextView) findViewById( R.id.tvResult );
29.          btCalcular = (Button) findViewById( R.id.btCalcular );
30.          btLimpar = (Button) findViewById( R.id.btLimpar );
31.
32.      }//fim do método onCreate
33.
34.      public void btCalcularOnClick( View v ) {
35.          if (etPeso.getText().toString().equals("")) {
36.              Toast.makeText(getApplicationContext(), "Campo Peso
deve ser preenchido",
```

```
37.                     Toast.LENGTH_LONG).show();
38.             etPeso.requestFocus();
39.             return;
40.         }
41.
42.         if (etAltura.getText().toString().equals("")) {
43.             Toast.makeText(getApplicationContext(), "Campo Altura
deve ser preenchido",
44.                     Toast.LENGTH_LONG).show();
45.             etAltura.requestFocus();
46.             return;
47.         }
48.
49.         double peso = Double.parseDouble(etPeso.getText().
toString());
50.         double altura = Double.parseDouble(etAltura.getText().
toString());
51.
52.         double imc = peso / Math.pow(altura, 2);
53.
54.         tvResultado.setText(new DecimalFormat("0.00").
format(imc));
55.
56.     }// fim do método btCalcularOnClick
57.
58.     public void btLimparOnClick( View v ) {
59.         etPeso.setText("");
60.         etAltura.setText("");
61.         tvResultado.setText( "0.0" );
62.     }// fim do método btLimparOnClick
63.
64.
65. } //fim da classe PrincipalActivity
```

A classe apresentada representa a estrutura básica de uma Activity, sendo declarados os componentes visuais presentes no arquivo de layout (linhas 15 a 19). Após, esses componentes são mapeados para a utilização no código Java (linhas 26 a 30) e *Calcular* e linhas 58 a 62 para o botão *Limpar*).

A função desses métodos é simples: btLimparOnClick( ) apenas inicializa os campos da tela com os espaços e textos iniciais, já btCalcularOnClick( ) valida os campos digitados na tela e após, calcula o IMC, apresentando o resultado do processamento com duas casas decimais para o usuário.

Desta forma, o aplicativo está pronto para ser executado a partir do emulador ou device real.

Tratando as literais de uma aplicação Android

É possível ver a interface gráfica e as restrições de digitação nos campos EditText, entretanto, embora a criação da interface utilizando literais diretamente no código xml seja mais prática e rápida, esta não é a melhor opção. O próprio Android Studio informa como *warning* a utilização de literais. Clicando no texto da literal, uma mensagem informativa é apresentada na tela, conforme a **Figura 6.3**.

Figura 6.3. *Warnings* apresentados na utilização de literais.

As vantagens de utilizar um "repositório central de literais" em uma aplicação Android são muitas. Imagine você colocar em um arquivo xml todas as literais do seu aplicativo, ou seja, todos os textos apresentados para o usuário, através de componentes visuais, mensagens informativas, títulos de janelas, menus, enfim, todas as mensagens e literais. Desta forma, ao término do programa, fica fácil fazer uma verificação gramatical e ortográfica, já que todos os textos estão em um único local.

Outra vantagem está na facilidade de padronizar os termos. Por exemplo, em um menu, você colocou a mensagem "Erro no sistema", em outra tela, colocou a mensagem "ERRO no sistema", em outro ponto, "Erro!!!", e assim por diante. Utilizando um repositório de literais, é possível percorrer todos os textos e verificar se algum termo está fora do padrão do aplicativo, deixando o aplicativo com um aspecto mais profissional.

Por fim, a terceira e maior vantagem de se utilizar um repositório de literais é a facilidade para internacionalizar o aplicativo, permitindo traduzi-lo para vários idiomas modificando somente o arquivo de literais, não havendo a necessidade de modificar o código-fonte.

Analisando o código da **Listagem 6.1**, onde temos uma série de literais (“Peso:”, “Altura:”, “IMC:”, “0.0”, “Calcular” e “Limpar”), podemos colocar todo esse conteúdo em um repositório de literais. No Android, esse repositório costuma ser o arquivo strings.xml, sendo que o arquivo se encontra na pasta res, subpasta values, conforme a **Figura 6.4**.

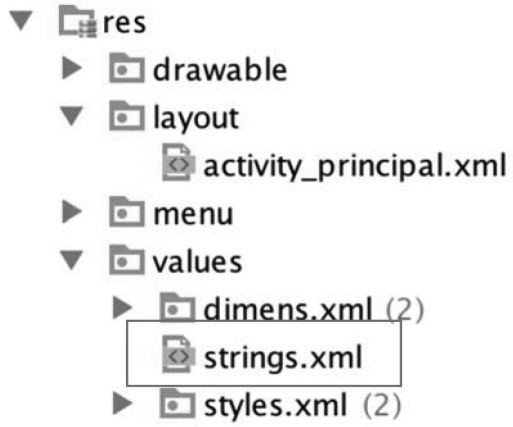


Figura 6.4. Arquivo utilizado como repositório de literais.

Embora possam ser utilizados outros arquivos como repositório de literais, comumente este é o arquivo utilizado. Abrindo-o, algumas literais do aplicativo já estão nele, como, por exemplo, o nome da aplicação (app_name).

Assim, iremos editar esse arquivo, adicionando novas literais, bastando dar dois cliques no arquivo e mudando seu conteúdo xml. O novo conteúdo do arquivo, já com as literais, é apresentado na **Listagem 6.3**.

Listagem 6.3. strings.xml – Repositório de literais do aplicativo.

```
01.  <?xml version=”1.0” encoding=”utf-8”?>
02.  <resources>
03.
04.      <string name=”app_name”>Calcula IMC</string>
05.      <string name=”action_settings”> Configurações </string>
06.      <string name=”altura”>Altura:</string>
07.      <string name=”peso”>Peso:</string>
08.      <string name=”imc”>IMC:</string>
09.      <string name=”zeros”>0.0</string>
10.      <string name=”calcular”>Calcular</string>
11.      <string name=”limpar”>Limpar</string>
12.      <string name=”erroaltura”>Campo altura deve ser preenchido</
string>
```

```
13.      <string name=”erropeso”>Campo peso deve ser preenchido</
string>
14.
15.  </resources>
```

Na **Listagem 6.3**, além do nome do aplicativo e das literais da tela, duas novas mensagens foram adicionadas, referentes às mensagens de erro para os campos altura e peso digitados de forma errada. Eles serão utilizados posteriormente pelo aplicativo.

Dica. Literais criadas com o aplicativo

Ao iniciar um aplicativo Android, algumas literais já foram criadas no strings.xml, como, por exemplo, as literais do menu da aplicação (action_setting) e uma literal hello_world, que é apresentada em um TextView adicionado na tela na criação de um projeto novo. Referente a essas variáveis, action_setting teve seu conteúdo traduzido para o português, já hello_world foi retirado, já que não será mais utilizado por este aplicativo.

Para utilizar o conteúdo do repositório de literais dentro do arquivo activity_principal.xml, é necessário, onde se utilizava uma literal, fazer referência ao arquivo strings.xml e ao nome da tag da literal, ficando o arquivo activity_principal.xml conforme a **Listagem 6.4**.

Listagem 6.4. activity_principal.xml – Interface gráfica principal do aplicativo utilizando literais.

```
01.  <LinearLayout xmlns:android=”http://schemas.android.com/apk/res/
android”
02.      xmlns:tools=”http://schemas.android.com/tools”
03.      android:layout_width=”match_parent”
04.      android:layout_height=”match_parent”
05.      android:orientation=”vertical”
06.      tools:context=”livro.exemple.calculaimc.PrincipalActivity” >
07.
08.      <TextView
09.          android:layout_width=”wrap_content”
10.          android:layout_height=”wrap_content”
11.          android:text=”@string/peso” />
12.
13.      <EditText
14.          android:id=”@+id/etPeso”
15.          android:layout_width=”wrap_content”
16.          android:layout_height=”wrap_content”
```

```
17.          android:inputType=”numberDecimal”
18.          android:text=”” />
19.
20.      <TextView
21.          android:layout_width=”wrap_content”
22.          android:layout_height=”wrap_content”
23.          android:text=”@string/altura” />
24.
25.      <EditText
26.          android:id=”@+id/etAltura”
27.          android:layout_width=”wrap_content”
28.          android:layout_height=”wrap_content”
29.          android:inputType=”numberDecimal”
30.          android:text=”” />
31.
32.      <TextView
33.          android:layout_width=”wrap_content”
34.          android:layout_height=”wrap_content”
35.          android:text=”@string/imc” />
36.
37.      <TextView
38.          android:id=”@+id/tvResult”
39.          android:layout_width=”wrap_content”
40.          android:layout_height=”wrap_content”
41.          android:text=”@string/zeros” />
42.
43.       <Button
44.          android:id=”@+id/btCalcular”
45.          android:layout_width=”wrap_content”
46.          android:layout_height=”wrap_content”
47.          android:text=”@string/calcular”
48.          android:onClick=”btCalcularOnClick”/>
49.
50.       <Button
51.          android:id=”@+id/btLimpar”
52.          android:layout_width=”wrap_content”
53.          android:layout_height=”wrap_content”
54.          android:text=”@string/limpar”
55.          android:onClick=”btLimparOnClick”/>
56.
57.    </LinearLayout>
58.
```

Como pode ser observado no código, todas as literais foram substituídas por suas respectivas referências no arquivo strings.xml. Assim, se o usuário deseja mudar um texto na tela, ele não precisa modificar o arquivo activity_principal.xml e sim, apenas strings.xml.

Literais no código da Activity

O aplicativo para o cálculo do IMC está graficamente pronto, porém, não foi codificada ainda a recuperação das literais a partir da classe PrincipalActivity.java. Essa classe possui uma série de literais em seu conteúdo, como o conteúdo presente na linha 62 (valorização de zeros no componente TextView) e linhas 36 e 42 (apresentação da mensagem de erro).

As literais apresentadas podem ser recuperadas também do arquivo strings.xml. Para isso, basta utilizar o comando:

```
getString( R.string.nome_da_literal)
```

Desta forma, o código dos métodos btLimparOnClick() e btCalcularOnClick() são apresentados conforme a **Listagem 6.5**.

Listagem 6.5. PrincipalActivity.java – Métodos para calcular e limpar usando o conteúdo de strings.xml

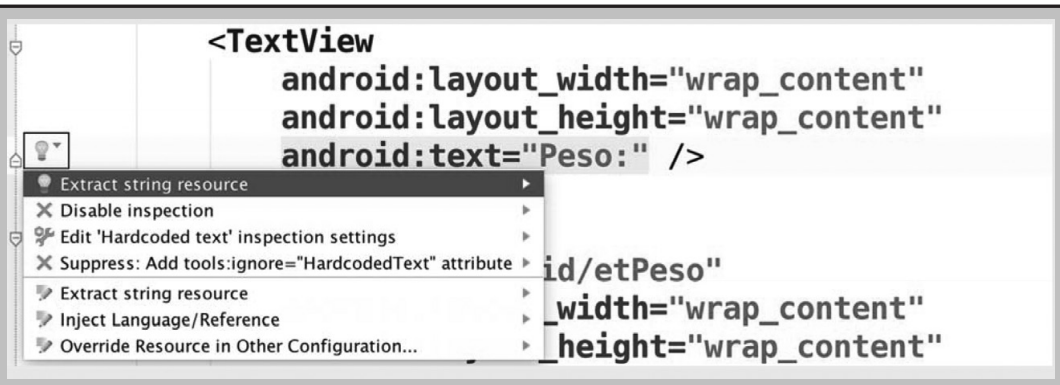
```
34.      public void btCalcularOnClick( View v ) {
35.          if (etPeso.getText().toString().equals("")) {
36.              Toast.makeText( this, getString( R.string.erropeso ),
Toast.LENGTH_LONG
37.                  ).show();
38.              etPeso.requestFocus();
39.              return;
40.          }
41.
42.          if (etAltura.getText().toString().equals("")) {
43.              Toast.makeText(this,getString( R.string.
erroaltura),Toast.LENGTH_LONG).show();45.
44.                  etAltura.requestFocus();
45.              return;
46.          }
47.
48.
49.          double peso = Double.parseDouble(etPeso.getText().
toString());
50.          double altura = Double.parseDouble(etAltura.getText().
toString());
51.
```

```
52.           double imc = peso / Math.pow(altura, 2);
53.
54.           tvResultado.setText(new DecimalFormat("0.00").
format(imc));
55.
56.       }// fim do método btCalcularOnClick
57.
58.       public void btLimparOnClick( View v ) {
59.           etPeso.setText("");
60.           etAltura.setText("");
61.           tvResultado.setText( getString( R.string.zeros) );
62.       }// fim do método btLimparOnClick
```

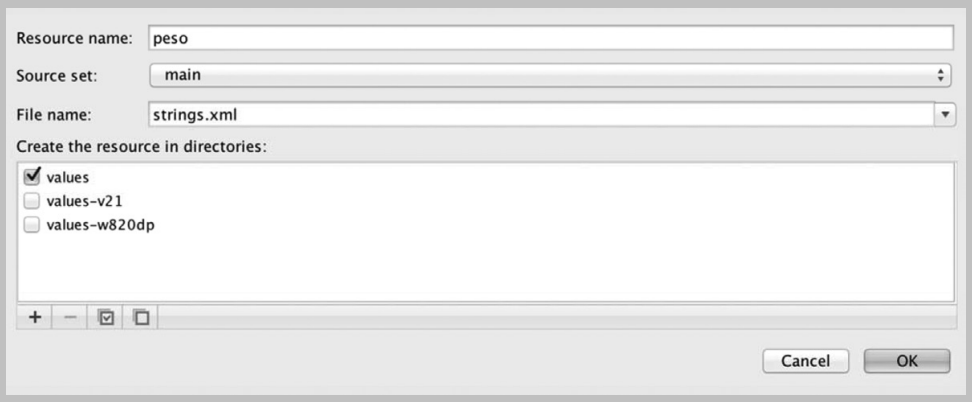
Desta forma, as literais utilizadas pelo código Java também são recuperadas no arquivo strings.xml, sendo este o repositório de literais do aplicativo.

Dica. Utilizando a IDE para incluir literais no arquivo strings.xml

Após o desenvolvimento de um aplicativo utilizando literais diretamente nos arquivos .xml e .java, o processo de movimentação dessas literais para o arquivo strings.xml é bastante trabalhoso, uma vez que devemos encontrar onde estamos utilizando as literais, acessar o arquivo strings.xml para adicionar uma entrada correspondente a estas, após, devemos retornar ao arquivo que faz uso da literal (.xml ou .java) e fazer referência a essa entrada. Quando o número de literais é pequeno, o processo é relativamente rápido, agora com muitas literais, o processo se torna bastante demorado. Para minimizar o problema, o Android Studio traz uma opção bastante interessante. Para adicionar uma literal ao arquivo strings.xml, basta encontrar a literal, deixar o cursor sobre ela e clicar na lâmpada que aparece do lado esquerdo, escolhendo a opção *Extract String Resource*, conforme a figura abaixo:



Na janela apresentada, o campo *Resource name* informa o nome da literal:



Por fim, a entrada é inserida no arquivo strings.xml e a referência para a literal na tela já é feita referenciando a entrada, tudo de forma automática.

O processo é o mesmo nos códigos-fontes Java (arquivos .java).

Internacionalizando as aplicações Android

Para internacionalizar o aplicativo desenvolvido até o momento, o primeiro passo é traduzir todas as literais para todos os idiomas desejados. Desta forma, na prática, deve-se ter uma versão do arquivo strings.xml para cada idioma desejado. Este processo é feito criando várias pastas values, diferenciando-as no sufixo (utiliza-se a sigla internacional do idioma, -pt para português, -en para inglês, -es para espanhol e assim por diante). Entretanto, de qualquer maneira, ainda deve permanecer uma pasta values sem sufixo, que será o idioma padrão do

aplicativo, assim, se o aplicativo for rodar em um device alemão e esse idioma não foi tratado, o conteúdo apresentado para o usuário será o armazenado na pasta values.

A IDE Android Studio trata as versões do arquivo strings.xml de uma forma muito interessante e para o desenvolvedor, as diferentes pastas de values (values, values-pt, values-en e values-es) ficam omitidas. Assim, para criar uma versão de strings.xml para o idioma inglês, por exemplo, basta que o programador clique com o botão direito na pasta *res* do projeto e selecione a opção *new – Android Resource File*...

Na tela apresentada, deve-se informar o nome do arquivo, strings.xml, o campo *Resource Type* deve estar marcado como *Values* e em *Available Qualifier*, escolhe-se *Language,* clicando no botão *>>*. Desta forma, será apresentada uma lista com todos os idiomas, assim como sua sigla. Para o exemplo, escolheremos *en:English*, conforme a **Figura 6.5**.

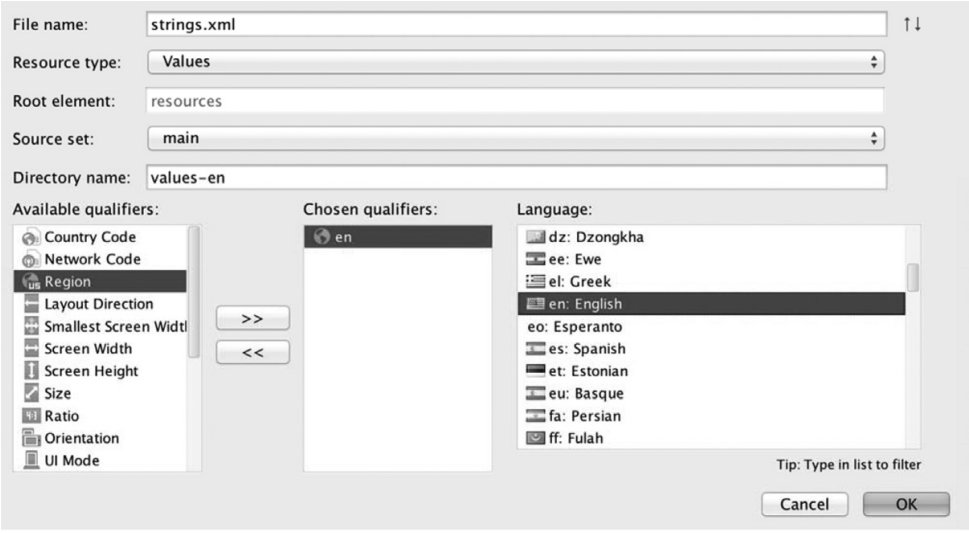


Figura 6.5. Criando um arquivo strings.xml para o idioma inglês.

O conteúdo do arquivo criado é apresentado na **Listagem 6.6**. Observem que o arquivo possui a mestra estrutura do arquivo modificado na **Listagem 6.3**. A única diferença está nas literais, que agora estão em inglês.

Listagem 6.6. strings.xml – Repositório de literais no idioma inglês armazenado na pasta values-en.

```
01.  <?xml version=”1.0” encoding=”utf-8”?>
02.  <resources>
03.
04.      <string name=”app_name”>BMI Calculate</string>
05.      <string name=”action_settings”> Settings </string>
06.      <string name=”altura”>Heigth:</string>
07.      <string name=”peso”>Weigth:</string>
08.      <string name=”imc”>BMI:</string>
09.      <string name=”zeros”>0.0</string>
10.      <string name=”calcular”>Calculate</string>
11.      <string name=”limpar”>Clear</string>
12.      <string name=”erroaltura”>Heigth field must be filled</string>
13.      <string name=”erropeso”>Weigth field must be filled</string>
14.
15.  </resources>
```

Após, para teste, criaremos também um arquivo de literais para o idioma espanhol – es:spanish, com o conteúdo apresentado na **Listagem 6.7.**

Listagem 6.7. strings.xml – Repositório de literais no idioma espanhol armazenado na pasta values-es.

```
01.  <?xml version=”1.0” encoding=”utf-8”?>
02.  <resources>
03.
04.      <string name=”app_name”>Calculo do IMC</string>
05.      <string name=”action_settings”> Ajustes </string>
06.      <string name=”altura”>Altura:</string>
07.      <string name=”peso”>Peso:</string>
08.      <string name=”imc”>IMC:</string>
09.      <string name=”zeros”>0.0</string>
10.      <string name=”calcular”>Calcular</string>
11.      <string name=”limpar”>Limpiar</string>
12.      <string name=”erroaltura”>Campo de altura debe ser llenado</
string>
13.      <string name=”erropeso”>Campo de peso debe ser llenado</
string>
14.
15.  </resources>
```

Assim, na pasta values do projeto, agora temos três versões do arquivo strings.xml, o primeiro default definido em português, uma ver-

são do arquivo em inglês e outra versão em espanhol, conforme a **Figura 6.6.**

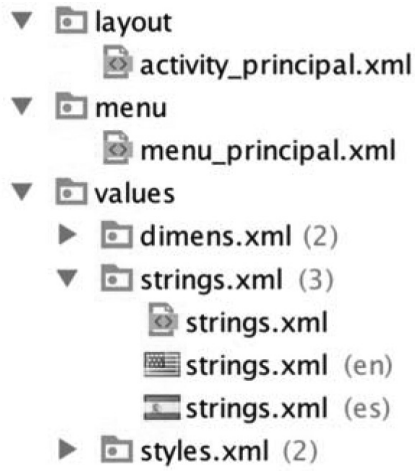


Figura 6.6. Versões do arquivo strings.xml.

Após, executando o aplicativo no emulador, é possível verificar que todas as literais da tela foram traduzidas para o inglês, já que este é o idioma padrão do emulador. Aos que forem executar em devices Android reais, é provável que as literais sejam apresentadas em português.

Para mudarmos o idioma do device/emulador, devemos acessar as configurações do dispositivo Android. Na lista de opções, deve-se escolher *Idioma e entrada*, conforme a **Figura 6.7**.

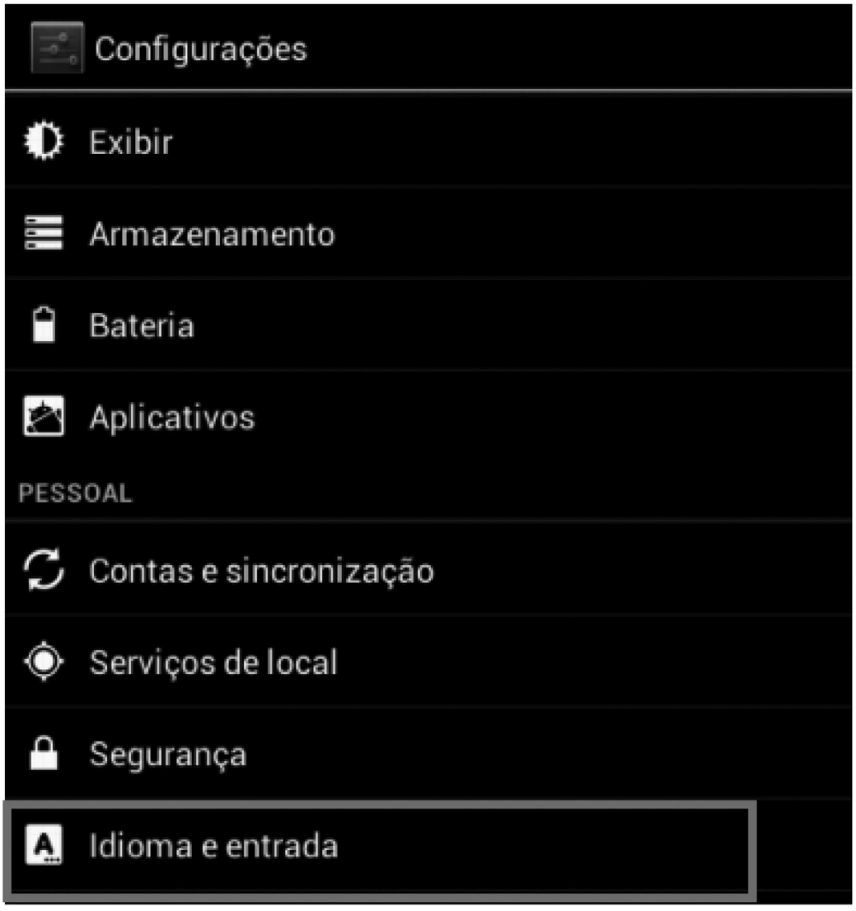


Figura 6.7. Escolhendo o idioma da aplicação.

Na tela apresentada, na opção *Selecionar Localização* (primeira opção da tela), pode-se escolher entre os idiomas disponíveis para o dispositivo Android. Se houver uma pasta values correspondente ao idioma (por exemplo, values-en para inglês, values-pt para português ou values-es para espanhol), o aplicativo será automaticamente traduzido, assim como todas as opções do smartphone.

Se não houver uma pasta values correspondente ao idioma selecionado, automaticamente o conteúdo do arquivo strings.xml da pasta values será carregado, sendo este o idioma default

Na **Figura 6.8**, é possível ver o aplicativo executado no idioma português (figura do lado esquerdo) e no idioma inglês (figura do lado direito).

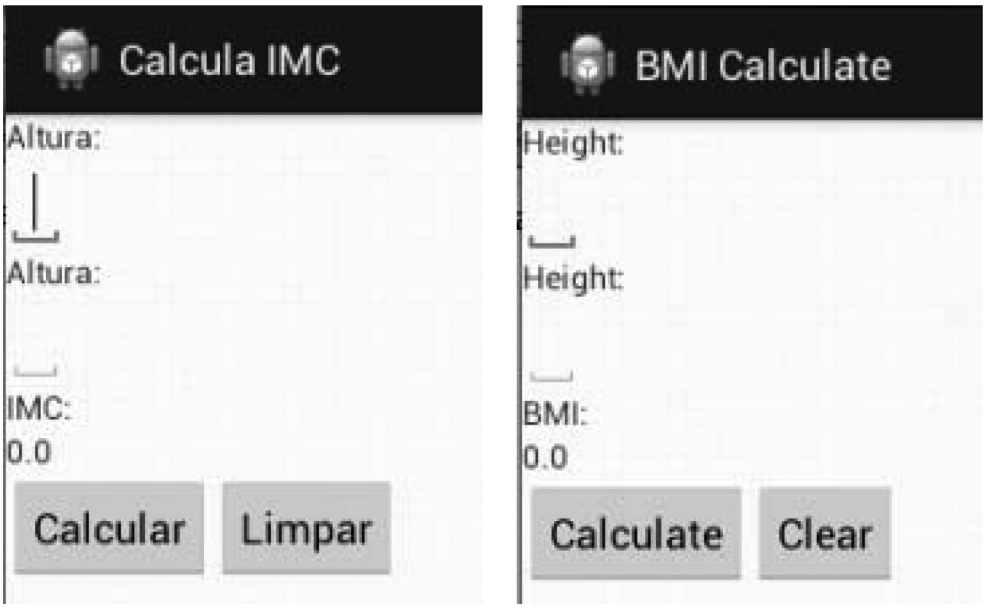


Figura 6.8. À esquerda, o layout da aplicação sendo executado em um dispositivo com o idioma português, à direita, o mesmo aplicativo sendo rodado em um dispositivo configurado com o idioma inglês.

Entretanto, para tratar a internacionalização do presente aplicativo, é necessário também personalizar a fórmula, uma vez que a maioria dos países trabalha com o sistema internacional de medidas (neste, o comprimento é medido em m e peso é medido em g), porém, alguns países, como os Estados Unidos, não utilizam esse padrão, preferindo polegadas para medir o comprimento e pounds para o peso. Assim, a fórmula do IMC para os americanos é um pouco diferente, conforme por apresentado na Figura 6.9:

$$BMI = \left(\frac{\text{weight in pounds} * 703}{\text{height in inches}^2} \right) \left(\text{lbs/inches}^2 \right)$$

Figura 6.9. Fórmula para IMC com pounds e polegadas

Para personalizar a lógica do botão *Calcular*, pode-se recuperar o idioma do dispositivo durante a execução utilizando o comando:

```
Locale.getDefault().getLanguage()
```

Assim, a lógica do botão *Calcular* fica como apresentada na **Listagem 6.8**.

Listagem 6.8. btCalcularOnClick() – Método que calcula o IMC tratando a internacionalização.

```
01.        private void btCalcularOnClick() {
02.             if ( etPeso.getText().toString().equals( "" ) ) {
03.               Toast.makeText( this, getString( R.string.erropeso ),
Toast.LENGTH_LONG ).show();
04.               etPeso.requestFocus();
05.               return;
06.             }
07.
08.             if ( etAltura.getText().toString().equals( "" ) ) {
09.               Toast.makeText( this, getString( R.string.erroaltura
), Toast.LENGTH_LONG ).show();
10.               etAltura.requestFocus();
11.               return;
12.             }
13.
14.             double peso = Double.parseDouble( etPeso.getText().
toString() );
15.             double altura = Double.parseDouble( etAltura.getText().
toString() );
16.
17.             double imc = 0;
18.
19.             if ( Locale.getDefault().getLanguage().equals( "en" ) )
{
20.                     imc = peso / Math.pow( altura, 2 );
21.             } else {
22.                     imc = peso * 703 / Math.pow( altura, 2 );
23.             }
24.
25.             tvResult.setText( new DecimalFormat( "0.00" ).format(
imc ) );
26.
27.        } //fim do método btCalcularOnClick
```

O código apresentado na **Listagem 6.8** diferencia do tradicional pela recuperação durante a execução do idioma do device Android, assim, na linha 19, é recuperado o idioma e comparado com o java.util.Locale (en para inglês, pt para português, es para espanhol, e assim por diante). A lógica foi tratada de forma personalizada para o idioma inglês; para qualquer outro idioma, é utilizado o sistema internacional de medidas.

Por fim, pode-se também personalizar a máscara dos campos, tais como, vírgula como separador decimal para o aplicativo rodando no Brasil ou ponto para o aplicativo rodando nos Estados Unidos e para isso, a linha 25 da **Listagem 6.8** deve ser substituída pelo código da **Listagem 6.9**.

Listagem 6.9. btCalcularOnClick( ) – Internacionalizando a formação dos números.

```
01.        NumberFormat nf = NumberFormat.getNumberInstance( Locale.
getDefault() );
02.        DecimalFormat df = (DecimalFormat) nf;
03.
04.        tvResultado.setText( df.format( imc ) );
```

O processo de internacionalização da formatação dos números utiliza a classe java.text.NumberFormat, mas também pode ser usada por outras classes, como, por exemplo, java.text.SimpleDateFormat, que permite a formatação de datas e horas.

Concluindo...

Neste capítulo, foram apresentados os principais conceitos da internacionalização e do uso de literais nos aplicativos Android. Recurso este que permite a um aplicativo rodar em diferentes idiomas, sem a necessidade de mudar o código-fonte ou recompilar.

Embora seja um processo um pouco lento e chato, o uso de um repositório de literais é justificado, uma vez que facilita as revisões ortográficas e gramaticais dos textos presentes no aplicativo, padroniza a escrita de palavras e expressões, além de facilitar o processo de tradução do aplicativo.

O capítulo foi além e também apresentou a customização do aplicativo em relação a suas funcionalidades, sendo recuperado durante a execução o idioma do celular, o que permitiu tratar diferentes funcionalidades.

Exercícios de fixação do capítulo

Exercício 1 – Internacionalizando uma aplicação

Realize uma pesquisa mais aprofundada sobre I18N (Internacionalization), verificando, além do idioma e das fórmulas matemáticas, o que mais pode ser internacionalizado (formato de navegação do aplicativo, figuras, cores, informações temporais etc.)

Exercício 2 – Tratando a internacionalização para o alemão e o francês

Aprimore o aplicativo desenvolvido ao longo deste capítulo para que também sejam considerados os idiomas alemão e francês.

Exercício 3 – Aplicativo para o cálculo do combustível mais barato

Altere o exercício 1 do Capítulo 4 (a tela segue abaixo) para fazer uso exclusivo das literais, tanto nos dados da tela como nas mensagens informativas.