

# Dominando

# ANDROID

```
package com.androidexample.webview;
```

```
import android.app.Activity;  
import android.app.ProgressDialog;  
import android.os.Bundle;  
import android.webkit.WebView;  
import android.webkit.WebViewClient;
```

```
public class ShowWebView extends Activity {
```

```
    //private Button button;  
    private WebView webView;  
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.show_web_view);
```

```
        //Get webview  
        webView = (WebView) findViewById(R.id.webView1);
```

```
        startWebView("http://www.androidexample.com");  
    }  
    view/log);
```

```
    private void startWebView(String url) {
```

```
        //Create new webview Object  
        //When opening a url or link
```

```
        webView.setWebViewClient(new WebViewClient());  
        ProgressDialog progressDialog;
```

```
        //If you will not use the progressDialog  
        //you can use webView.loadUrl(url) not in  
        webView
```

```
        public boolean shouldOverrideUrlLoading(WebView  
            view, String url) {  
            view.loadUrl(url);  
            return true;  
        }
```

# Introdução ao Android

Um guia introdutório sobre aplicações Android

Daniel Schmitz

Esse livro está à venda em <http://leanpub.com/livro-dominando-android>

Essa versão foi publicada em 2014-07-23



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

©2014 Daniel Schmitz

# **Outras Obras De Daniel Schmitz**

The MarkDown Cheat Sheet

AngularJS na prática

Bootstrap 3

Dominando Action Script 3

# Conteúdo

<b>Capítulo 1 - Introdução</b> . . . . .	<b>1</b>
O que é Android? . . . . .	1
Por que Android cresceu tanto? . . . . .	1
Recursos para o desenvolvedor . . . . .	2
<b>Capítulo 2 - Preparando o ambiente android</b> . . . . .	<b>3</b>
Antes de tudo, o Java . . . . .	3
Instalando o Android . . . . .	4
Configurando o Android . . . . .	4
Criando um dispositivo mobile virtual . . . . .	6
Configurando um dispositivo real . . . . .	8
<b>Capítulo 3 - Criando o HelloWorld</b> . . . . .	<b>10</b>
Conhecendo um pouco mais o projeto . . . . .	13
Executando o HelloWorld . . . . .	15
<b>Capítulo 4 - Conceitos fundamentais</b> . . . . .	<b>17</b>
Arquitetura do Android . . . . .	17
Ciclo de vida de uma aplicação Android . . . . .	20
Intents . . . . .	22
Services . . . . .	22
Permissões . . . . .	22
Resources . . . . .	24
Métricas . . . . .	24
<b>Capítulo 5 - Além do HelloWorld</b> . . . . .	<b>26</b>
Criando o projeto . . . . .	26
Compreendendo layouts . . . . .	26
Alterando um layout . . . . .	28
Inserindo Botões . . . . .	29
Utilizando Resources alternativos . . . . .	37
Alterando o Layout da tela com Resources . . . . .	41
Alterando o idioma com Resources . . . . .	43
Criando a tela Sobre . . . . .	44
Preparando o botão Sobre . . . . .	47
Criando o intent . . . . .	50
Aplicando um estilo diferente na tela About . . . . .	51
Implementando o botão Sair . . . . .	52

## CONTEÚDO

Debug is on the table . . . . .	52
<b>Capítulo 6 - Configurações personalizadas . . . . .</b>	<b>53</b>
Criando a tela de opções . . . . .	53
Utilizando o assistente . . . . .	53
Como acessar as opções . . . . .	57

# **Capítulo 1 - Introdução**

## **O que é Android?**

Android é uma a plataforma open source para dispositivos mobile. Com Android é possível executar aplicações simples ou complexas diretamente do seu celular ou tablet. Devido as restrições tanto de tamanho de tela quanto de performance, sem falar na variedade de dispositivos existentes, não é tão simples assim criar um aplicação para estes dispositivos sem um framework que gerencie todo este processo, e o Android cumpre exatamente este papel.

Com o Android temos uma infinidade de facilidades que só ele pode oferecer, fazendo que o sistema Android seja a cada dia mais utilizado entre os dispositivos existentes. Android também é o sistema operacional mais utilizado no mundo, detentor de 81% do mercado de smartphones atual.

## **Por que Android cresceu tanto?**

Existem diversas carreirísticas que tornaram o Android um sistema tão popular no mundo todos. Pessoalmente, eu penso que o mercado foi dominado por ele devido a compatibilidade com a maioria dos smartphones existentes, mesmo aqueles mais simples. É claro que existem diversos outros fatores, nos quais vamos explicar a seguir.

### **Um ambiente realmente open source**

O sistema operacional Android é na verdade um port do Linux, completamente open source e sem restrições que possam atrapalhar a sua evolução. Desenvolvedores podem usar esse “poder” para customizar suas distribuições e se sentem mais seguros com esta liberdade. Você pode criar uma aplicação do zero com um suporte quase que infinito em termos de IDE, SDK, API, tutoriais e vídeos sem a necessidade de um investimento financeiro agressivo.

### **Uma arquitetura baseada em componentes**

Todo o desenvolvimento Android é baseado em componentes, de forma que você tem o poder de reutilizar e alterar diversas partes não somente da sua aplicação, mas também de particularidades do dispositivo. Você pode, por exemplo, criar uma nova aplicação de calendário e substituí-la pela original existente no smartphone, ou criar uma aplicação de entrada (home screen) completa.

### **Dezenas de funcionalidades prontas para uso**

O Android permite que se utilize todos os recursos principais dos smartphones, além de possuir diversas funcionalidades que irão facilitar o desenvolvimento da sua aplicação. Por exemplo, temos um completo gerenciamento de dados através de SQL, acesso a dispositivos como a câmera do dispositivo, acesso ao GPS, aos contatos do smartphone, e muito

mais. Além disso, existe uma complexa arquitetura destinada a prover um gerenciamento automático do ciclo de vida de uma aplicação. Por exemplo, é fácil programar uma situação na qual o dispositivo recebe uma ligação enquanto a aplicação está sendo executada.

### Gráficos

Android possui amplo suporte a gráficos 2D e 3D, o que permite a criação de jogos e aplicações comerciais sofisticadas. Esse suporte inclui OpenGL, codecs para execução de vídeo e som, entre diversos outros. Android também oferece suporte para a criação das mais variadas interfaces nos diferentes tamanhos de tela e resolução de um dispositivo.

### Portabilidade

Ainda vamos explicar sobre a “máquina virtual Android”, mas saiba que isso garante que a sua aplicação irá funcionar nos mais diversificados dispositivos existentes no mercado, incluindo os que não foram criados ainda. Com isso, as suas estimativas de venda crescem na medida que novos aparelhos são criados. E não importa se o dispositivo é top de linha ou mediado, a sua aplicação vai funcionar.

## Recursos para o desenvolvedor

O Android é toda uma plataforma de desenvolvimento de software. Tudo que você precisa para desenvolver em Android está nele, mas precisamente em um termo que chamamos de ‘Android SDK’, ou seja, ‘Android Software Development Kit’. Iremos aprender a instalá-lo no próximo capítulo.

Outro recurso importante é o site oficial dos desenvolvedores para Android, localizado no endereço [d.android.com](http://d.android.com)<sup>1</sup>, onde você encontra as principais novidades da plataforma, bem como seus recursos. Neste site, ao acessar o menu ‘Develop’, você tem a sua disposição toda a documentação oficial do Android para consulta.

---

<sup>1</sup><http://d.android.com>

# Capítulo 2 - Preparando o ambiente android

Os usuários “normais” do android não precisam instalar absolutamente nada para começar a utilizar a sua aplicação android, bastando apenas ter o sistema android no dispositivo. Para nós desenvolvedores, precisamos instalar um conjunto de ferramentas que servem não apenas para escrever código, mas também para emular a aplicação em diversos dispositivos, debugar a mesma, editá-la visualmente, etc.

## Antes de tudo, o Java

O Java é a base de tudo para o nosso desenvolvimento. Ele é a primeira ferramenta que você deve instalar. Se você já trabalha com desenvolvimento Java possivelmente já tem ele no sistema, mas caso não tenha, devemos instalá-lo.

### Como saber se tenho o Java

Você deve abrir uma janela de console no seu sistema. No Windows, clique em **Iniciar** e depois em **Executar**. Digite **cmd** e um console irá aparecer. Neste console, digite:

```
1 java -version
```

O resultado deste comando irá mostrar se o java está instalado.

```
1 C:\Users\daniel>java -version
2 java version "1.7.0_07"
3 Java(TM) SE Runtime Environment (build 1.7.0_07-b11)
4 Java HotSpot(TM) 64-Bit Server VM (build 23.3-b01, mixed mode)
```

Se aparecer alguma mensagem do tipo **java não é reconhecido como um comando**, então você deverá instalar o Java.

### Java SDK ou Java JRE ?

Existem duas versões bem distintas do Java. O **SDK**, que já conhecemos como **Software Development Kit**, é o kit de desenvolvimento que precisamos para escrever aplicações em Java e, consequentemente, em android. O **Java JRE** é o **Runtime Environment**, ou basicamente o que “roda” o Java. Um usuário comum que usa aplicações Java (como um autenticador do homebank) tem necessariamente o JRE instalado em seu sistema, pois ele precisa apenas executar a aplicação, e não desenvolvê-la. Geralmente esta instalação encontra-se no endereço [java.com/getjava<sup>2</sup>](http://java.com/getjava).

---

<sup>2</sup><http://java.com/getjava>

## Instalando o JDK

Nosso foco aqui é desenvolver em Java, então apenas o JRE não é suficiente. Precisamos instalar o Java SDK, e ele é encontrado neste endereço:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html<sup>3</sup>>

Ao acessar o endereço, clique no botão download onde está escrito Java Platform (JDK) 7u51.

Nesta obra estaremos utilizando o Java SDK versão 7, update 51. Talvez você possa estar em uma versão de update superior quando for acessar a página de download.

Após clicar em download, uma nova página é carregada e deve-se aceitar a licença de uso e realizar o download do JDK de acordo com o seu sistema operacional. Após realizar o download, execute o arquivo e instale o JDK no sistema.

## Instalando o Android

Tudo que você precisa para desenvolvimento Android está em uma única instalação. Tudo que você precisa fazer é acessar [d.android.com<sup>4</sup>](http://d.android.com<sup>4</sup>) e clicar no botão Get the SDK. Na página Get the Android SDK clique no botão Download the SDK e faça o download do arquivo zipado.

Após terminar o download, descompacte o arquivo em algum local do sistema (usaremos a raiz c:\ ) e, ao acessar C:\adt-bundle-windows-x86\_64-20131030, verifique nele se existem as pastas eclipse, sdk e o arquivo SDK Manager .exe. Neste momento, você pode renomear a pasta adt-bundle-windows-x86\_64-20131030 para algo mais simples, como por exemplo adt, então teremos apenas c:\adt para acessar. Neste momento, o seu kit de desenvolvimento Android está pronto para uso.

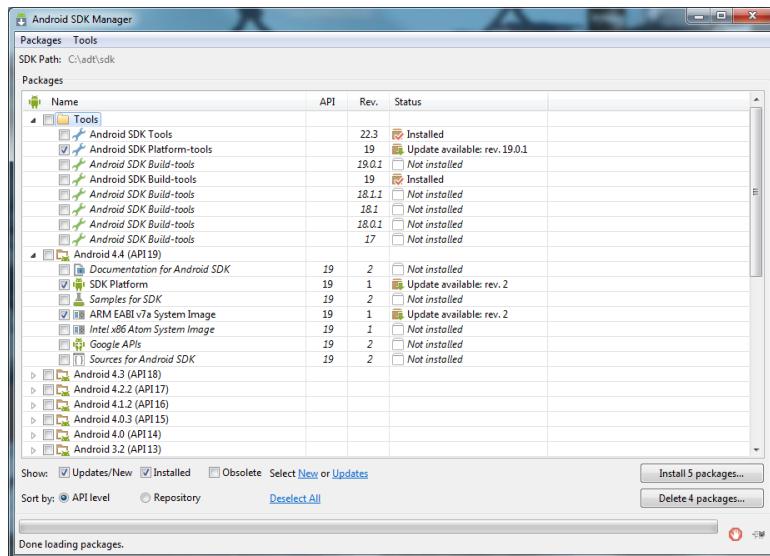
## Configurando o Android

### SDK Manager

Este gerenciador é utilizado para que você possa verificar quais as versões do Android SDK você tem, além de poder instalar novas versões e funcionalidades.

<sup>3</sup><http://www.oracle.com/technetwork/java/javase/downloads/index.html>

<sup>4</sup><http://d.android.com>



SDK manager

## Eclipse

A IDE eclipse não nasceu com o Android. Ela é uma IDE totalmente funcional e amplamente extensível, usada há vários anos em diversas linguagens. Por ser amplamente customizável, o Google a adotou como IDE padrão.

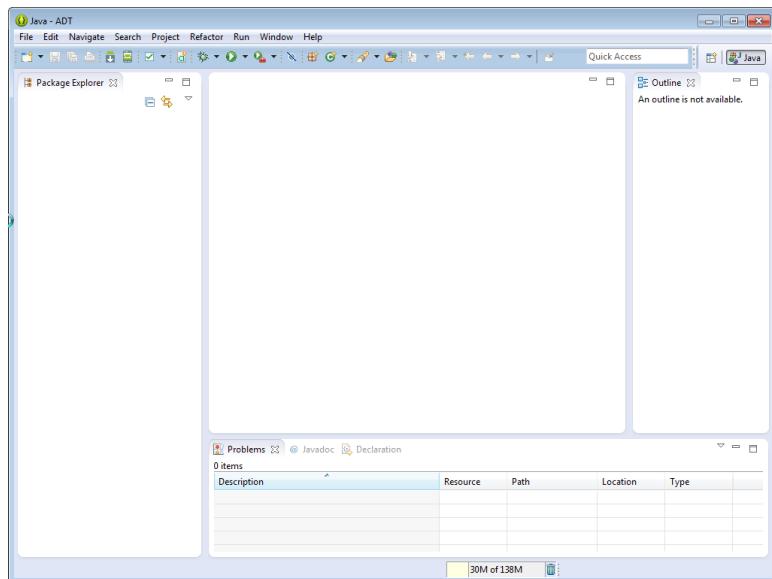
Geralmente o eclipse não é uma ferramenta instalável. Em teoria, para executar o eclipse basta abrir o executável, que no nosso caso está em C:\adt\ eclipse\ eclipse.exe. Recomendo inclusive que crie um atalho deste executável na área de trabalho.

O eclipse que vem junto com o Android está totalmente customizado para ele. Ao executá-lo pela primeira vez, você verá a tela **Workspace Launcher**, no qual é configurada uma pasta para as suas configurações e projetos. Deixe o padrão e marque o item **Use this as the default....** para que esta janela não apareça novamente.

### O que é workspace?

No eclipse, o workspace é uma área de trabalho que guarda projetos e configurações. Ele é mais usado quando você possui diversos projetos e deseja separá-los por área. Por exemplo, você pode ter 3 projetos em Java de uma empresa X, e mais 4 projetos de outra empresa Y, e você deseja separar isso de forma que os projetos não se misturem. Para alterar entre esses projetos, você acessa o menu **File>Change workspace** no próprio eclipse.

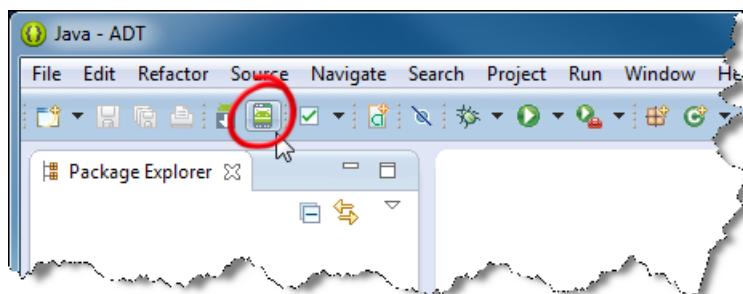
Finalmente o Android ADT é carregado. Na primeira tela, temos uma aba inicial com algumas informações, você pode fechá-la para que a IDE seja exibida, de acordo com a imagem a seguir.



Eclipse ADT

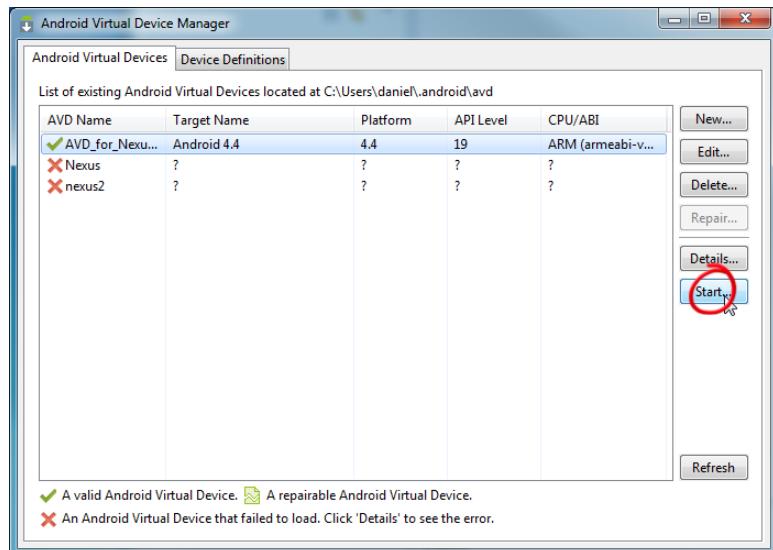
## Criando um dispositivo mobile virtual

Como vamos criar aplicações para dispositivos mobile, precisamos de um para testa. Caso você não tenha, então deve utilizar o Android Virtual Device Manager, que é acessado através do eclipse, de acordo com o ícone exibido a seguir.



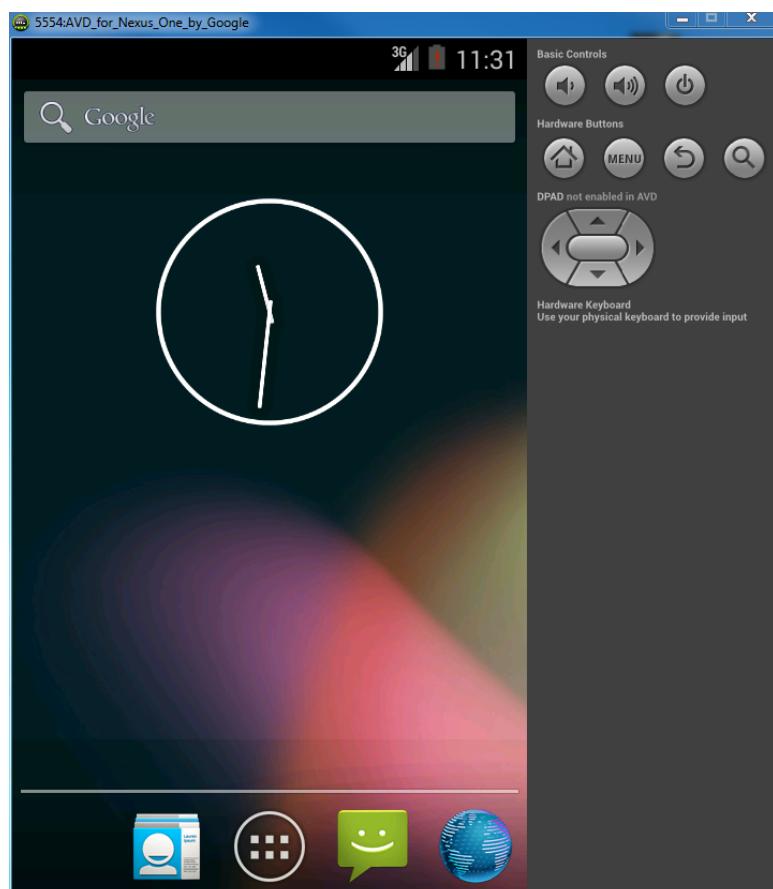
Android Virtual Device Manager

O AVD é responsável em criar dispositivos dos mais variados tipos e tamanhos. Para facilitar o nosso trabalho, acessa a aba **Device Definitions**, que possui alguns devices conhecidos, e escolha um, como por exemplo **Nexus One**. Após selecioná-lo, clique no botão **Create AVD** e em seguida, clique em **Ok**. Uma nova device será criada, de acordo com a imagem a seguir. Para executá-la, selecione e clique no botão **Start** e depois em **Launch**.



Iniciando um device

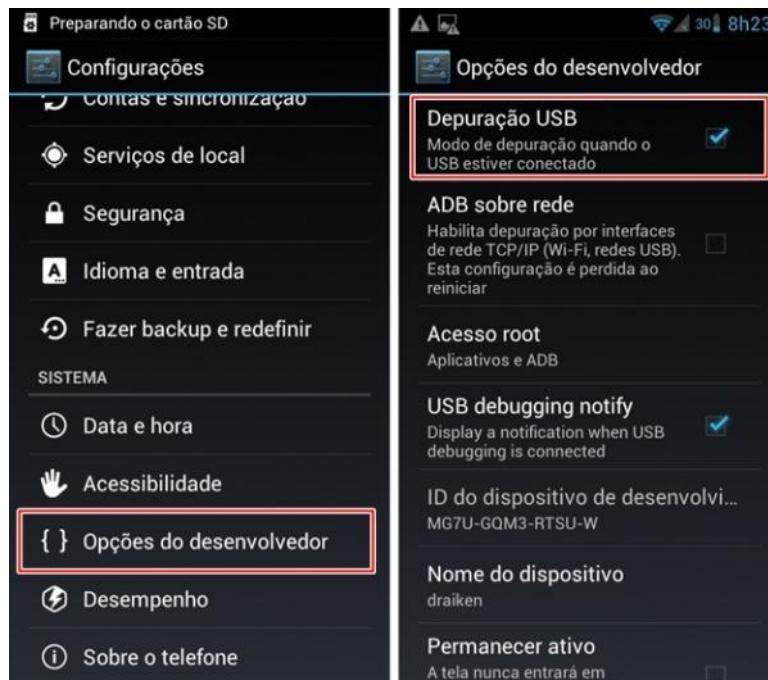
Agora é a hora de pegar um café e esperar o dispositivo carregar. Ele demora mesmo, é normal. Lembre-se que, quando estiver programando a sua aplicação, não é necessário reiniciar o dispositivo. Deixe-o sempre iniciado para poupar tempo.



Device devidamente carregado

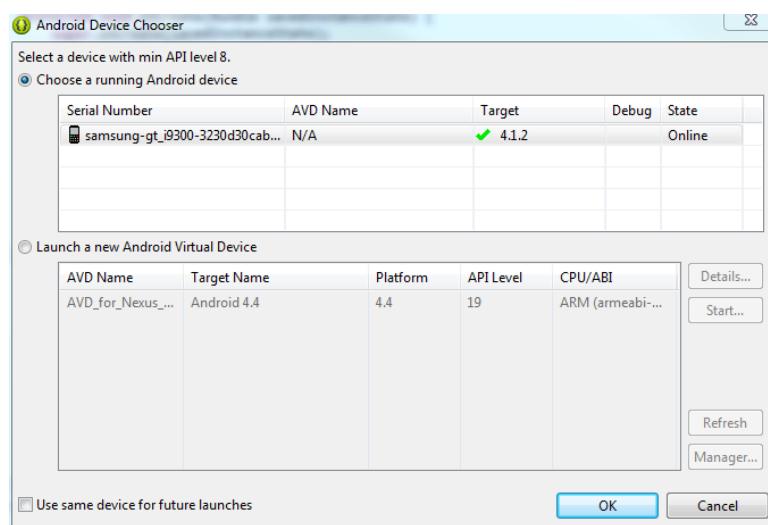
## Configurando um dispositivo real

Se você tiver um dispositivo mobile real, poderá testar as suas aplicações nele pelo próprio. Para isso, você deverá, na configuração do dispositivo, habilitar o modo de depuração USB, acessando ‘Configurações > Opções do Desenvolvedor’, de acordo com a imagem a seguir:



Habilitando a depuração USB

Com o USB ligado no dispositivo, o eclipse será capaz de carregar as configurações dele e, quando executar a aplicação, ela será carregada no seu dispositivo real, conforme a figura a seguir.



Executando no dispositivo real

Lembre-se de fechar o emulador virtual para que a aplicação seja executada no dispositivo real.

Caso o dispositivo real não tenha aparecido na lista, significa que houve algum problema de Driver e isto está relacionado ao sistema operacional. Baixe o driver do fabricante do seu dispositivo mobile e tente novamente.

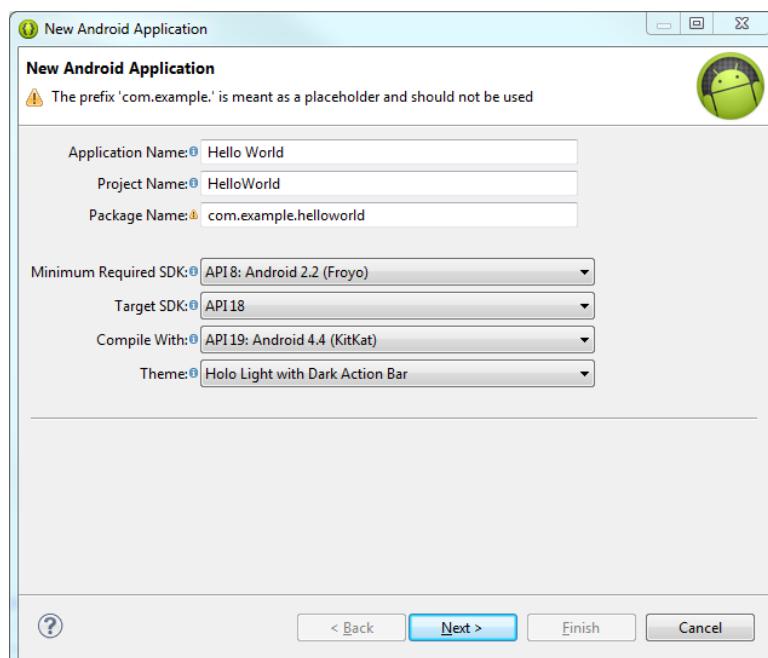
# Capítulo 3 - Criando o HelloWorld

Neste capítulo veremos como usar o ADT para criar uma aplicação muito simples, a já conhecida HelloWorld. Criaremos esta aplicação através do assistente do eclipse, de forma que, inicialmente, não teremos programar nada.

Com o ADT aberto, navegue até o menu File > New > Android Application Project. Surge um assistente com diversos passos, no qual vamos esclarecer cada um deles.

Na primeira tela, definimos o nome da aplicação, o nome do projeto e o nome do pacote no qual o projeto será criado. Nome da aplicação será o nome que aparecerá no dispositivo, bem como na loja do Google (Play Store). Nome do projeto é o nome que aparece no projeto do eclipse, geralmente sem espaços ou caracteres especiais. Já em nome do pacote, inserimos um identificador que deve ser único globalmente. Para facilitar a criação deste nome, usa-se uma convenção que é chamada de domínio reverso, que é simplesmente reverter o nome de domínio da sua empresa ou site na internet. Por exemplo, se o seu site é www.firulagames.com.br, e você está criando um jogo chamado “Campo Minado”, o nome do pacote será br . com . firulagames . campominado.

Até este momento temos a seguinte configuração:



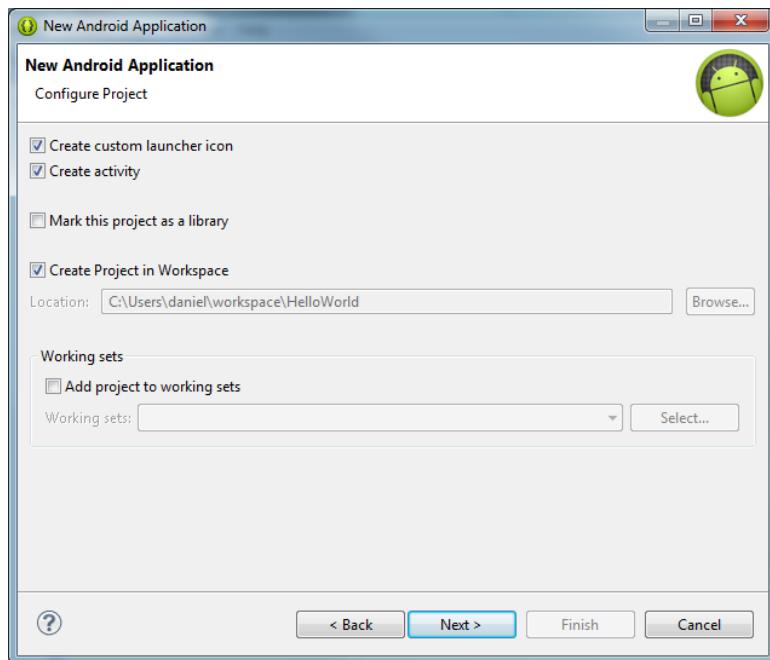
Executando no dispositivo real

No próximo campo temos a configuração Minimum Required SDK. Esta configuração irá configurar qual a versão mínima do Android para que a aplicação seja executada. Aqui é preciso ter um cuidado triplicado quando formos criar uma aplicação real. Você não poderá escolher uma versão muito baixa, pois irá perder diversas funcionalidades extras que as novas versões trazem, mas também não poderá escolher uma versão muito alta, senão poucos dispositivos estarão disponíveis para usar ou comprar a sua aplicação.

Atualmente, a versão mais básica é a 2.2 (Froyo), isso aqui no Brasil onde os dispositivos são um pouco mais atrasados em relação ao exterior. Muitos dispositivos mais baratos, como o Galaxy Y, usam esta versão. Se você deseja atingir esse público, deve selecionar esta versão. Para celulares um pouco mais caros, como o Galaxy S3, pode-se escolher a versão 4.1 (Jelly bean), mas esteja ciente que quem possuir uma versão anterior não poderá instalar a sua aplicação. Geralmente, se for desenvolver uma aplicação comercial, use a versão 2.2. Se for criar um jogo ou uma app com recursos gráficos avançados, deverá escolher a versão 4.1.

No próximo campo, Target SDK, você escolhe a versão de API na qual estaria mais apto a trabalhar. Não é porque você escolheu a versão 2.2 que deve usar a sua API, você pode usar a API da versão 4 (claro que com algumas limitações). Neste momento, é melhor deixar a maior versão estável disponível. O Target SDK também diz ao Android que esta é a versão mais testada e que, logicamente, não possui erros relacionados a API do android. Nesta obra utilizaremos a API 18.

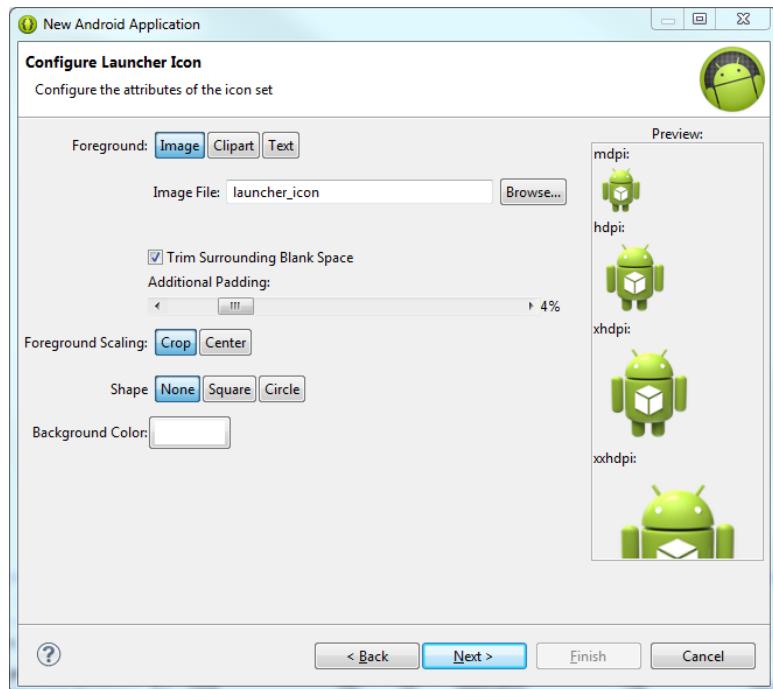
No próximo campo, Compile With, defina em qual API a sua aplicação será compilada. Geralmente deixamos a maior versão possível, no caso desta obra, a API 19. O próximo campo, Theme, indica o tema que será utilizado para criar a aplicação. Deixe o padrão e clique no botão Next. O resultado desta primeira tela é exibido a seguir:



Criando um novo projeto

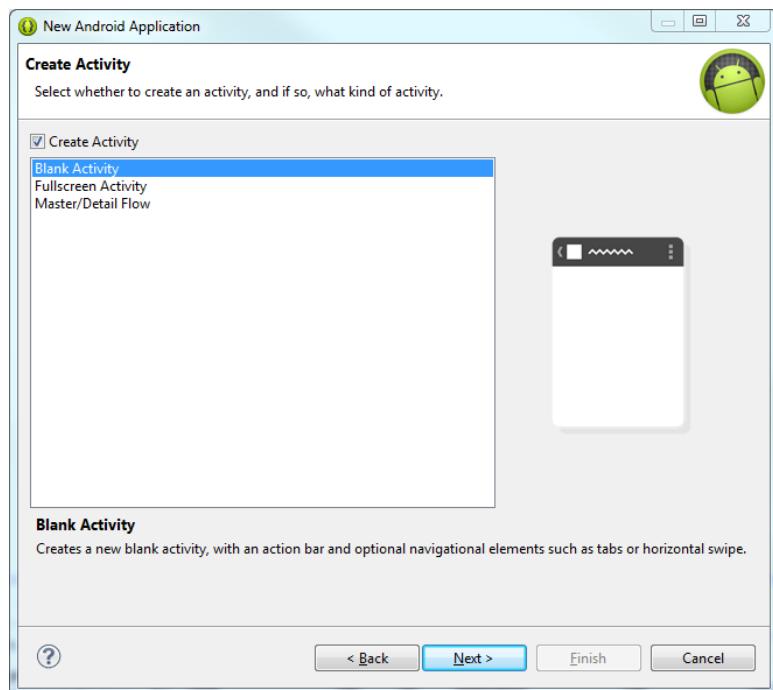
A próxima tela é configurada mais algumas opções, como por exemplo a criação do botão da aplicação e a criação de um activity (iremos explicar activity mais a diante).

O campo **Mark this project as a library** irá formatar o projeto como uma biblioteca, ou seja, não haverá telas ou informações visuais sobre a aplicação. Use essa opção para dividir a sua aplicação em camadas. Por enquanto, deixe desmarcado. O workspace é o lugar onde os arquivos serão criados, deixe o padrão. No campo Working Sets, deixe desmarcado, já que não queremos adicionar o projeto a um working set existente. Clique no botão Next para exibir a próxima tela do assistente, conforme a figura a seguir.



Criando um novo projeto

Nesta tela, definimos o ícone da aplicação. Você pode escolher um ícone existente ou adicionar ou clipart ou até mesmo um texto. Deixaremos com você esta parte. Clique em Next para visualizar a próxima tela do assistente, exibido a seguir:

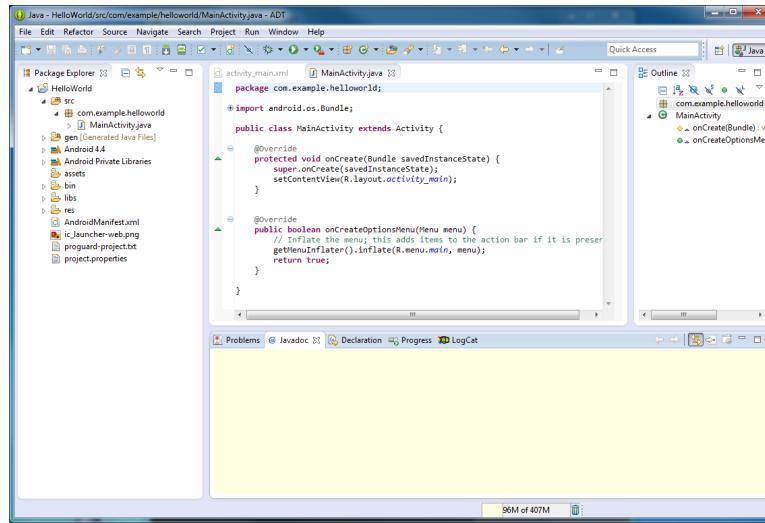


Criando um novo projeto

Nesta parte do assistente, definimos como será a tela inicial da aplicação. Cada tela de uma aplicação é chamada de Activity (não vamos traduzir para o português). Veremos com mais detalhes sobre activities no próximo capítulo. Como queremos dar um título a aplicação e

escrever um texto, deixaremos no padrão com um Blank Activity. Clique em Next para finalizar as configurações deste Activity, deixe tudo com o padrão e clique em Finish.

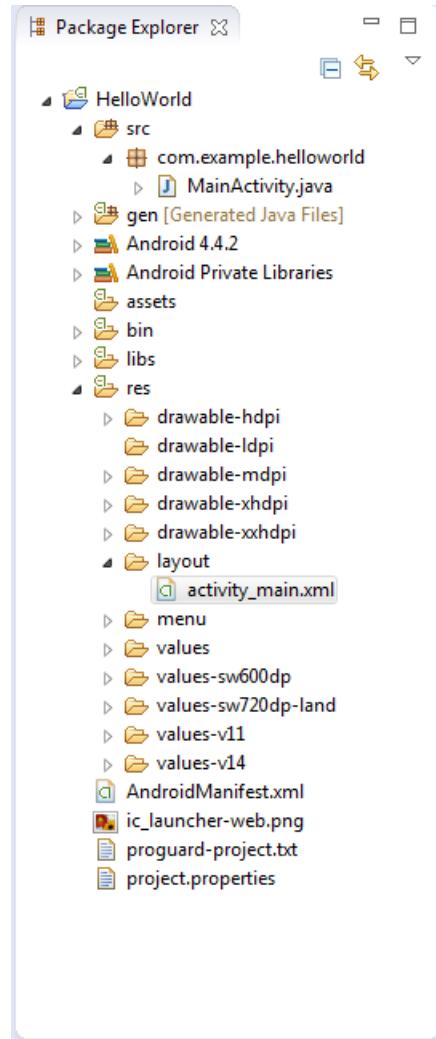
Como veremos na imagem a seguir, o assistente criou todos arquivos necessários para que a sua aplicação esteja pronta para uso.



Projeto criado no eclipse ADT

## Conhecendo um pouco mais o projeto

Vamos dar uma olhada na aba Package Explorer, onde encontram os arquivos desta aplicação.



Package Explorer

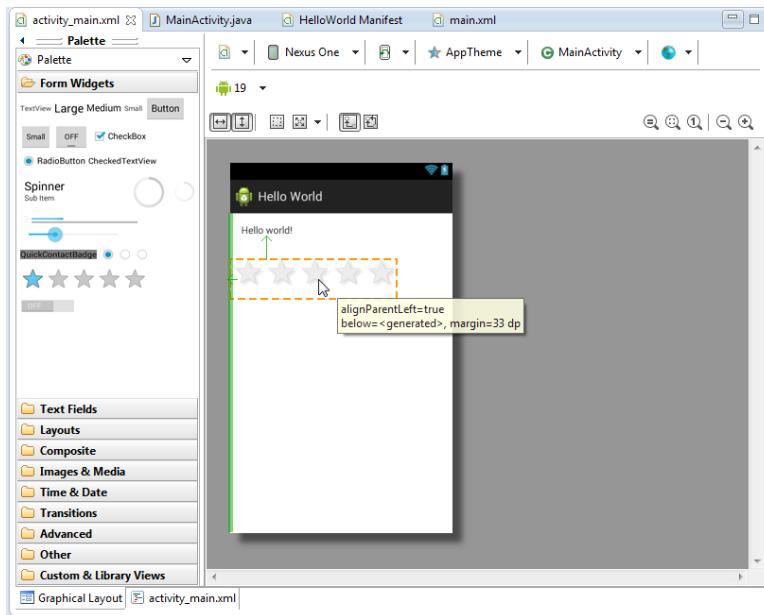
De todos os arquivos do projeto, vamos destacar 2 deles que são, nesse primeiro momento, os mais importantes.

#### **src/com.example.helloworld/MainActivity.java**

Já vimos que uma Activity pode representar uma tela da aplicação, e veremos mais detalhes no próximo capítulo. Este arquivo java é responsável em gerenciar a Activity inicial carregando o layout e definido diversas configurações da mesma.

#### **res/layout/activity\_main.xml**

Este arquivo XML é responsável em desenhar a tela da aplicação. Se você abrir o arquivo no eclipse adt, verá um editor visual (figura a seguir), onde é possível arrastar componentes para o editor. Pode-se abrir o arquivo no modo XML também, para visualizar o seu conteúdo.

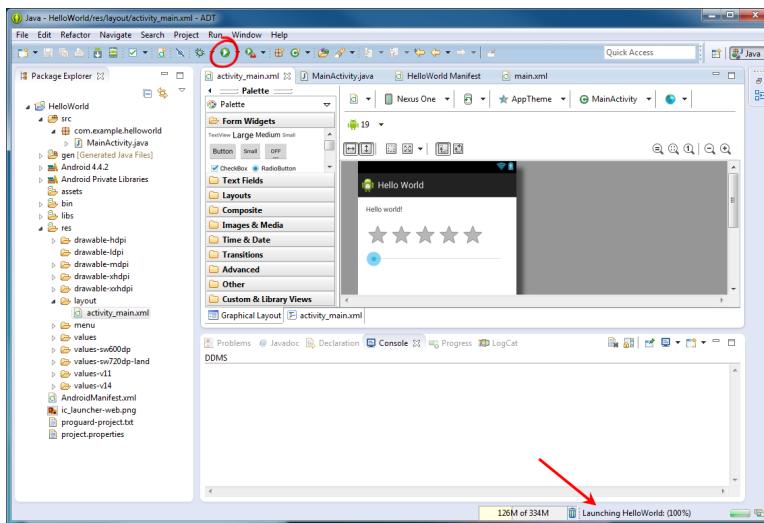


Editor visual do Layout

Experimente adicionar alguns componentes na tela e testar a aplicação.

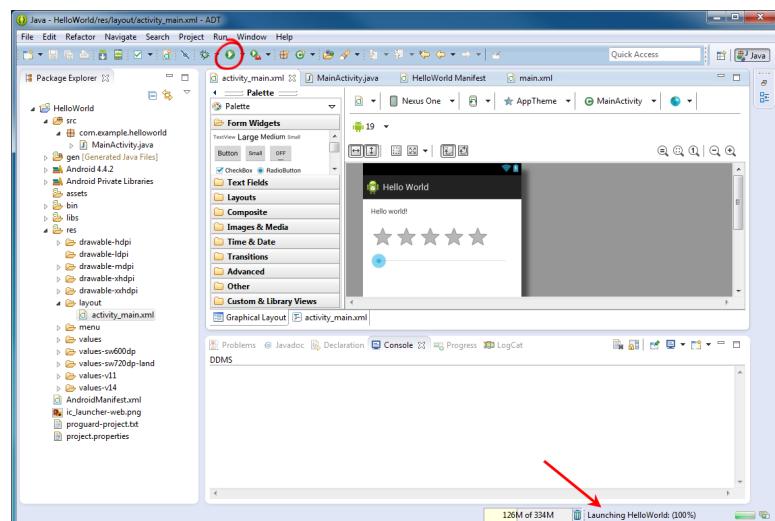
## Executando o HelloWorld

Para executar a aplicação, você deve clicar no botão verde Run, localizado na barra de botões. Perceba que uma mensagem de status aparece na barra inferior do eclipse, indicando que a aplicação está sendo inicializada.



Executando a aplicação Hello World

Lembre-se que o dispositivo deve estar emulado ou conectado via USB para que a aplicação seja carregada. No emulador, teremos o seguinte resultado:



Aplicação Hello World

# Capítulo 4 - Conceitos fundamentais

Neste capítulo estaremos esclarecendo os principais conceitos que envolvem o desenvolvimento Android. É importante ler este capítulo com calma e entender cada item, de forma que quando reapresentá-los na prática, você possa compreender do o que está fazendo.

Alguns conceitos já são conhecidos pela maioria, como por exemplo o kernel do linux, o que é OpenGL, SQL etc. Estes conceitos serão apenas abordados de forma bastante sucinta. Nosso foco é compreender os novos conceitos apresentados pelo Android.

## Nota do autor

Gostaria de compartilhar um fato que aconteceu há bastante tempo, em 2006 mais precisamente. Naquela época, o então Macromedia Flex entrava no mercado e ganhava uma notoriedade incrível. Ainda não havia conteúdo aqui no Brasil sobre isso, e como eu gostei da tecnologia desde o início, resolvi investir.

Só que, ao invés de partir para os exemplos práticos e tentar copiá-los, eu decidi fazer um caminho diferente. Inicialmente, obtive toda a documentação oficial existente naquela época e comecei a estudar apenas os conceitos principais do framework. Eu lembro de ter ficado pelo menos 20 dias compreendendo toda a arquitetura do Flex e isso me deu uma base muito boa quando fui para a prática.

Quando comecei a criar formulários com caixas de texto, botões e tudo mais, todo o processo fluía normalmente e, dado o meu conhecimento na estrutura do framework, quase tudo funcionava perfeitamente. Felizmente fui muito bem sucedido naquela época, consegui um ótimo emprego por causa disso

O que eu quero deixar nesse comentário pessoal é uma lição para todos os leitores, pois sim, eu conheço muito bem vocês, pois escrevo artigos há 15 anos. \*Quando estiverem aprendendo uma tecnologia nova, não foquem inicialmente nos exemplos, ou em copiar código e ver a tela funcionando. Foquem na teoria mais sucinta possível sobre ela. E a partir dela, construa seus exemplos. Você notará uma evolução muito boa no domínio de qualquer tecnologia.

## Arquitetura do Android

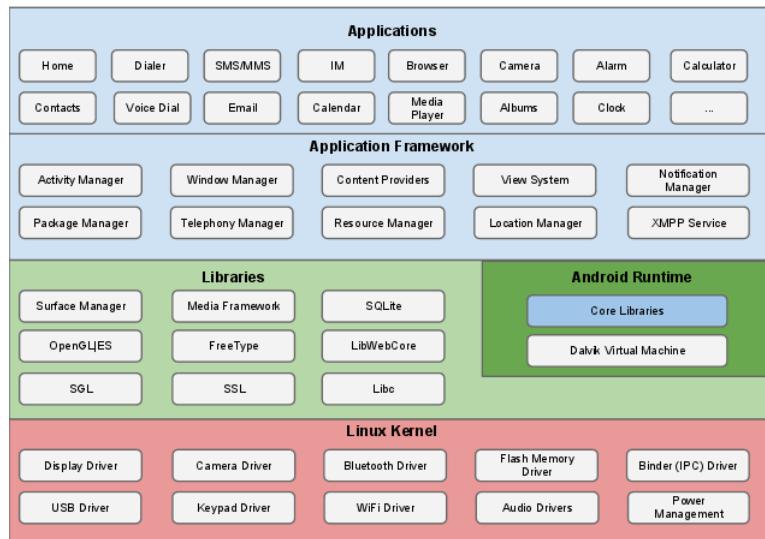
Todo o sistema Android, que roda nos dispositivos mobile, é construído sobre uma das plataformas mais solidas existentes no mercado: o linux. Criado por Linus Torvalds em 1991, o linux é um sistema operacional open source utilizado desde pequenos sistemas (quem sabe até sua geladeira usa) até super computadores da Nasa! O Android usa o linux para o gerenciamento de memória, de processos, conexões em rede, serviços, arquivos.

Apesar do Android utilizar o Linux, em teoria a sua aplicação nunca vai acessá-lo, não é possível executar um comando em Linux diretamente pelo dispositivo, já que a arquitetura do Android

não permite isso. Como um desenvolvedor Android, fique ciente que isso não será uma boa prática.

## Camadas e mais camadas

O sistema Android é dividido em camadas, fazendo com que cada uma delas tenha uma proposta bem específica. A distribuição destas camadas está ilustrada na figura a seguir:



Arquitetura de camadas do sistema Android

Veja que, no topo da camada estão as aplicações Android, como a sua lista de contatos, o navegador, widgets etc. Na parte mais inferior está o kernel do linux, no qual somente a camada mais acima acessa. Vamos exemplificar estas camadas a seguir

## A camada Libraries (Bibliotecas)

As Libraries do Android contém todo o código fonte do sistema Android e são responsáveis em lidar diretamente com o núcleo do sistema (o linux). Elas foram escritas em C ou C++ e compiladas diretamente em código de máquina, ou seja, cada tipo de smartphone teve a sua compilação realizada de acordo com o hardware, e foi pré instalada pelo fabricante do mesmo. Algumas dessas bibliotecas são descritas a seguir:

### Surface Manager

Responsável diretamente por desenhar a tela no dispositivo (semelhante ao compiz), só que de uma forma muito mais simples. A ideia foi, ao invés de escrever diretamente no buffer da tela, o android cria e combina bitmaps e somente então envia-os pra tela. Isso permite criar efeitos e transições de forma mais suave.

### Gráficos 2d e 3d

Aqui o Android usa o hardware para poder combinar gráficos 2D e 3D, podendo inclusive usar recursos nativos da OpenGL. Pode-se inclusive desenhar formas em 3D, aplicar efeitos e animações, tudo com uma API bem estruturada fornecida pelo Android.

### Media Codecs

Diversos codecs podem ser executados pelo Android, incluindo H.263, MPEG-4, entre outros. Uma boa lista de formatos de áudio são suportados tais como Wav, ACC, MP3, OGG, MIDI entre outros.

### SQLite

Assim como em outros sistemas, o Android trabalha nativamente com o SQLite, empregado em diversos projetos como Firefox e Iphone. Você pode usar o SQLite para persistir dados na sua aplicação.

### Browser interno

Usado para exibir HTML em suas aplicações, baseado no projeto Webkit, o mesmo do Google Chrome por exemplo.

Todas estas bibliotecas não podem ser acessadas diretamente pela sua aplicação, mas é claro que elas são acessadas de alguma forma e com isso temos a nossa próxima camada que é o que chamamos de framework Android, que será visto no próximo capítulo.

## E o Dalvik Virtual Machine?

Como podemos ver, não é possível acessar estas bibliotecas internas do Android, mas, caso seja necessário, podemos criar mais bibliotecas! Você pode criar uma biblioteca nativa usando o NDK (Native Development Kit) e testá-las usando o DVM, que é a máquina virtual disponível. Este recurso não será comentado na obra, mas é interessante você saber que é possível criar bibliotecas nativas

## A camada Application Framework

É nesta camada que estaremos mais focados, pois nela está quase tudo que precisamos do Android. Através do framework que conseguimos acessar as bibliotecas nativas (não de uma forma direta, claro), além de diversos recursos. Você pode reparar que a maioria dos itens desta camada são Managers, ou seja, eles gerenciam alguma coisa. Vamos ver cada item a seguir:

### Activity Manager

Controla basicamente o ciclo de vida da aplicação, incluindo a “pilha de telas” para a navegação do usuário. Quando você clica no botão Voltar do seu dispositivo, e a tela volta para a anterior, é o Activity Manager que está operando sem que você tenha que programar para isso.

### Content Providers

Este recurso encapsula a forma de acesso a dados externos do dispositivo, tais como contatos, configurações, entre outros. Você poderá tanto ler quanto escrever nestes dados.

### **Resource Manager**

Resources são qualquer informação no seu programa que não seja código, tal como textos, imagens, vídeos etc. Qualquer texto que você pode usar na aplicação (como o label de um botão) pode ser gerenciado pelo Reosource Manager para que possa ser traduzido para outros idiomas. O mesmo para imagens, você pode criar diversas imagens que irão ser aplicadas de acordo com a resolução e dpi do dispositivo.

### **Location Manager**

Usado para saber a localização do dispositivo, através do gps.

### **Notification Manager**

Usado para aquelas notificações que você recebe no dispositivo, que ficam no topo da tela.

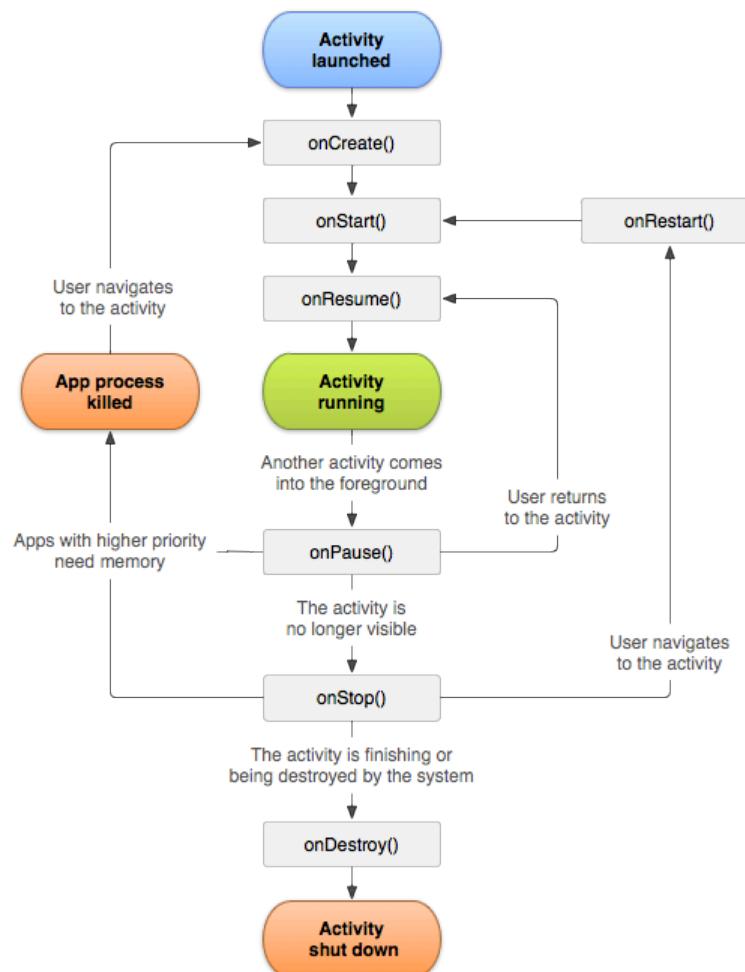
Estes são os principais itens das camadas do frameowrk Android. Existem outras, claro, mas não serão abordadas nesta obra. Ainda há alguns conceitos importantes para se explicar e vamos dar foco à eles.

## **Ciclo de vida de uma aplicação Android**

Um ciclo de vida é um conjunto de ações que são tomadas quando sua aplicação está sendo executada. Em ambientes multi janela, como Windows ou Linux(KDE, Gnome...), pode-se abrir diversas janelas e programas, exibir-los de um ao lado do outro, minimizar etc. No Android, esse processo não funciona assim. Inicialmente, o Android trabalha com o conceito de apenas uma aplicação ativa naquele momento. Não importa se você abriu uma aplicação, voltou o Home e abriu outra. Para o Android, é sempre uma aplicação por vez que está aberta.

Este processo é gerenciado pelo Activity Manager, e ele possui um ciclo de vida que é conduzido através de eventos. Um evento é um acontecimento, no qual você pode querer saber que ele existe (ou não). Se você já trabalhou com frameworks baseados em eventos, estará confortável em compreender o ciclo de vida de uma aplicação Android. Caso negativo, apenas aceite a ideia inicial e tente implementá-lá na prática.

No Android, um evento pode ser capturado através de métodos existentes que começam com a palavra “on”, como por exemplo, `onCreate`. Na imagem a seguir, temos um resumo destes eventos, e de quando eles acontecem.



Ciclo de vida de uma Activity

Pode parcer confuso a princípio, mas vamos comentar cada evento que é executado:

#### **onCreate()**

é chamado quando a aplicação é criada, momentos antes dela aparecer na tela do usuário. É neste evento que você poderá, por exemplo, adicionar dados a uma lista de um combobox, recuperar dados de uma instância anterior, criar views etc.

#### **onStart()**

Indica que a Activity está prestes a ser apresentada ao usuário. Neste ponto não é recomendado alterar dados ou informações visuais.

#### **onResume()**

Iniciada quando o usuário está iniciando uma interação com a sua aplicação e sempre é seguida pelo evento onPause.

#### **onPause()**

Iniciada quando a Activity fica em background, geralmente quando o usuário para de interagir com a aplicação. Este é o momento que você pode persistir dados ou realizar alterações que não influenciam na interface.

### onStop()

Iniciado quando a Activity não está mais visível ao usuário. Neste momento você pode persistir dados, mas não deixe de fazer isso também no onPause() caso seja necessário, pois existe uma chance do evento onStop() não ocorrer, caso a memória do dispositivo esteja cheia.

### onRestart()

Iniciado quando a Activity torna-se visível novamente, após o evento onStop() ter ocorrido.

### onDestroy()

Evento é chamado antes da aplicação ser encerrada (o processo ser destruído).

## Intents

Já vimos que uma Activity pode representar uma tela do usuário. Mas uma aplicação completa em Android é feita através de várias telas, então é preciso de algo para que possamos abrir mais telas além da principal.

Para isso, temos o Intent, que é um mecanismo que descreve uma ação específica, entre ela a ação de carregar outra Activity. No ambiente Android, qualquer ação específica é uma Intent, inclusive as mais básicas, que podem inclusive ser substituídas.

## Services

Um Service (não iremos traduzir para “serviço”) é uma tarefa que está sendo executada em background pela aplicação. Por exemplo, se você for construir um player de música, deverá programar um service para que a música continue sendo executada mesmo se a aplicação perder o foco.

## Permissões

O Android também se preocupa com a segurança do usuário. Você não ia querer, por exemplo, que uma aplicação qualquer pudesse de acessar a internet do dispositivo para enviar dados, ou acessar a sua câmera sem você saber.

Para isso, existem diversas permissões nas quais o usuário deverá permitir que a sua aplicação execute. Por exemplo, se você deseja ler a lista de contatos do dispositivo, você deverá primeiro pedir permissão ao usuário. Claro que essa permissão não é feita manualmente, você pode adicionar permissões logo que cria o seu projeto pelo assistente. Tenha cuidado ao adicionar permissões que não vá usar. Por exemplo, se você está criando uma aplicação que persiste dados comuns no banco SQLite, não há necessidade de adicionar a permissão para acesso à câmera.

Alguns exemplos de permissões:

### INTERNET

Permissão para acesso a Internet

**READ\_CONTACTS e WRITE\_CONTACTS**

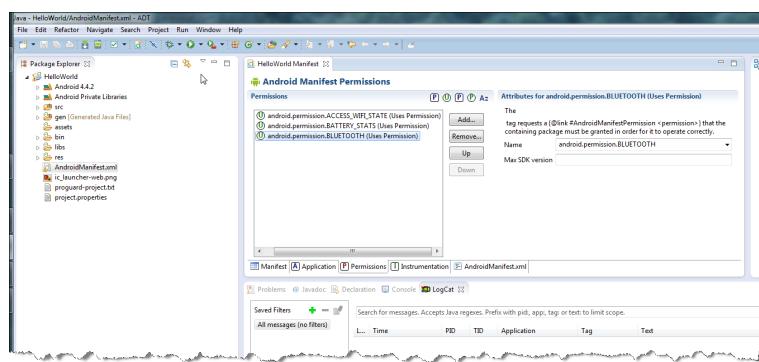
Ler e alterar informações sobre os contatos do dispositivo.

**RECIEVE\_SMS**

Monitorar o recebimento de SMS do dispositivo

Existem dezenas de outras permissões, que podem ser acessadas [neste link<sup>5</sup>](http://developer.android.com/reference/android/Manifest.permission.html).

As permissões podem ser facilmente adicionadas no arquivo `AndroidManifest.xml` através do editor visual, conforme a figura a seguir.



Adicionando permissões ao projeto

Uma permissão também pode ser adicionada manualmente, bastando adicionar um item no arquivo `AndroidManifest.xml`, conforme o código a seguir:

```
1 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
2 <uses-permission android:name="android.permission.BATTERY_STATS"/>
3 <uses-permission android:name="android.permission.BLUETOOTH"/>
```

## Arquivos XML?

Como você já começou a ver, o Android usa extensivamente arquivos XML. Seja no `AndroidManifest.xml` ou na definição de interface, o XML é usado. Com certeza você ficou na dúvida se este método é o mais melhor, pois sabemos que processar arquivos XML é algo lento.

Mas o Android usa XML apenas para facilitar o nosso desenvolvimento, pois é humanamente entendível. Quando o Android compila toda a aplicação, todos os XML são convertidos em forma binária, para que sejam acessados e trabalhados com mais rapidez. Em um binário de aplicação do Android, praticamente não existe arquivos XML para serem lidos.

<sup>5</sup><http://developer.android.com/reference/android/Manifest.permission.html>

## Resources

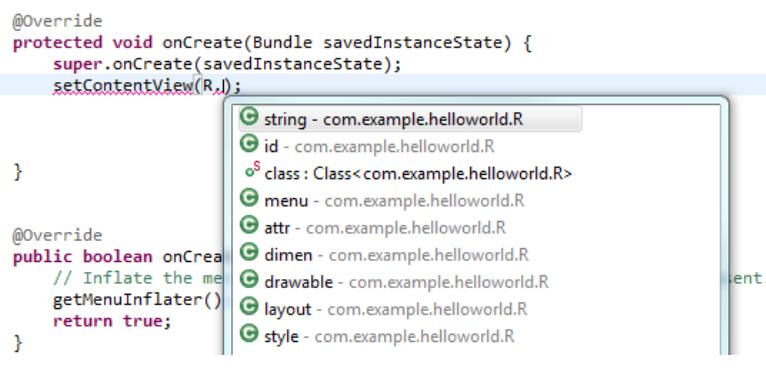
Um resource, ou recurso, é um texto, uma imagem bitmap, um vídeo etc. Qualquer conteúdo que pode ser alterado dado uma informação externa, como o locale ou tamanho do dispositivo, deve ser armazenado em um resource.

Os Resources são armazenados na pasta `res`, e são devidamente compilados para serem utilizados no código java através de uma classe chamada de, simplesmente, `R`. Por exemplo, em `res/values/strings.xml` temos:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <string name="app_name">Hello World</string>
5     <string name="action_settings">Settings</string>
6     <string name="hello_world">Hello world!</string>
7
8 </resources>
```

Isso significa que, quando compilado, você poderá acessar estes textos através do código java: `R.string.app_name`. No próprio código Java, quando você digita `R.`, o próprio eclipse já complementa com as strings disponíveis, conforme a figura a seguir.



Usando resources

Com este recurso, podemos facilmente criar aplicações em várias línguas, e também podemos adicionar imagens para diversos tipos de dispositivo.

## Métricas

É importante comentar sobre as principais métricas utilizadas pelo sistema Android. Todos nós estamos mais acostumados com o tamanho formado por pixels, que são os “pontos por polegada” na tela de um monitor. Só que as telas dos dispositivos mais modernos já apresentam mais pontos por polegada, fazendo que com que o uso da métrica `pixels` seja ineficiente.

O Android suporta uma variedade de métricas, dentre elas temos:

**px (pixels)**

Pontos por polegada, muito utilizada em desktop

**in (inches)**

É um tamanho fixo, a famosa “polegada”, utilizada mais nos EUA.

**mm (milímetros)**

É um tamanho fixo, utilizado no Brasil

**pt (pontos)**

É um tamanho fixo, representa 1/72 de uma polegada

**dp ou dip(densidade-independente por pixels)**

É uma unidade abstrata baseada na densidade de pontos da tela do dispositivo. Ou seja, dependendo da quantidade de pontos da tela, essa densidade por ser maior ou menor. Se uma tela possui 160 pontos por polegada, então 1dp é exatamente igual a 1px. Em celulares mais avançados (pelo menos quando esta obra foi escrita), uma tela podia ter até 306 pontos por polegada (Galaxy S3<sup>6</sup>), o que representaria quase 0,5px.

**sp (escala independente em pixels)**

Esta escala é usada de acordo com as preferências de texto do dispositivo android, em relação ao tamanho do texto.

Em suma, o Android usa 2 unidades principais. Para tamanho de texto, usa-se sempre SP, já que o usuário pode querer mudar o tamanho da letra. Para qualquer outra métrica, usamod dp (ou dip). Não é recomendado usar pixels ou qualquer outra unidade.

---

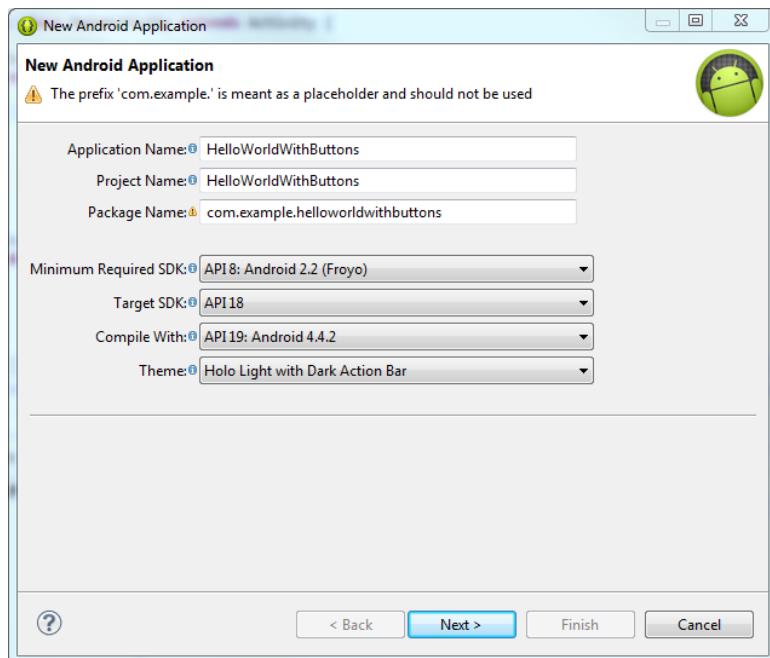
<sup>6</sup><http://www.samsung.com/ae/microsite/galaxys3/en/specifications.html>

# Capítulo 5 - Além do HelloWorld

Neste capítulo vamos criar outro projeto, um pouco mais completo que o HelloWorld que fizemos no capítulo 2. A ideia agora é adicionar alguns botões na tela e explicar mais alguns conceitos importantes, como o gerenciamento de locales, a disposição do dispositivo e o uso do Intent para criar uma ligação entre duas Activities.

## Criando o projeto

Usando o assistente que vimos no capítulo 2, crie um novo projeto chamado conforme a figura a seguir:

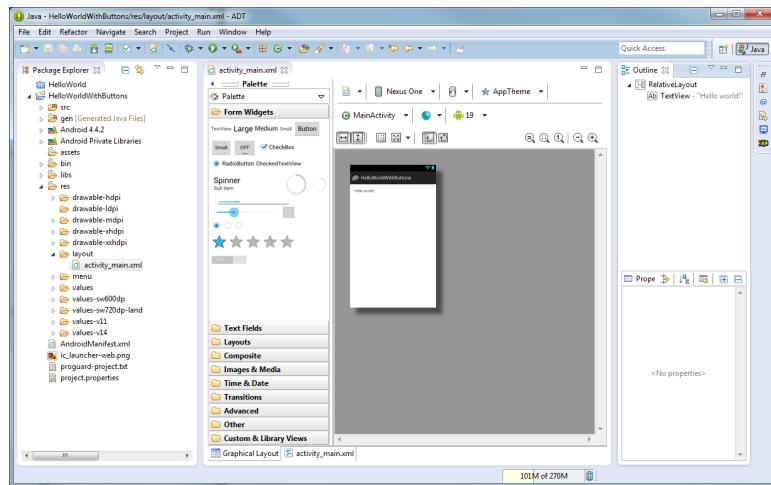


Usando resources

Nas próximas telas do assistente, você pode deixar tudo conforme o padrão. Se achar necessário, altere o ícone da aplicação para um outro qualquer e deixe o assistente criar uma Blank Activity chamada MainActivity.

## Compreendendo layouts

Vamos inicialmente adicionar alguns botões na tela. Para isso, acesse `res/layout/activity_main.xml` para ver uma tela conforme a figura a seguir:



Usando resources

Na parte inferior da tela, temos a forma de representação do `activity_main.xml`, que pode ser gráfica (Graphical Layout) ou em modo texto (activity\_main.xml). Ao selecionar o modo texto, temos o seguinte código:

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:paddingBottom="@dimen/activity_vertical_margin"
6   android:paddingLeft="@dimen/activity_horizontal_margin"
7   android:paddingRight="@dimen/activity_horizontal_margin"
8   android:paddingTop="@dimen/activity_vertical_margin"
9   tools:context=".MainActivity" >
10
11   <TextView
12     android:layout_width="wrap_content"
13     android:layout_height="wrap_content"
14     android:text="@string/hello_world" />
15
16 </RelativeLayout>
```

Temos muitas informações neste código XML, mesmo em poucas linhas. Inicialmente, temos o `<RelativeLayout>` que determina a forma como os objetos dentro deste Layout são organizados. Na linha 3, temos a definição da largura do componente, que nesse caso é `match_parent`, que significa ocupar todo o espaço possível, com um mínimo de margem. Em versões anteriores este valor era conhecido como `fill_parent`, então você ainda pode encontrar este valor em alguns projetos. Na linha 5 temos algumas definições de `padding` (espacamento) que são definidas através de um Resource.

## Tipos de Layouts

Temos basicamente quatro tipos de layouts para pode usar em nossas views.

### FrameLayout

Adiciona os seus componentes filhos sempre no topo da área do layout, à esquerda. Ou seja, se você adiciona dois componentes à um FrameLayout, eles ficarão sobrepostos, caso não haja uma informação da posição do item. Use FrameLayout quando for adicionar somente um item, como uma imagem por exemplo.

### LinearLayout

Os componentes filho do LinearLayout são adicionados um abaixo do outro, ou um ao lado do outro. É o estilo mais usado nas aplicações.

### RelativeLayout

Neste layout, os componentes podem ser arranjados de forma que um dependa do outro. Ou seja, o posicionamento de um componente é medido em relação a outro componente, e não ao Layout em que ele está inserido.

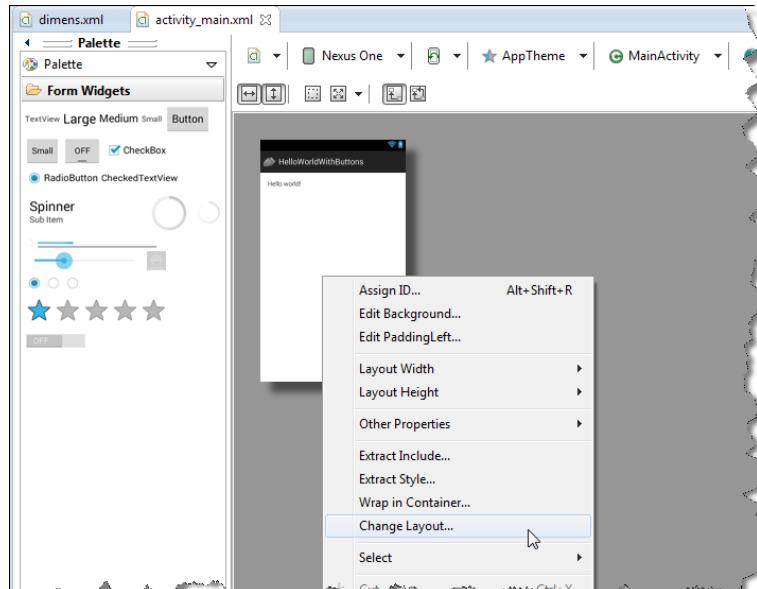
### TableLayout

Adiciona os componentes em linhas e colunas, semelhante a uma tabela HTML.

## Alterando um layout

Na criação do projeto, o layout usado no arquivo `res/layout/activity_main.xml` foi o RelativeLayout. Podemos mudá-lo de duas formas, tanto na interface gráfica quanto no arquivo XML.

Pela interface gráfica, clique com o botão direito do mouse na Activity e selecione o item Change Layout, conforme a figura a seguir



Usando resources

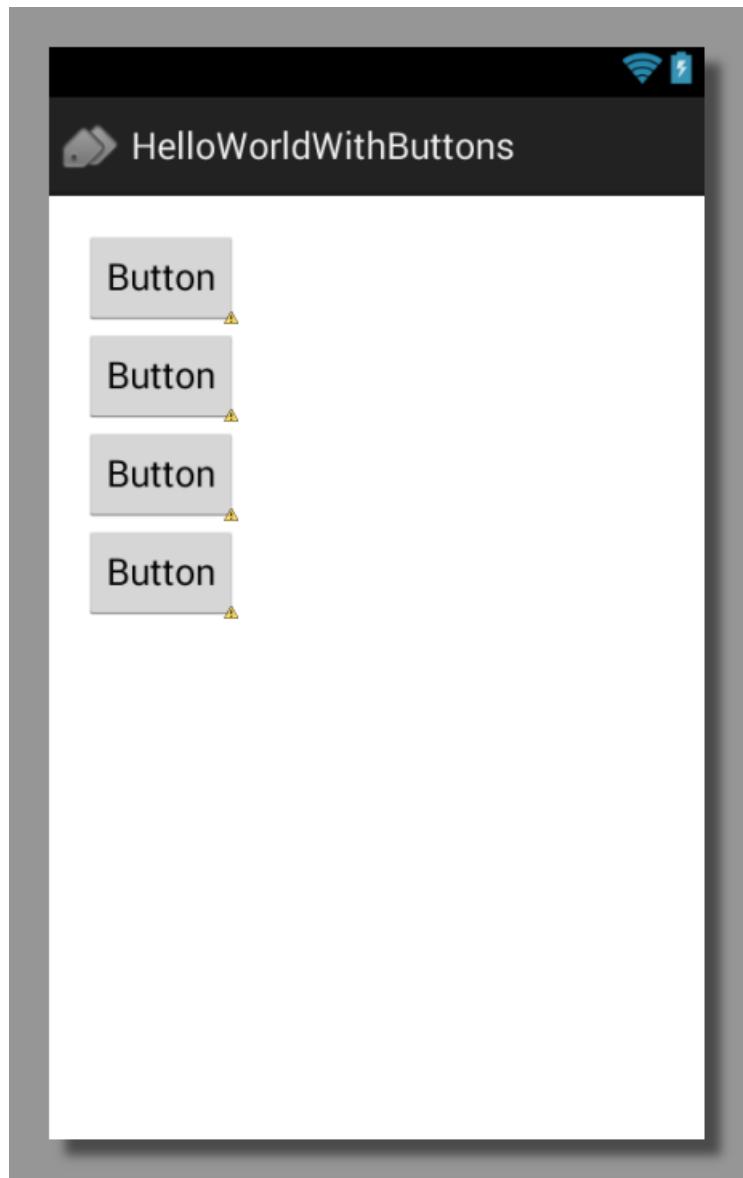
Altere o Layout para “LinearLayout (Vertical)”. Os componentes inseridos dentro deste layout serão inseridos um abaixo do outro. Verifique o novo código XML do `activity_main.xml`, que deve estar semelhante ao exibido a seguir:

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:id="@+id/LinearLayout2"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical"
7     android:paddingBottom="@dimen/activity_vertical_margin"
8     android:paddingLeft="@dimen/activity_horizontal_margin"
9     android:paddingRight="@dimen/activity_horizontal_margin"
10    android:paddingTop="@dimen/activity_vertical_margin"
11    tools:context=".MainActivity" >
12
13    <TextView
14        android:layout_width="wrap_content"
15        android:layout_height="wrap_content"
16        android:text="@string/hello_world" />
17
18 </LinearLayout>
```

## Inserindo Botões

Agora que alteramos o Layout, vamos inserir 4 botões na view. Usando o editor visual, apague o TextView onde está escrito Hello World e arraste quatro botões para o LinearLayout.

Neste momento, temos a seguinte tela:



Após inserir 4 botões

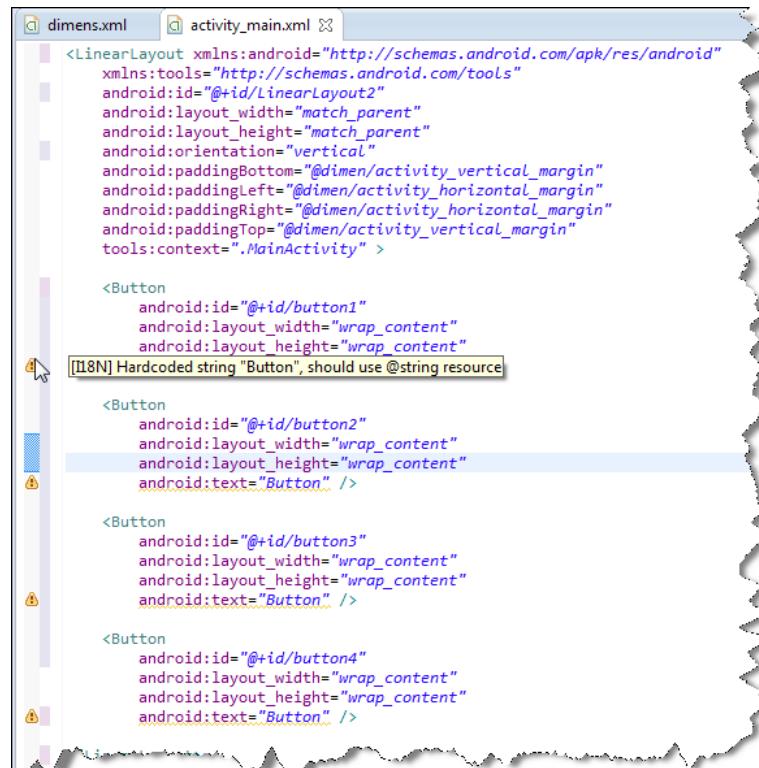
E temos o seguinte XML:

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

```
1     xmlns:tools="http://schemas.android.com/tools"
2     android:id="@+id/LinearLayout2"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical"
6     android:paddingBottom="@dimen/activity_vertical_margin"
7     android:paddingLeft="@dimen/activity_horizontal_margin"
8     android:paddingRight="@dimen/activity_horizontal_margin"
9     android:paddingTop="@dimen/activity_vertical_margin"
10    tools:context=".MainActivity" >
```

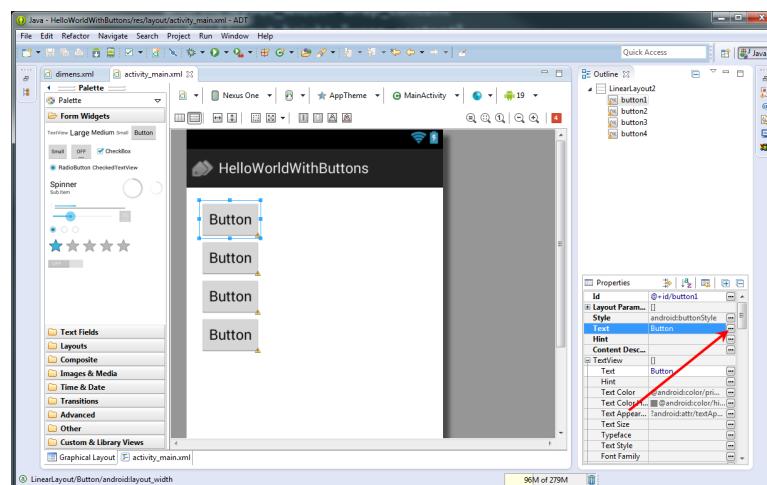
```
11      <Button
12          android:id="@+id/button1"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:text="Button" />
16
17      <Button
18          android:id="@+id/button2"
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:text="Button" />
22
23
24      <Button
25          android:id="@+id/button3"
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:text="Button" />
29
30      <Button
31          android:id="@+id/button4"
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:text="Button" />
35
36  </LinearLayout>
```

Repare que existe um ponto de exclamação em cada botão da tela, inclusive no código XML. Ao aproximar o mouse no ícone de exclamação, surge uma mensagem recomendando o uso de Resources na propriedade text do botão.



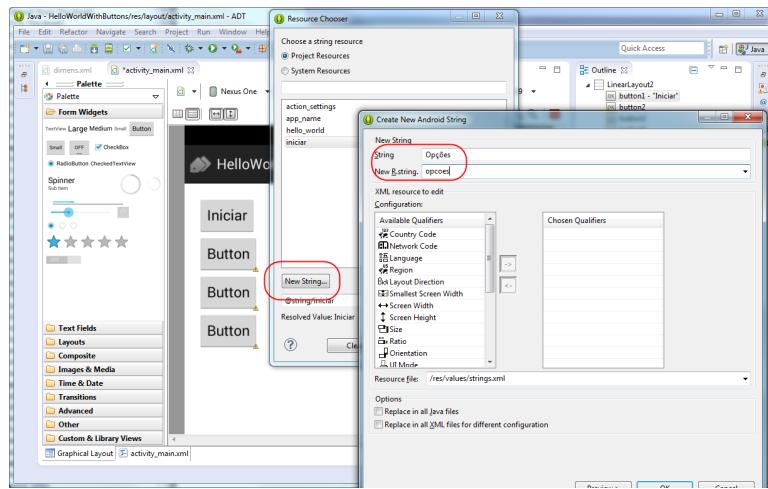
Layout após inserir 4 botões

Para resolver este problema, volte no layout gráfico e selecione o botão em questão. Na aba de propriedades, encontre a propriedade Text e clique no assistente, conforme a figura a seguir:



Alterando o Text do botão

Na tela que surge, você poderá adicionar quatro strings em Project Resource. Para isso, clique em New String e adicione a chave e o valor da String.

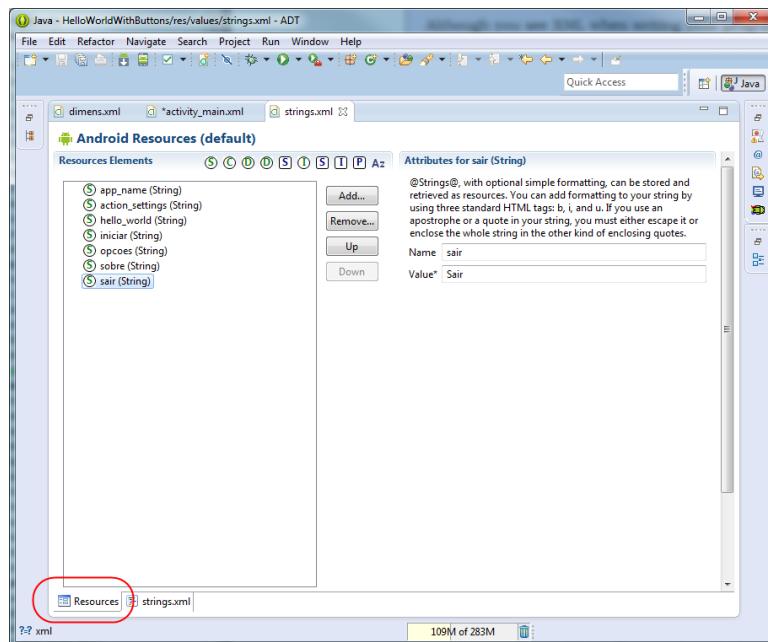


Inserindo uma String

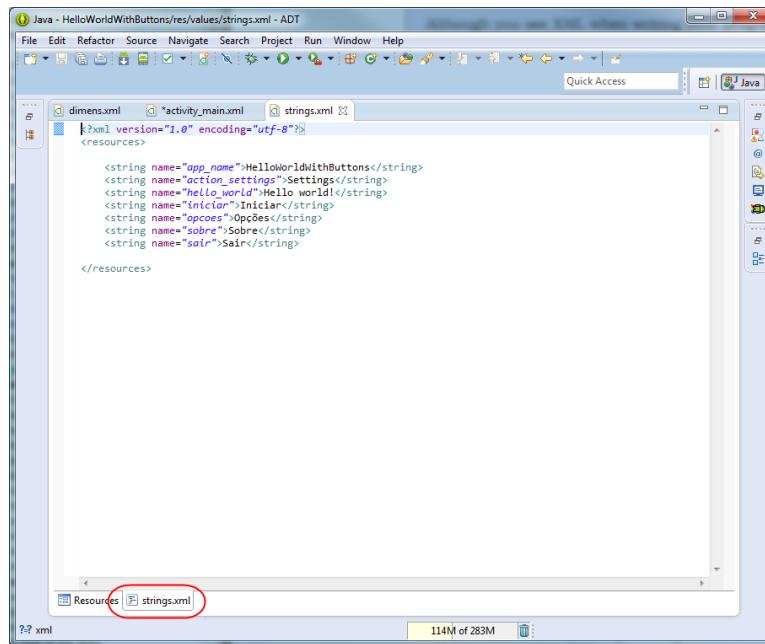
Vamos adicionar os seguintes valores:

- Name: start Value: Iniciar
- Name: options Value: Opções
- Name: about Value: Sobre
- Name: exit Value: Sair

Se você não se sentir confortável em adicionar as Strings desta forma, você pode abrir o arquivo `res/values/strings.xml` e editar manualmente o arquivo. Lembre-se que a maioria dos XMLs de configuração do Android possuem um editor pele eclipse. Por exemplo, no caso do `strings.xml`, temos o seguinte editor:



Editando Strings.xml no editor visual



Editando Strings.xml em modo texto

Edite a propriedade Text dos quatro labels, para ter algo semelhante a imagem a seguir:



Editando Strings.xml em modo texto

Agora precisamos deixar os botões com o mesmo tamanho, e ocupando a largura da tela. Para isso, precisamos alterar a propriedade Width de cada componente para `fill_parent`.

O resultado até agora é visto na figura a seguir:



Editando Strings.xml em modo texto

E o código XML deve ser semelhante ao exibido a seguir:

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:id="@+id/LinearLayout2"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical"
7     android:paddingBottom="@dimen/activity_vertical_margin"
8     android:paddingLeft="@dimen/activity_horizontal_margin"
9     android:paddingRight="@dimen/activity_horizontal_margin"
10    android:paddingTop="@dimen/activity_vertical_margin"
11    tools:context=".MainActivity" >
12
13    <Button
```

```
14     android:id="@+id/button1"
15     android:layout_width="fill_parent"
16     android:layout_height="wrap_content"
17     android:text="@string/start" />
18
19 <Button
20     android:id="@+id/button2"
21     android:layout_width="fill_parent"
22     android:layout_height="wrap_content"
23     android:text="@string/options" />
24
25 <Button
26     android:id="@+id/button3"
27     android:layout_width="fill_parent"
28     android:layout_height="wrap_content"
29     android:text="@string/about" />
30
31 <Button
32     android:id="@+id/button4"
33     android:layout_width="fill_parent"
34     android:layout_height="wrap_content"
35     android:text="@string/exit" />
36
37 </LinearLayout>
```

## Utilizando Resources alternativos

Neste capítulo iremos mostrar um pouco de como o sistema de Resources do Android é poderoso e simples de usar. Inicialmente, vamos supor a seguinte situação: Suponha que você quer deixar os botões da aplicação mais centralizado na tela, adicionando um espaço em branco no topo do primeiro botão. Existem diversas formas de fazer isso, mas vamos propor uma solução simples, que é alterar a propriedade `android:paddingTop` do `<LinearLayout>`.

É claro que não vamos alterar esta propriedade para algo como `android:paddingTop="100dp"`, mas sim usar os Resources para configurar a propriedade. Neste caso, não alteramos em nada o `paddingTop` do componente, deixamos como está, ou seja: `android:paddingTop="@dimen/activity_vertical_margin"` e alteramos o arquivo `res/values/dimens.xml`, que foi previamente criado pelo eclipse.

Ao acessar `res/values/dimens.xml`, temos:

```
1 <resources>
2
3     <!-- Default screen margins, per the Android Design guidelines. --&gt;
4     &lt;dimen name="activity_horizontal_margin"&gt;16dp&lt;/dimen&gt;
5     &lt;dimen name="activity_vertical_margin"&gt;16dp&lt;/dimen&gt;
6
7 &lt;/resources&gt;
8
9 E podemos alterar para:</pre>
```

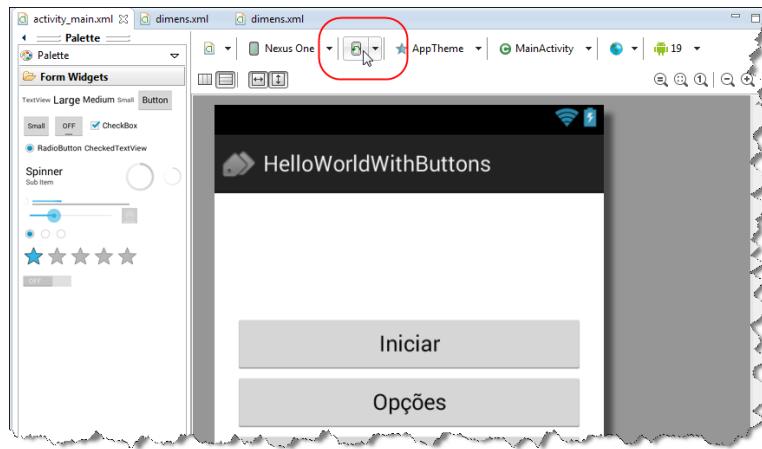
```
1 <resources>
2
3     <!-- Default screen margins, per the Android Design guidelines. --&gt;
4     &lt;dimen name="activity_horizontal_margin"&gt;16dp&lt;/dimen&gt;
5     &lt;dimen name="activity_vertical_margin"&gt;100dp&lt;/dimen&gt;
6
7 &lt;/resources&gt;</pre>
```

Ou seja, alteramos de 16dp para 100dp a propriedade `activity_vertical_margin`. Como a propriedade `paddingTop` aponta para esta entrada no Resources, ao acessar a tela com os botões teremos o seguinte resultado:



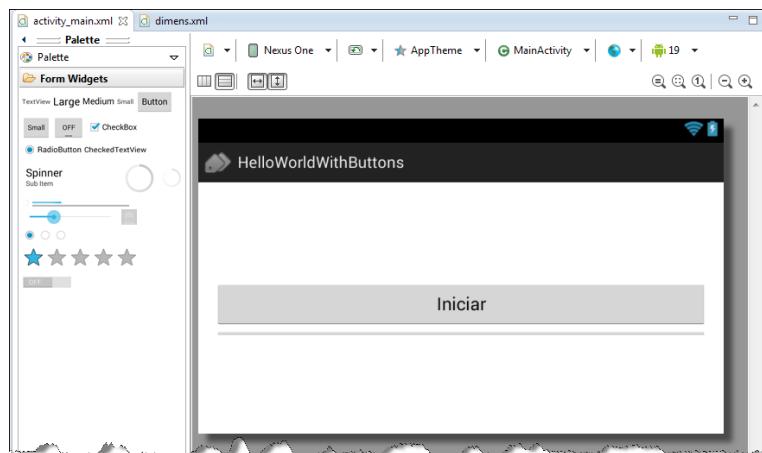
Alterando o paddingTop

Lembrando que, estamos fazendo essa modificação apenas para explicar os recursos que o Resource possui. Vamos agora mostrar o problema que esta modificação gerou. No eclipse, altere a visualização do dispositivo para horizontal. Isso é feito clicando no botão landscape conforme o detalhe a seguir:



Alterando o paddingTop

Esta alteração provocará o seguinte resultado:

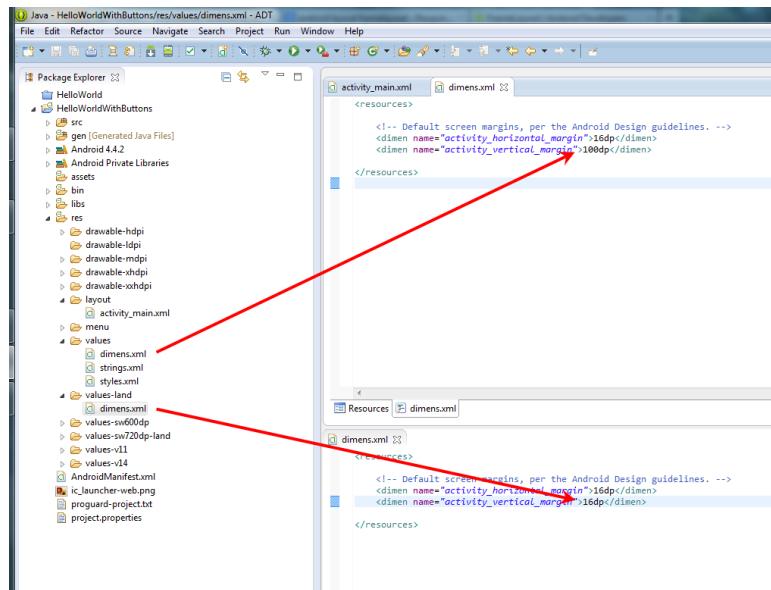


Alterando o paddingTop

Ou seja, os botões que estão usando a margem no topo ficaram abaixo da tela. Para resolver este problema, podemos novamente usar os Resources e adicionar uma entrada diferente de margem para quando o dispositivo estiver na orientação horizontal.

Para isso, basta criar uma pasta chamada `values-land` abaixo da pasta `res`. Ou seja, se você deseja adicionar uma configuração diferente para a sua tela quando ela está na horizontal, basta adicionar `-land` (de `landscape`) na pasta em que se deseja alterar.

Então, como a informação de altura da tela está na pasta `values`, podemos criar uma pasta `values-land` e copiar o arquivo `values/dimens.xml` para a pasta `values-land`, conforme a figura a seguir:



Alterando o paddingTop

Agora que existe a pasta `values-land`, ela será usada quando o dispositivo estiver no modo landscape, fazendo com que os botões não usem mais a altura selecionada no modo vertical.

## Alterando o Layout da tela com Resources

Continuando nesta mesma abordagem, podemos alterar completamente o design da tela de acordo com a sua orientação. Para isso, crie a pasta `layout-land` abaixo da pasta `res` e copie o arquivo `activity_main.xml` para ele.

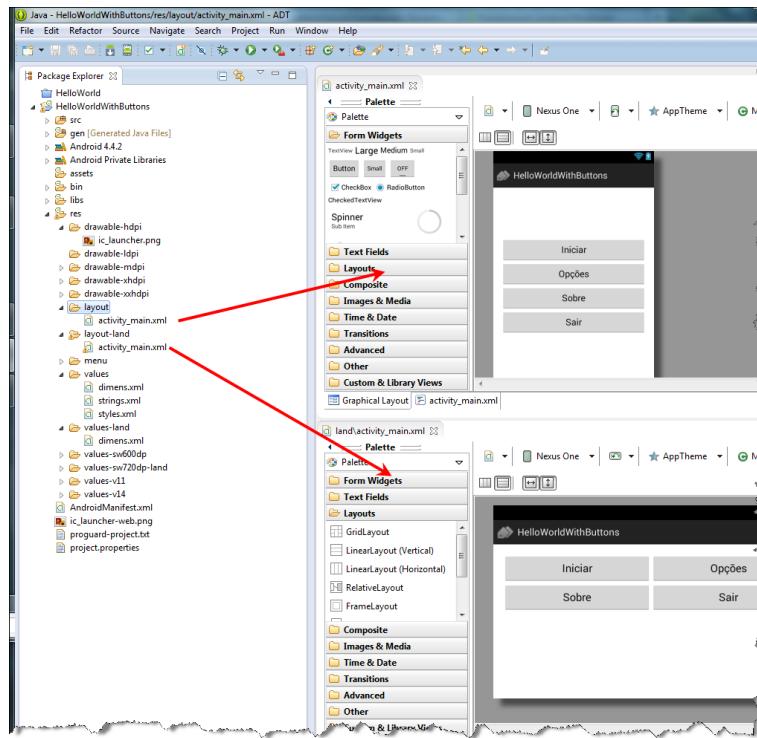
Faremos uma alteração no arquivo `res/layout-land/activity_main.xml` de forma a incluir 2 botões na horizontal, conforme o código a seguir:

```

1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:id="@+id/LinearLayout2"
4   android:layout_width="match_parent"
5   android:layout_height="match_parent"
6   android:orientation="vertical"
7   android:paddingBottom="@dimen/activity_vertical_margin"
8   android:paddingLeft="@dimen/activity_horizontal_margin"
9   android:paddingRight="@dimen/activity_horizontal_margin"
10  android:paddingTop="@dimen/activity_vertical_margin"
11  tools:context=".MainActivity" >
12
13  <TableLayout
14    android:layout_width="fill_parent"
15    android:layout_height="fill_parent"
16    android:stretchColumns="*" >
```

```
17
18    <TableRow
19        android:id="@+id/tableRow1"
20        android:layout_width="wrap_content"
21        android:layout_height="wrap_content" >
22
23        <Button
24            android:id="@+id/button1"
25            android:layout_width="fill_parent"
26            android:layout_height="wrap_content"
27            android:text="@string/start" />
28
29        <Button
30            android:id="@+id/button2"
31            android:layout_width="fill_parent"
32            android:layout_height="wrap_content"
33            android:text="@string/options" />
34    </TableRow>
35
36    <TableRow
37        android:id="@+id/tableRow2"
38        android:layout_width="wrap_content"
39        android:layout_height="wrap_content" >
40
41        <Button
42            android:id="@+id/button3"
43            android:layout_width="fill_parent"
44            android:layout_height="wrap_content"
45            android:text="@string/about" />
46
47        <Button
48            android:id="@+id/button4"
49            android:layout_width="fill_parent"
50            android:layout_height="wrap_content"
51            android:text="@string/exit" />
52    </TableRow>
53 </TableLayout>
54
55 </LinearLayout>
```

No código acima, usamos TableLayout para criar colunas na tela e adicionar os botões nela. O resultado das duas telas é visto na figura a seguir:

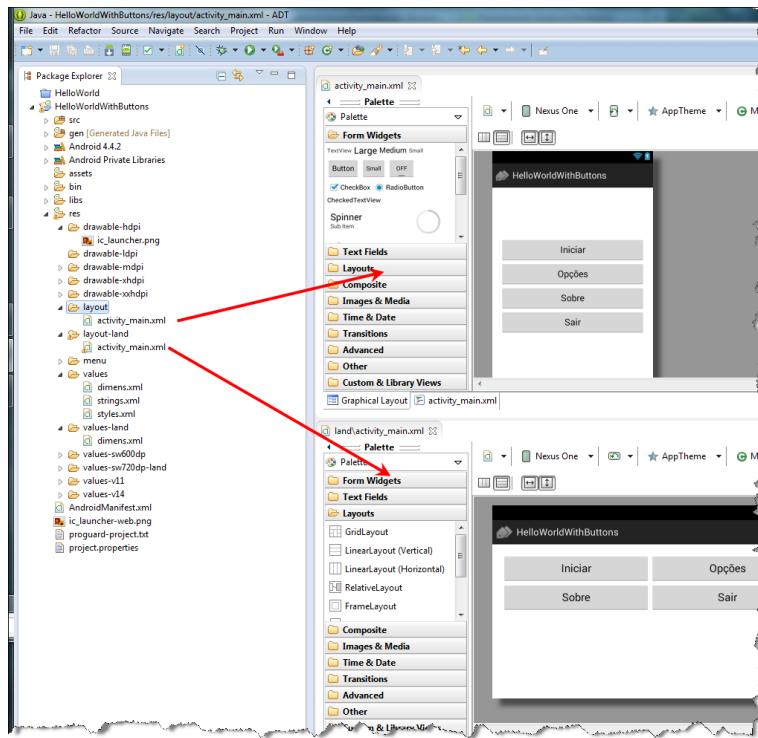


Telas no formato vertical e horizontal

## Alterando o idioma com Resources

Da mesma forma que alteramos valores para uma determinada orientação de tela, também podemos alterar o idioma da aplicação utilizando Resources. Seguindo a mesma ideia que fizemos nos capítulos anteriores, precisamos criar uma pasta que identifique o locale da aplicação.

Isso significa criar uma pasta `values-<locale>` na pasta res. Por exemplo, na pasta `values` usamos o idioma português, que é o nosso idioma principal, e para criar uma versão em inglês, basta criar a pasta `values-en` e copiar o arquivo `values/strings.xml` para `values-en/strings.xml`, conforme a figura a seguir:



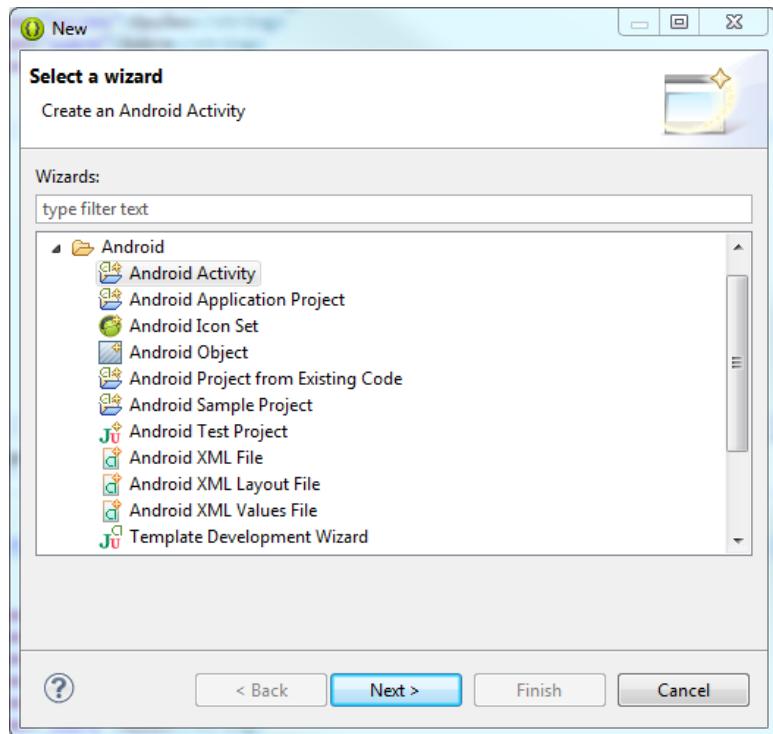
Locales diferentes

Em cada dispositivo android existe a informação sobre o locale selecionado pelo usuário, e o idioma será carregado automaticamente de acordo com esta configuração.

## Criando a tela Sobre

Neste exemplo hipotético criamos uma tela simples com alguns botões, e a ideia aqui é apenas demonstrar alguns conceitos básicos sobre o funcionamento do Android. Neste momento, vamos criar uma outra tela (um outro Activity), e vamos criar um simples texto utilizando Resources, como se fosse uma tela sobre o sistema em questão.

Para criar uma nova Activity, acesse File > New > Other. No assistente que surge, escolha a pasta Android e o item Android Activity, conforme a figura a seguir:



Criando uma nova Activity

Clique em Next e crie um Blank Activity, com o nome AboutActivity. A Activity é criada, juntamente com os arquivos `res/layout/activity_about.xml` e `src/com/example/helloworldwithbuttons/MainA`

A tela “Sobre” ainda o código gerado pelo assistente, ou seja, um `RelativeLayout` e um `TextView`. Na interface gráfica, você pode alterar o `RelativeLayout` para um `ScrollView` usando a ferramenta `Change Layout`. Também alteramos o `TextView` do Hello World para About, conforme o código a seguir:

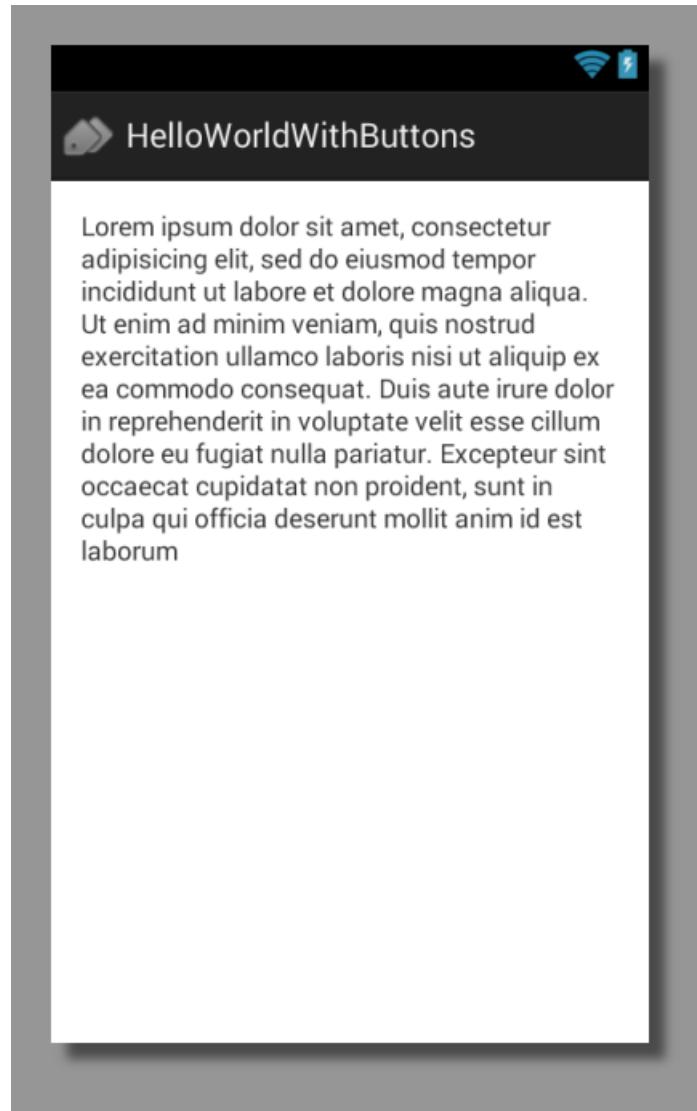
```

1 <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:id="@+id/ScrollView1"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     android:paddingBottom="@dimen/activity_vertical_margin"
7     android:paddingLeft="@dimen/activity_horizontal_margin"
8     android:paddingRight="@dimen/activity_horizontal_margin"
9     android:paddingTop="@dimen/activity_vertical_margin"
10    tools:context=".AboutActivity" >
11
12    <TextView
13        android:layout_width="wrap_content"
14        android:layout_height="wrap_content"
15        android:text="@string/about_text" />
16
17 </ScrollView>
```

Repare que o TextView possui em sua propriedade text o seguinte parâmetro: @string/about\_text. Isso significa que precisamos prover esta entrada e isso é realizado editando o arquivo res/values/strings.xml, que ficará da seguinte forma:

```
1 <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3
4     <string name="app_name">HelloWorldWithButtons</string>
5     <string name="action_settings">Configurações</string>
6     <string name="hello_world">Hello world!</string>
7     <string name="start">Iniciar</string>
8     <string name="options">Opções</string>
9     <string name="about">Sobre</string>
10    <string name="exit">Sair</string>
11    <string name="title_activity_about">AboutActivity</string>
12
13    <string name="about_text">Lorem ipsum dolor sit amet, consectetur adipisicing\\
14      elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut\\
15      enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip\\
16      ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate\\
17      velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat\\
18      cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est\\
19      laborum</string>
20
21 </resources>
```

O resultado final desta Activity é semelhante a figura a seguir.



Tela “sobre”

## Preparando o botão Sobre

Agora que temos a view pronta, precisamos abri-la quando o botão “sobre” for clicado. Para isso, você deve adicionar um listener ao evento click do botão, que tem uma forma um pouco complexa de se fazer, por isso, vamos fazer isso linha a linha.

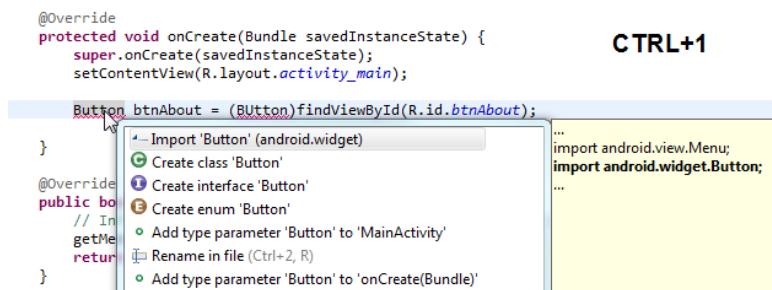
Primeiro, lembre-se que podemos adicionar este código no evento `onCreate` do `MainActicity`, logo após o método `setContentView`. Nossa primeira tarefa é *pegar a instância do botão*. Quando criamos os quatro botões na tela, eles foram criados com os ids: `button1`, `button2`, `button3`, `button4`. É chegado o momento de alterar estes IDs, então o `button3` que representa o botão Sobre será modificado para `btnAbout`. Você pode fazer isso no `activity_main.xml`. Quando fizer isso, surgirá uma tela para atualizar as instâncias desse id em todo o projeto, clique em `Ok` e altere-as.

Agora que sabemos que `btnAbout` é o nosso botão na tela, temos que pegar a sua instância, e isso fazemos através de um método especial chamado `findViewById`. Você usará muito este método,

da seguinte forma:

```
1 Button btnAbout = (Button) findViewById(R.id.btnAbout);
```

Aqui temos um código 100% java. Inicialmente temos a criação de uma variável chamada btnAbout que é do tipo Button. Neste momento, a IDE irá apontar um erro dizendo que a classe Button é desconhecida, pois no Java nós precisamos importar as classes na qual estamos trabalhando. Você faz isso rapidamente na IDE apertando CTRL+1 em cima da palavra Button, obtendo o seguinte resultado:



Corrigindo o erro de classe não encontrada

O CTRL+1 é um atalho para uma ferramenta no eclipse chamada Quick Fix, usada para corrigir rapidamente os erros de programação no Java. Neste caso, você deverá escolher a primeira opção, que é importar a classe android.widget.Button.

Após criarmos a variável btnAbout, vamos achar a instância do botão na tela e fazer a referência. Para isso, usamos o método findViewById. Este método retorna um objeto que é do tipo android.View.view, que é um objeto bem genérico da tela. Como o nosso btnButton é do tipo Button, devemos alterar o tipo de retorno do método e isso é feito através da definição da classe entre parêntesis. Chamamos esse processo de TypeCast. O método findViewById aceita um parâmetro, que é o ID do objeto da tela. Este ID está disponível através do nosso querido Resource que está sempre disponível para nos dar uma ajuda. Então basta usar R. id. e escolher o Id btnAbout.

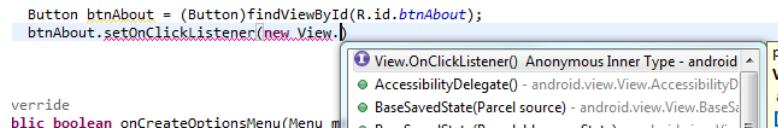
Agora que temos a instancia do botão na tela, representada pela variável btnAbout, vamos adicionar o listener para o clique do botão. Isso é feito de forma relativamente rápida, pois a IDE trabalha para você. Vamos fazer esse procedimento passo a passo. Inicialmente, adiciona o seguinte código:

```
1 btnAbout.setOnClickListener|
```

O “|” é uma forma de representar o cursor. Após escrever setOn, aperte ctrl+barra de espaços para que a complementação de código apareça! Escolha o método setOnClickListener e aperte enter, teremos algo assim:

`btnAbout.setOnClickListener()`

Quando o cursor estiver dentro do parêntesis, digite new View. e aperte ctrl+barra de espaços de novo, surgindo algo semelhante a figura a seguir:



### Adicionando um listener

Escolha `View.OnClickListener` e deixe a IDE gerar o código para você! Teremos algo muito semelhante ao código a seguir:

```

1 btnAbout.setOnClickListener(new View.OnClickListener() {
2
3     @Override
4     public void onClick(View v) {
5         // TODO Auto-generated method stub
6
7     }
8 });

```

Agora poderemos colocar o código no método `onClick`, e ele será executando quando o usuário pressiona o botão. O código completo da classe `MainActivity` até agora é exibido a seguir:

```

1 package com.example.helloworldwithbuttons;
2
3 import android.os.Bundle;
4 import android.app.Activity;
5 import android.view.Menu;
6 import android.view.View;
7 import android.widget.Button;
8
9 public class MainActivity extends Activity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15
16         Button btnAbout = (Button)findViewById(R.id.btnAbout);
17         btnAbout.setOnClickListener(new View.OnClickListener() {
18
19             @Override
20             public void onClick(View v) {
21                 // TODO Auto-generated method stub
22
23             }
24         });
25

```

```

26     }
27
28     @Override
29     public boolean onCreateOptionsMenu(Menu menu) {
30         // Inflate the menu; this adds items to the action bar if it is present.
31         getMenuInflater().inflate(R.menu.main, menu);
32         return true;
33     }
34 }
35
36 }
```

## Criando o intent

Como já vimos anteriormente, o Intent é a ação de fazer alguma coisa no Android. Pode ser abrir uma nova Activity, tocar um som ou vídeo, entre outras coisas. No nosso caso, vamos criar uma instância para o Intent e carregar a Activity, da seguinte forma:

```

1     @Override
2     public void onClick(View v) {
3         // TODO Auto-generated method stub
4         Intent openAbout = new Intent(v.getContext(), AboutActivity.class);
5         startActivity(openAbout);
6     }
```

É preciso ter muito cuidado nesse método `onClick`. Inicialmente você precisa ter uma boa definição de escopo. O Escopo deste método é o listener em si, e não a Activity. Ou seja, se você usa `this` dentro deste método, você está referenciando o listener e não a classe `MainActivity`. Dominando isso, fica fácil perceber que o método `onClick` possui o parâmetro `v` que é a instancia da view na qual o evento foi chamado.

Ao criarmos o Intent `openAbout`, perceba que passamos como primeiro parâmetro o seguinte método: `v.getContext()`. Isso retorna o contexto da View na qual o botão foi clicado. O segundo parâmetro é a nova Activity que sera chamada. Nesse caso, a classe `AboutActivity`. O código completo do `MainActivity` é definido a seguir.

```

1 package com.example.helloworldwithbuttons;
2
3 import android.os.Bundle;
4 import android.app.Activity;
5 import android.content.Intent;
6 import android.view.Menu;
7 import android.view.View;
8 import android.widget.Button;
9
```

```

10
11 public class MainActivity extends Activity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17
18         Button btnAbout = (Button)findViewById(R.id.btnAbout);
19         btnAbout.setOnClickListener(new View.OnClickListener() {
20
21             @Override
22             public void onClick(View v) {
23                 Intent openAbout = new Intent(v.getContext(), AboutActivity.clas\ss);
24                 startActivity(openAbout);
25             }
26         });
27     }
28
29 }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33         // Inflate the menu; this adds items to the action bar if it is present.
34         getMenuInflater().inflate(R.menu.main, menu);
35         return true;
36     }
37 }
38
39 }
```

## Aplicando um estilo diferente na tela About

Para finalizar este capítulo, uma dica rápida sobre temas. Qualquer Activity pode ter um tema pré definido, existem vários prontos para você usar. No caso da Activity About, você poderá alterar o arquivo Mainfest alterando o tema padrão. Para fazer isso, acesse o arquivo `AndroidManifest.xml` e adicione a seguinte linha na Activity About:

```

1 <activity
2     android:name=".AboutActivity"
3     android:label="@string/title_activity_about"
4     android:theme="@android:style/Theme.Dialog"
5 >
```

Reexecute o projeto no seu dispositivo e veja a diferença.

## Implementando o botão Sair

Apesar de não ser muito usual “sair de aplicação android”, você pode implementar esta ação através do método `finish()`, no qual pode relacionar ao botão “Sair”. A forma de fazer isso é a mesma do botão “Sobre”, deixaremos a seguir o código pronto, mas é interessante que você escreva todo o código ao invés de copiar e colar.

```
1 Button btnSair = (Button)findViewById(R.id.btnSair);
2     btnSair.setOnClickListener(new View.OnClickListener() {
3
4         @Override
5         public void onClick(View v) {
6             // TODO Auto-generated method stub
7             finish();
8         }
9     });
10 );
```

## Debug is on the table

Umas das atividades mais rotineiras do programador é debugar uma aplicação, principalmente se algo de errado estiver acontecendo. Existem duas formas de debugar uma aplicação, sendo a primeira, mais comum em tecnologias mais antigas, na qual não se havia uma IDE para facilitar este processo, através de logs da aplicação. Ou seja, na sua aplicação, você adiciona um log que sera salvo em algum lugar, geralmente em um arquivo texto.

Se você programou muito em PHP, principalmente há anos atrás, lembrará do comando `var_dump` que no android é representado pelo comando `Log`. Existem 5 níveis de Log no android, sendo eles:

- `Log.e()`: Exibir erros
- `Log.w()`: Exibir warnings
- `Log.i()`: Exibir informações
- `Log.d()`: Exibir debug
- `Log.v()`: Exibir verbose

E o terrível:

- `Log.wtf()`: Exibe um “What a Terrible Failure” :)

Lembre-se que os usuários “comuns” jamais verão essas mensagens de erro, elas estarão sempre na janela de Log da aplicação, no caso a janela “LogCat” da IDE.

Outra forma de Debug, e esta é mais eficiente que a primeira, é através da IDE, na qual você insere *breakpoints* no código e pode inspecionar, linha a linha, o que está acontecendo com a aplicação. Você pode analisar variáveis em tempo de execução e verificar diversas particularidades no código.

# **Capítulo 6 - Configurações personalizadas**

Toda aplicação, por mais simples que seja (até a Hello Wolrd, se você quiser), necessita persistir dados em algum momento. O sistema Android permite a persistência de dados em vários níveis, desde dados simples e pequenos até o mais comum que é através de SQL e banco de dados SQLite. Neste capítulo veremos como personalizar dados de uma aplicação através da criação de uma tela de opções.

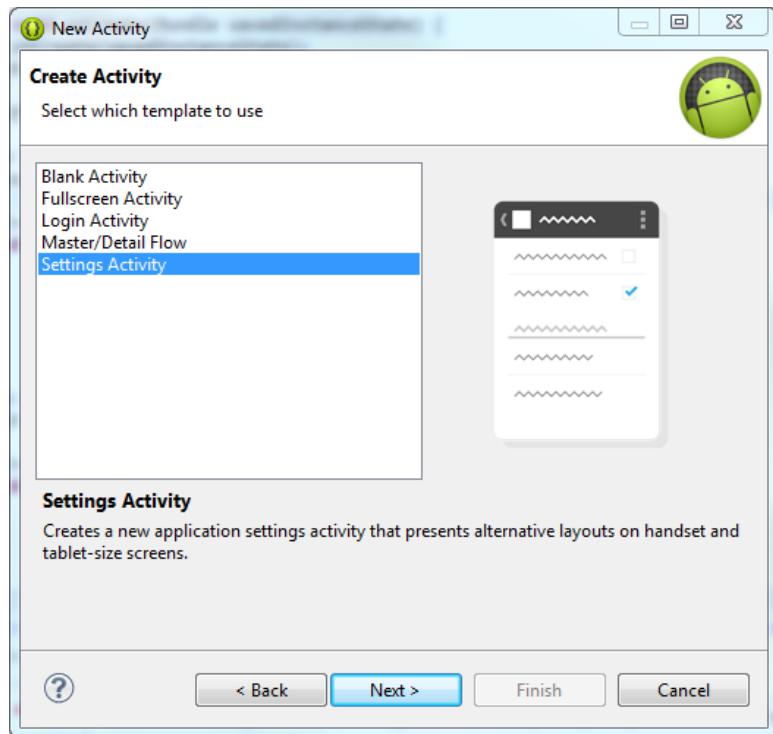
## **Criando a tela de opções**

Geralmente toda aplicação tem uma tela de opções, na qual informações simples são armazenadas. Em um jogo pode ser a dificuldade do mesmo, ou se deverá usar som ou não. Em uma aplicação, pode ser o login do usuário, se deve conectar na internet via HTTPS, ou o IP do servidor remoto, informações de Proxy etc.

## **Utilizando o assistente**

Assim como na maioria das tarefas rotineiras existentes em uma aplicação, o Android possui uma API pronta para uso, de forma que armazenar as preferências da sua aplicação não será problema.

Continuando com o projeto HelloWorldWithButtons, vamos dar vida ao botão de opções da aplicação. Precisamos, inicialmente, criar uma nova Activity, mas desta vez, quando formos escolher o template da Activity, escolheremos “Settings Activity”, conforme a imagem a seguir:



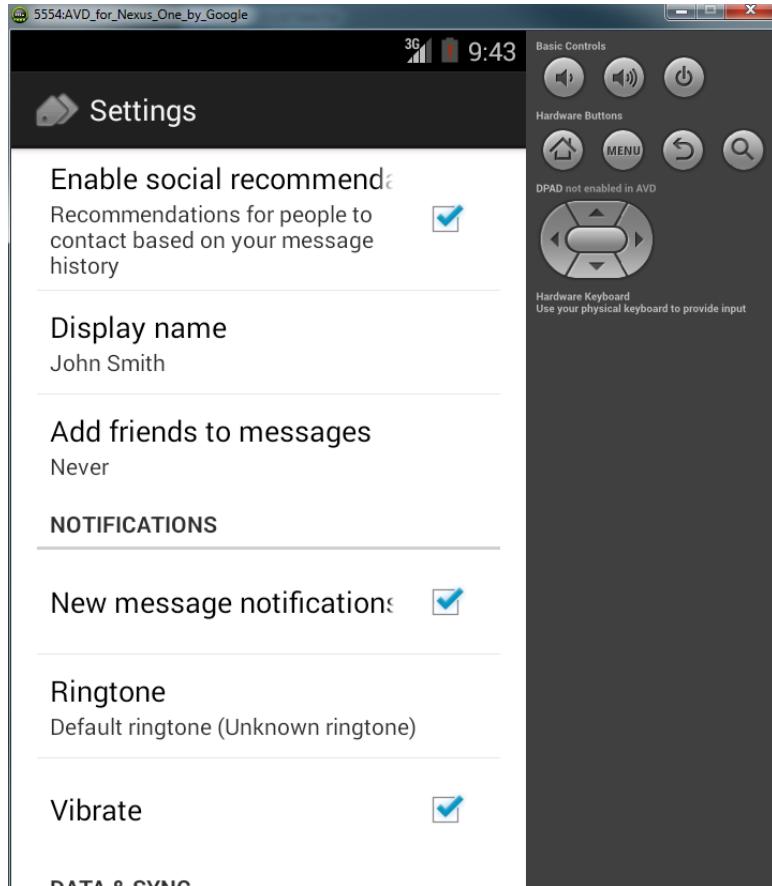
Settings Activity

Vamos chamar esta Activity de “SettingsActivity” e finalizar o assistente da criação da tela. A Activity `SettingsActivity.java` é criada, mas como você vai notar, não é criado o arquivo `res/layout/activity_settings.xml`, mas são criados diversos outros arquivos que dão suporte a tela de opções.

Inicialmente vamos fazer o botão “Opções” carregar esta tela, e isso é feito da mesma forma do capítulo anterior, de acordo com o código a seguir:

```
1 public class MainActivity extends Activity {  
2  
3     protected void onCreate(Bundle savedInstanceState) {  
4         ...  
5         ...  
6         ...  
7         Button btnSettings = (Button)findViewById(R.id.btnSettings);  
8         btnSettings.setOnClickListener(new View.OnClickListener() {  
9             @Override  
10            public void onClick(View v) {  
11                Intent openSettings = new Intent(v.getContext(), SettingsActivity.class);  
12                startActivity(openSettings);  
13            }  
14        });  
15        ...  
16        ...  
17        ...  
18    }  
19}
```

Agora execute a aplicação e clique no botão Opções, você verá algo semelhante a tela a seguir:



Tela de opções em ação

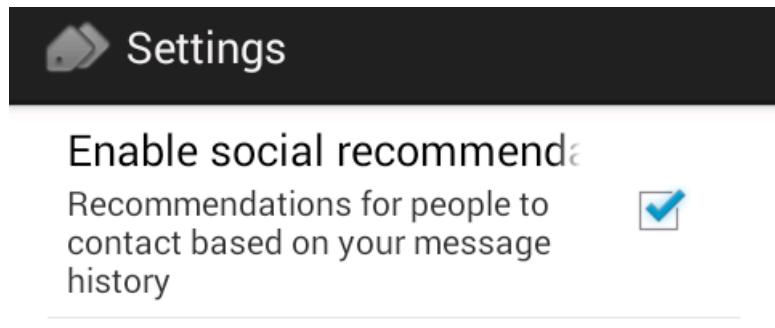
Agora vamos dar uma olhada melhor no arquivo `SettingsActivity.java`. Você verá muito código gerado, alguns foram criados para se adaptar melhor aos tablets, enquanto outros configuraram outras coisas. Neste arquivo, uma das linhas mais importantes é essa:

```
1 addPreferencesFromResource(R.xml.pref_general);
```

O comando `addPreferencesFromResource` vai usar o arquivo `res/xml/pref_general.xml` para criar a tela de opções. Isso só é possível porque essa Activity herda de `PreferenceActivity`. Vamos então dar uma olhada por completo neste arquivo:

```
1 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
2
3     <CheckBoxPreference
4         android:defaultValue="true"
5         android:key="example_checkbox"
6         android:summary="@string/pref_description_social_recommendations"
7         android:title="@string/pref_title_social_recommendations" />
8
9     <!-- NOTE: EditTextPreference accepts EditText attributes. -->
10    <!-- NOTE: EditTextPreference's summary should be set to its value by the \
11 activity code. -->
12    <EditTextPreference
13        android:capitalize="words"
14        android:defaultValue="@string/pref_default_display_name"
15        android:inputType="textCapWords"
16        android:key="example_text"
17        android:maxLines="1"
18        android:selectAllOnFocus="true"
19        android:singleLine="true"
20        android:title="@string/pref_title_display_name" />
21
22    <!--
23        NOTE: Hide buttons to simplify the UI. Users can touch outside the di\
24 alog to
25            dismiss it.
26    -->
27    <!-- NOTE: ListPreference's summary should be set to its value by the acti\
28 vity code. -->
29    <ListPreference
30        android:defaultValue="-1"
31        android:entries="@array/pref_example_list_titles"
32        android:entryValues="@array/pref_example_list_values"
33        android:key="example_list"
34        android:negativeButtonText="@null"
35        android:positiveButtonText="@null"
36        android:title="@string/pref_title_add_friends_to_messages" />
37
38 </PreferenceScreen>
```

Neste arquivo, temos em cada nó uma definição de itens que serão carregados na tela de opções. O primeiro item que é um CheckBoxPreference com o id example\_checkbox representa este campo:



Primeiro campo da tela de opções

O próximo campo é um `EditPreference` e o outro um `ListPreference`. Perceba que algumas configurações de cada campo apontam para campos de outros arquivos XML, como por exemplo `android:title=@string/pref_title_social_recommendations`.

## Atenção

Para compreender tanto os itens quanto as propriedades deste recurso, recomenda-se ter a API do Android sempre à mão. Por exemplo, talvez a propriedade `entryValues` do componente `ListPreference` seja desconhecida para você, então você pode buscar esta propriedade na api.

## Como acessar as opções

Após configurar a tela de opções e colocar os itens de cada opção no arquivo `res/xml/pref_general`, você poderá acessar as opções através da própria API do Android, utilizando o método `PreferenceManager.getDefaultSharedPreferences()`. No exemplo a seguir, adicionamos um “Hello, usuário” no título da aplicação (`MainActivity.java`):

```
1 String nomeSettings = (String)PreferenceManager.getDefaultSharedPreferences().get\\
2 tApplicationContext().getString("example_text", "usuario(a)");
3
4 setTitle("Olá, " + nomeSettings);
```

Neste exemplo, usamos a classe `PreferenceManager` e o método `getDefaultSharedPreferences` para obter o texto que está ligado a chave “`example_text`”, que é o id que foi criado pelo assistente.