Android Cookbook

Ian F. Darwin

Novatec

Editor: Rubens Prates
Tradução: Rafael Zanolli
Revisão técnica: Edgard Damiani
Revisão gramatical: Giacomo Leone Neto
Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-323-9

Histórico de impressões:

Setembro/2012 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
Email: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Darwin, Ian F.
Android cookbook / Ian F. Darwin. -- São Paulo : Novatec Editora ; Sebastopol, CA : O`Reilly, 2012.

ISBN 978-85-7522-323-9

1. Android (Programa de computador)
2. Aplicação de programa - Desenvolvimento
3. Computação móvel 4. Google 5. Internet sem fio
6. Telefones celulares I. Título.

12-11335 CDD-005.26

Índices para catálogo sistemático:

1. Android : Plataforma de desenvolvimento para aplicativos móveis : Programa de computador 005.26

OG20120918

CAPÍTULO 1

Primeiros passos

1.1 Introdução: primeiros passos

Ian Darwin

Discussão

O famoso padrão "Olá, mundo" (*Hello, world*) surgiu quando Kernighan e Plaugher quiseram escrever uma "receita" sobre como começar a utilizar qualquer nova linguagem e ambiente de programação. Este capítulo é carinhosamente dedicado a esses nobres senhores, e a todos que já lutaram para dar seus primeiros passos em qualquer novo paradigma de programação.

1.2 Aprendendo a linguagem Java

Ian Darwin

Problema

Aplicativos Android são escritos na linguagem de programação Java antes de serem convertidos no formato de arquivo de classe próprio do Android, DEX. Se você não sabe como programar em Java, vai ter dificuldade em escrever aplicativos Android.

Solução

Muitos recursos estão disponíveis para aprender Java. A maioria deles ensinará o que você precisa, mas também mencionará algumas classes de API (Application Programming Interface) que não estão disponíveis para o desenvolvimento Android. *Evite* quaisquer seções em qualquer recurso que fale sobre os tópicos listados na coluna esquerda da tabela 1.1.

Tabela 1.1 – Componentes da API Java a ignorar

API Java	Equivalente Android
Swing, applets	GUI do Android; veja o capítulo 7

API Java	Equivalente Android
Ponto de entrada de aplicativo `main()`	Veja a receita 1.6
J2ME/Java ME	A maior parte do `android.*` substitui a API Java ME
Servlets/JSP, J2EE/Java EE	Projetados para uso no lado do servidor

Discussão

Aqui estão livros e recursos sobre programação Java:

- *Java in a Nutshell* (lançado no Brasil pela editora Bookman, com o título *Java: o guia essencial*) de David Flanagan (O'Reilly) é uma boa introdução para programadores, especialmente àqueles que estão vindo de C/C++. Esse livro cresceu muito em tamanho para acompanhar o crescimento do Java SE ao longo de seu tempo de vida.
- *Head First Java* (lançado no Brasil pela editora Starlin Alta, com o título *Use a cabeça! Java*) de Kathy Sierra e Bert Bates (O'Reilly). Esse livro fornece uma ótima apresentação orientada ao aprendizado visual para essa linguagem.
- *Thinking in Java* (em uma tradução livre, *Pensando em Java*) de Bruce Eckel (Prentice-Hall).
- *Learning Java* (lançado no Brasil pela editora Campus, com o título *Aprendendo Java*) de Patrick Niemeyer e Jonathan Knudsen (O'Reilly).
- *"Great Java: Level 1"* (em uma tradução livre, *Java excelente: nível 1*), um vídeo feito por Brett McLaughlin (O'Reilly). Esse vídeo fornece uma apresentação visual para a linguagem.
- *Java: The Good Parts* (lançado no Brasil pela editora Campus, com o título *O melhor do Java*) de Jim Waldo (O'Reilly).
- *Java Cookbook*, o qual eu escrevi e a O'Reilly publicou. Esse livro é tido como um bom segundo livro para desenvolvedores Java. Ele tem capítulos completos sobre strings, expressões regulares, números, data e hora, estruturas de dados, E/S e diretórios, internacionalização, threads, e redes, todo temas que se aplicam ao Android. Ele também tem vários capítulos específicos para Swing e algumas tecnologias com base em EE.

Entenda, por favor, que essa lista provavelmente nunca vai estar totalmente atualizada. Você também deve consultar o *Android Development Bibliography*, da O'Reilly, disponível gratuitamente para download (mediante registro), uma compilação de todos os livros de várias editoras cujos livros estão no serviço online Safari. Esse livro também é distribuído sem custos em conferências relevantes nas quais a O'Reilly tem um estande.

Veja também

O autor principal deste livro mantém uma lista online de recursos Java em *http://www.darwinsys.com/java/*.

A O'Reilly tem alguns dos melhores livros Java do mercado; veja uma lista completa em *http://oreilly.com/pub/topic/java*.

1.3 Criação de um aplicativo "Olá, mundo" a partir da linha de comando

Ian Darwin

Problema

Você deseja criar um novo projeto Android sem utilizar o plugin ADT do Eclipse.

Solução

Utilize a ferramenta `android` do kit de desenvolvimento do Android (Android Development Kit, ou ADK) com o argumento `create project` e alguns argumentos adicionais para configurar seu projeto.

Discussão

Além de ser o nome da plataforma, `android` também é o nome de uma ferramenta de linha de comando para criação, atualização e gerenciamento de projetos. Você pode navegar até o diretório *android-sdk-xxx*, ou definir sua variável `PATH` para incluir seu subdiretório *tools*.

Depois, para criar um novo projeto, utilize o comando `android create project` com alguns argumentos. O exemplo 1.1 é um exemplo executado em MS-DOS.

Exemplo 1.1 – Criação de um novo projeto

```
C:> PATH=%PATH%;"C:\Documents and Settings\Ian\My Documents\android-sdk-windows\tools"; \
          "C:\Documents and Settings\Ian\My Documents\android-sdk-windows\platform-tools"
C:> android create project --target android-7 --package com.example.foo
    --name Foo --activity FooActivity --path .\MyAndroid
Created project directory: C:\Documents and Settings\Ian\My Documents\MyAndroid
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\src\com\example\foo
Added file C:\Documents and Settings\Ian\My
    Documents\MyAndroid\src\com\example\foo\FooActivity.java
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\bin
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\libs
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values
```

```
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values\strings.xml
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout\main.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\AndroidManifest.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\build.xml

C:>
```

A tabela 1.2 lista os argumentos para o código `create project`.

Tabela 1.2 – Lista de argumentos de `create project`

Nome	Significado	Exemplo
`--activity`	Nome de sua "classe principal" e nome-padrão para o arquivo *.apk* gerado	`--activity HelloActivity`
`--name`	Nome do projeto e do arquivo *.apk* gerado	`--name MyProject`
`--package`	Nome do pacote Java para suas classes	`--package com.example.hello`
`--path`	Path (caminho) no qual o projeto será criado (não cria um subdiretório dentro dele, por isso não utilize `/home/you/workspace`, mas, em vez disso, `/home/you/workspace/NewProjectName`)	`--path /home/ian/workspace/MyProject` (veja o exemplo anterior para Windows)
`--target`	Nível de API almejado da plataforma Android; utilize `android list targets` para ver a lista de alvos. Um número é um "ID", e não um nível de API; para tanto, utilize `android-` com o nível de API que você deseja	`--target android-10`

Se ele não puder completar a operação solicitada, o comando `android` apresentará uma extensa mensagem de "uso do comando", listando todas as operações que ele pode realizar, e os argumentos para elas. Se tiver sucesso, o comando `android create project` criará os arquivos e diretórios listados na tabela 1.3.

Tabela 1.3 – Artefatos criados por `create project`

Nome	Significado
`AndroidManifest.xml`	Arquivo de configuração do projeto, com informações essenciais sobre o projeto para o Android
`bin`	Binários gerados (arquivos de classe compilados)
`build.properties`	Arquivo de propriedades editável
`build.xml`	Arquivo de controle de compilação Ant padrão
`default.properties` ou `project.properties` (dependendo da versão das ferramentas)	Armazena a versão do SDK e as bibliotecas utilizadas; mantido por plugin
`gen`	Elementos gerados
`libs`	Bibliotecas, é claro
`res`	Arquivos de recursos importantes (*strings.xml*, layouts etc.)
`src`	Código-fonte de seu aplicativo

Nome	Significado
src/packagename/ActivityName.java	Fonte da atividade de início equivalente a "main"
test	Cópias da maioria dos anteriores

É uma prática normal e recomendada no Android criar sua interface de usuário em XML utilizando o arquivo de layout criado em *res/layout*, mas é certamente possível escrever todo o código em Java. Para manter este exemplo autocontido, faremos do jeito "errado" por enquanto. Utilize seu editor de texto favorito para substituir o conteúdo do arquivo *HelloWorld.java* com o conteúdo do exemplo 1.2.

Exemplo 1.2 – HelloWorld.java

```
import android.app.Activity;
import android.widget.*;

public class Hello extends Activity {
    /**
    * Este método é invocado quando a atividade é instanciada em resposta,
    * por exemplo, a um clique seu no ícone do aplicativo na tela inicial
    */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Cria uma TextView para a atividade atual
        TextView view = new TextView(this);
        // Faz ele dizer algo
        view.setText("Hello World");
        // Coloca essa visão recém-criada dentro da atividade,
        // mais ou menos como JFrame.getContentPane().add(view)
        setContentView(view);
    }
}
```

Supondo que você tenha a ferramenta Ant Build Tool da Apache Software Foundation instalada (e ela está incluída em versões recentes do SDK do Android), você pode (em uma janela de linha de comando) ir até o diretório do projeto (...*MyDocuments**MyAndroid* no exemplo 1.1) e utilizar o comando:

```
ant debug
```

Isso vai criar um arquivo chamado, por exemplo, *MyAndroid.apk* (com "apk" representando Android Package) no diretório *bin*.

Se esta é sua primeira vez aqui, você pode ter de criar um dispositivo virtual Android (Android Virtual Device, ou AVD), que é simplesmente uma configuração nomeada

para o emulador Android especificando a resolução-alvo, o nível de API e assim por diante. Você pode criar um emulador utilizando:

```
android create avd -n my_droid -t 7
```

Para mais detalhes sobre a criação de um AVD (veja a receita 3.3).

Você pode então iniciar o servidor do ADB (Android Debug Bridge) e o emulador:

```
adb start-server
emulator -avd my_droid -t 5
```

Presumindo que agora você tenha o emulador sendo executado ou seu dispositivo conectado e reconhecido via USB, você pode fazer:

```
adb -e install -r bin/MyAndroid.apk
```

A flag `-e` é para o emulador; utilize `-d` para um dispositivo real.

Se você tiver prática com scripts shell ou arquivos em lote, poderá criar um arquivo chamado, digamos, *download*, para evitar ter de digitar a invocação `adb` a cada ciclo de compilação.

Finalmente você pode iniciar seu aplicativo! Você pode utilizar a lista **Application**: toque no pequeno ícone que se parece com uma linha 5×5 de pontos, encontre seu aplicativo pelo nome e toque em seu ícone.

Você provavelmente vai considerar conveniente criar um ícone para seu aplicativo na tela inicial do dispositivo ou emulador; esse ícone sobreviverá a vários ciclos `install -r`, portanto, essa é a forma mais fácil de testar a execução de seu aplicativo.

Veja também

Receita 1.4. O blog "a little madness" tem uma formulação mais detalhada. O site oficial de referência do Android tem uma página sobre desenvolvimento sem Eclipse.

1.4 Criação de um aplicativo "Olá, mundo" no Eclipse

Ian Darwin

Problema

Você quer utilizar o Eclipse para desenvolver seu aplicativo Android.

Solução

Instale o Eclipse, o SDK (Software Development Kit) do Android e o plugin ADT (Android Development Tools). Crie seu projeto e comece a escrever seu aplicativo. Compile e teste esse aplicativo no emulador, dentro do Eclipse.

Discussão

Assim que tiver estes itens instalados, você estará pronto para iniciar:

- O IDE Eclipse
- O SDK do Android
- O plugin ADT

Se você quiser uma explicação mais detalhada da instalação desses três itens, por favor, consulte a receita 1.5.

Para iniciar, crie um novo projeto a partir do menu File -> New (veja a figura 1.1).

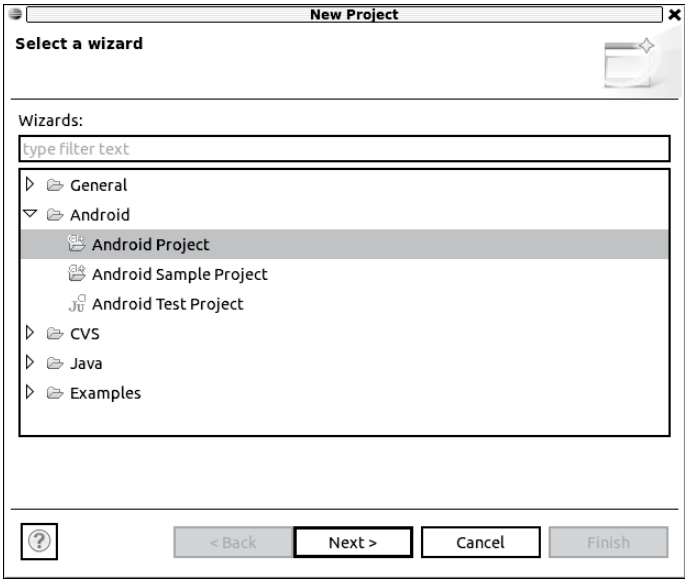


Figura 1.1 – Começando a criar um projeto Eclipse.

Clique em Next. Dê um nome ao seu novo projeto e clique em Next (veja a figura 1.2).

Selecione uma versão de SDK como alvo. A versão 2.1 oferece para você praticamente todos os dispositivos atualmente em uso; as versões 3.x ou 4.x oferecem os últimos recursos (veja a figura 1.3). Você decide.

A figura 1.4 mostra a estrutura do projeto expandida no painel Project à direita. Ela também mostra a que ponto você pode chegar na utilização do recurso de preenchimento automático do Eclipse dentro do Android – eu incluí o atributo `gravity` para o rótulo, e o Eclipse está oferecendo uma lista completa de valores de atributos possíveis. Eu escolhi `center-horizontal`, por isso o rótulo deverá estar centralizado quando colocarmos o aplicativo para funcionar.

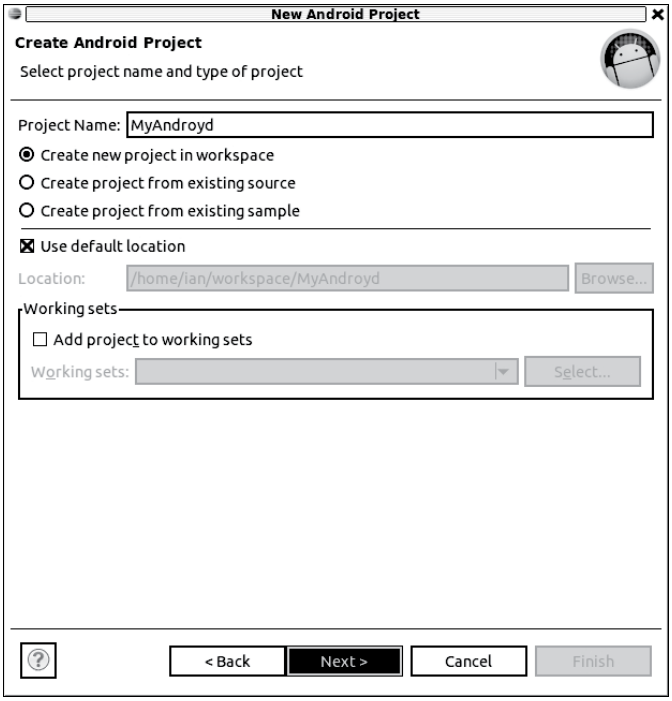


Figura 1.2 – Configuração de parâmetros para um novo projeto Eclipse.

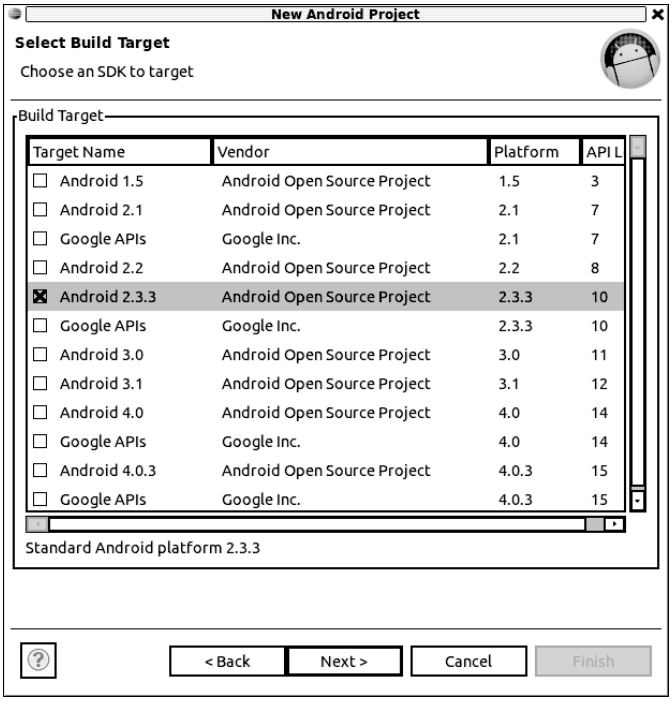


Figura 1.3 – Definição do SDK alvo para um novo projeto Eclipse.

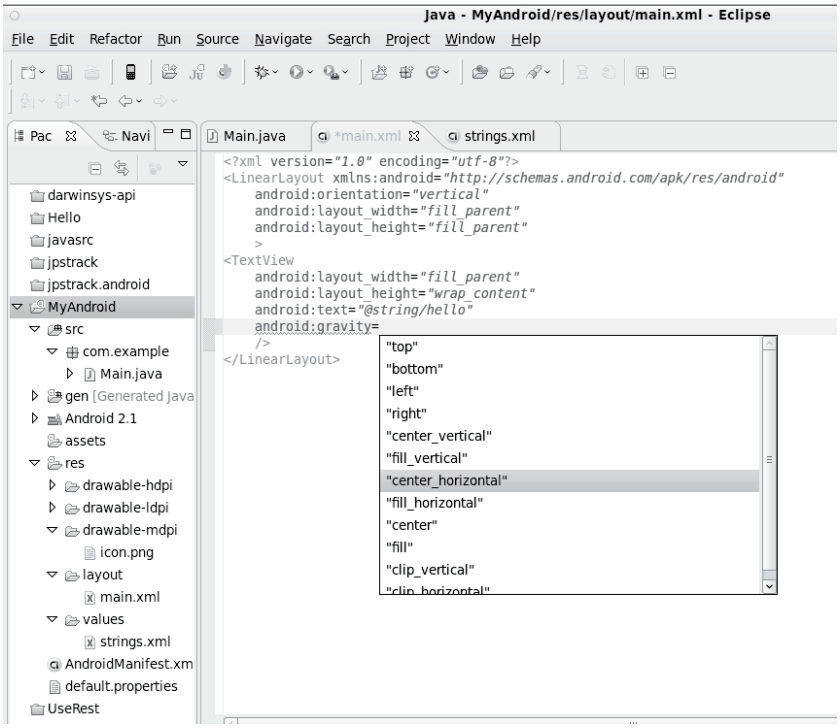


Figura 1.4 – Uso do editor do Eclipse para definir gravity em um TextView.

Na realidade, se você definir `gravity` como `center_vertical` no `LinearLayout` e defini-lo como `center_horizontal` na `TextView`, o texto estará centralizado tanto vertical quanto horizontalmente. O exemplo 1.3 é o arquivo de layout *main.xml* (localizado em *res/layout*) que realiza isso.

Exemplo 1.3 – Layout do XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_vertical"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:gravity="center_horizontal"
    />
</LinearLayout>
```

Como sempre, o Eclipse gera uma versão compilada sempre que você salva um arquivo-fonte. Além disso, em um projeto Android, ele também executa uma compilação Ant para criar o APK compilado e empacotado que estará pronto para ser executado. Por isso você precisa apenas executá-lo. Clique com o botão direito no projeto em si e selecione **Run As -> Android Project** (veja a figura 1.5.).

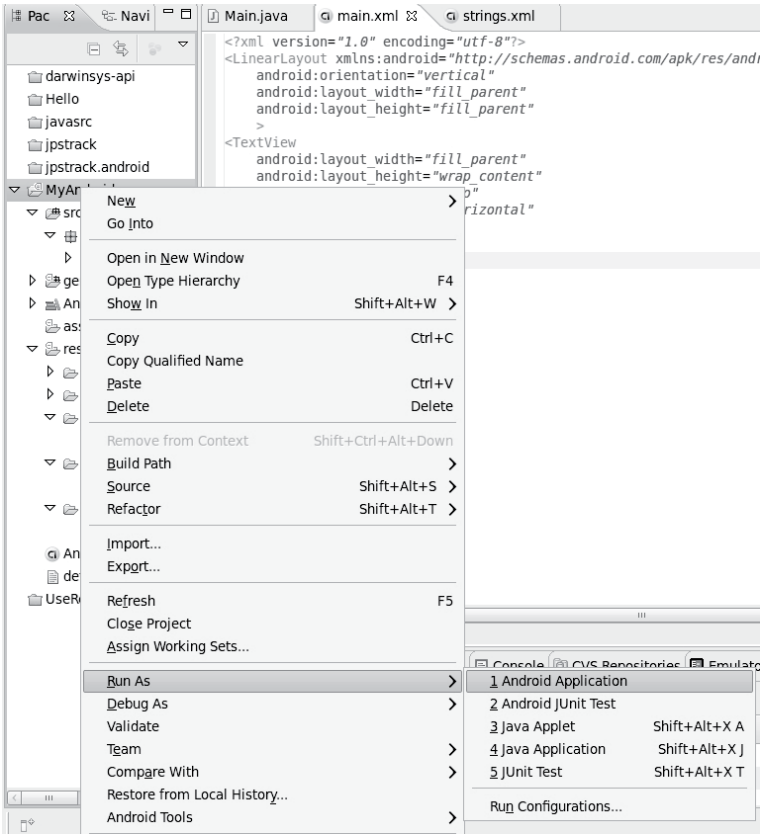


Figura 1.5 – Executando um projeto Android Eclipse.

Isso iniciará o emulador do Android se ele ainda não estiver sendo executado. O emulador começará com a palavra *Android* em texto simples, depois trocará para a fonte Android mais elaborada, com um detalhe em branco se movendo sobre as letras em azul – lembra da inicialização do Microsoft Windows 95 (veja a figura 1.6).

Depois de um pouco mais de tempo, seu aplicativo deverá inicializar. A figura 1.7 mostra apenas uma tela do aplicativo em si, já que o restante da visão do emulador é redundante.

Veja também

Receita 1.3.

Figura 1.6 – O projeto Android sendo inicializado no emulador.

Figura 1.7 – Projeto Eclipse sendo executado no emulador.

1.5 Configurando um IDE no Windows para desenvolvimento Android

Daniel Fowler

Problema

Você deseja desenvolver seus aplicativos Android utilizando um PC Windows, por isso um guia conciso da configuração de um IDE para essa plataforma seria útil.

Solução

O uso do IDE Eclipse é recomendado no desenvolvimento de aplicativos Android. Configurar o Eclipse no Windows não é uma instalação com um único passo; vários estágios precisam ser completados. Esta receita fornece detalhes sobre esses estágios.

Discussão

No desenvolvimento de aplicativos para Android, o IDE (Integrated Development Environment) Eclipse para Java é recomendado. O plugin ADT (Android Development Tools) está disponível para aprimorar o Eclipse, o qual utiliza o SDK do Android, fornecendo programas essenciais para desenvolvimento de software para o Android. Para configurar um ambiente de desenvolvimento, você precisa efetuar o download e a instalação destes componentes:

- Java Standard Edition Development Kit
- Eclipse para desenvolvimento Java
- SDK do Android
- plugin ADT (dentro do Eclipse)

Nas subseções a seguir abordaremos esses estágios detalhadamente para um PC Windows (testado em XP, Vista e Windows 7).

Instalação do JDK (Java Development Kit)

Vá até a página de downloads do Java, em *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

Selecione o ícone do Java para acessar os downloads do JDK:

A lista de downloads do JDK será mostrada. Clique no botão de opção **Accept License Agreement**; do contrário, você não poderá continuar. Faça o download e execute o JDK mais recente presente; quando da elaboração deste texto, eles eram o *jdk-7u2-windows-i586.exe* (ou o *jdk-7u2-windows-x64.exe* para o Windows 64 bits). Você pode precisar selecionar a localização do site de download. Aceite quaisquer avisos de segurança que apareçam, mas apenas se você estiver efetuando o download na página oficial de downloads do Java.

Quando o download tiver terminado e for executado, você terá de percorrer as telas de instalação, clicando **Next** até que o instalador do JDK tenha concluído. Você não deve ter de alterar nenhuma das opções apresentadas. Quando o instalador do JDK tiver concluído, clique no botão **Finish**. Pode ser que uma página de registro de produto seja carregada; você pode fechá-la ou escolher registrar sua instalação.

Instalação do Eclipse para desenvolvimento Java

A página de download do Eclipse está em *http://www.eclipse.org/downloads/*.

O Windows precisa ser selecionado na lista suspensa **Packages**; selecione o link relevante de download do IDE Eclipse para desenvolvedores Java (veja a figura 1.8).

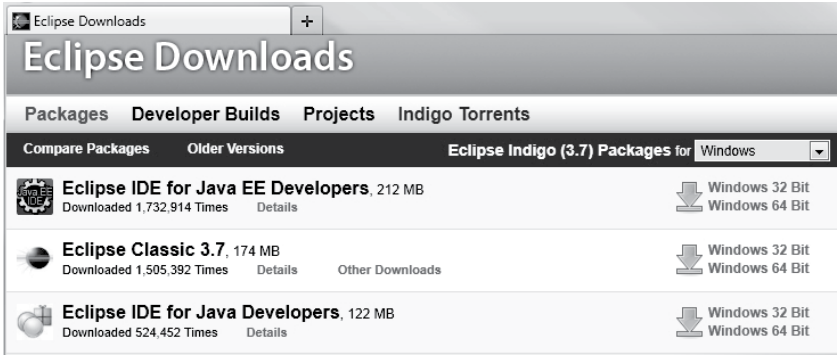


Figura 1.8 – Escolha de um download do Eclipse.

Faça o download e abra o arquivo ZIP. No arquivo haverá um diretório *eclipse* contendo vários arquivos e subdiretórios. Copie o diretório *eclipse* e todo o seu conteúdo como fornecido (Figura 1.9). O local habitual para o qual copiar os arquivos é na raiz do drive C ou dentro de *C:\Arquivos de Programas*. Você pode ter de selecionar **Continuar** quando o Windows pedir permissão para a cópia.

Crie um atalho de desktop para *eclipse.exe*.

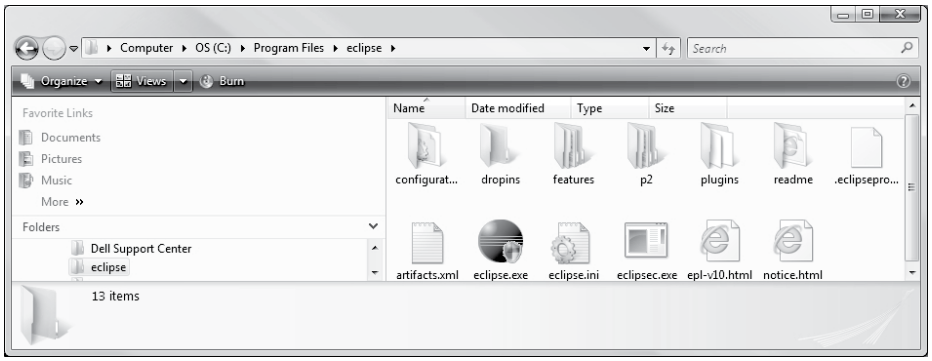


Figura 1.9 – Conteúdo da pasta do Eclipse.

Execute o Eclipse para que ele configure um workspace (espaço de trabalho); fazer isso também verificará que tanto o Java quanto o Eclipse foram instalados corretamente. Quando você executar o Eclipse, pode ser que um aviso de segurança seja mostrado; selecione Run para continuar. Aceite a localização padrão do workspace ou utilize um diretório diferente.

Instalação do SDK do Android

Vá até a página de download do SDK do Android em *http://developer.android.com/sdk/index.html*.

Escolha o pacote EXE mais recente para Windows (no momento em que escrevo isso, *installer_r16-windows.exe*) e selecione Run. Aceite o aviso de segurança apenas se você estiver efetuando o download na página oficial do site do SDK do Android. O instalador Android SDK Tools mostrará algumas telas. Selecione o botão Next em cada uma delas; você não deve ter de alterar nenhuma opção. Uma vez que *C:\Arquivos de Programas* é um diretório protegido, você pode ou obter permissão para instalar nele, ou, como fazem alguns desenvolvedores, instalar em sua pasta de usuário ou outro diretório – por exemplo, *C:\Android\android-sdk*.

Quando o botão Install for clicado, uma tela de progresso aparecerá brevemente enquanto os arquivos do Android são copiados. Clique no último botão Next e no botão Finish ao final da instalação. Se você deixou a caixa Start SDK Manager selecionada, o SDK Manager será executado. Do contrário, selecione o SDK Manager a partir do grupo de programas Android SDK Tools (Iniciar->Todos os Programas->Android SDK Tools->SDK Manager). Quando o SDK Manager iniciar, os pacotes Android disponíveis para download serão verificados. Então, uma lista de pacotes disponíveis será mostrada com alguns deles pré-selecionados para download. Uma coluna de Status mostrará se um pacote está instalado ou não. Na figura 1.10 você pode ver que as Android SDK Tools acabaram de ser instaladas e isso está refletido na coluna Status.

Marque cada pacote que precisa ser instalado. Vários pacotes estão disponíveis. Estes incluem pacotes da plataforma SDK para cada nível de API, exemplos de aplicativos para a maioria dos níveis de API, APIs do Google Maps, APIs de fabricantes específicos, documentação, código-fonte e os pacotes extras a seguir para o Google:

Android Support

Utilizado para oferecer suporte a APIs Android mais recentes em dispositivos mais antigos.

AdMob Ads SDK

Para incorporação de publicidade em aplicativos.

Analytics SDK

Para oferecer suporte à análise das compras de clientes.

Market Billing

Acrescenta suporte para compras dentro do aplicativo.

Market Licensing

Ajuda a proteger aplicativos de serem ilegalmente copiados.

USB Driver

Para depuração em dispositivos físicos (ou utilizando um driver do fabricante).

Webdriver

Ajuda a testar a compatibilidade de um site com o navegador Android.

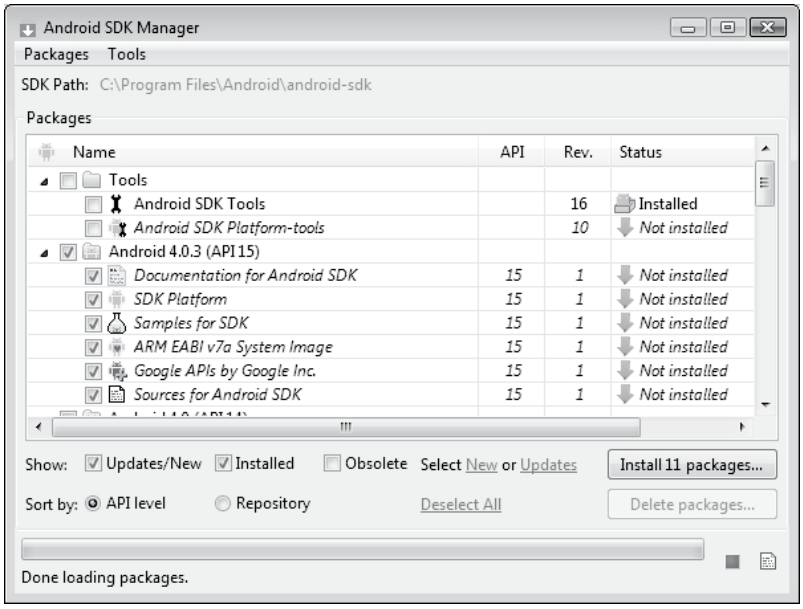


Figura 1.10 – Android SDK Manager, mostrando componentes instalados e disponíveis para download.

É recomendado que você efetue o download de várias plataformas de SDK para permitir testes de aplicativos em várias configurações de dispositivo. Vale notar que computadores mais antigos terão dificuldade em executar os emuladores de dispositivos virtuais para as APIs mais recentes do Android; portanto, nesses computadores, desenvolva utilizando as plataformas SDK mais antigas. Se estiver em dúvida quanto a qual download efetuar, aceite as escolhas iniciais e execute novamente o SDK Manager para obter outros pacotes como e quando necessário; ou marque todos os pacotes para efetuar o download de tudo (o download pode demorar um pouco). Clique no botão "Install packages".

Os pacotes selecionados serão mostrados em uma lista; se um pacote tiver termos de licença que requerem aceitação, ele será mostrado com um ponto de interrogação. Destaque cada pacote que tem um ponto de interrogação para ler os termos de licença. Você pode aceitar ou rejeitar o pacote utilizando os botões de opção. Pacotes rejeitados são marcados com um × vermelho. Alternativamente, clique em Accept All para aceitar tudo que está disponível Clique no botão Install e uma barra de progresso mostrará os pacotes sendo instalados, bem como quaisquer erros que ocorram. No Windows, um erro comum ocorre quando o SDK Manager é incapaz de acessar ou de renomear diretórios. Execute novamente o SDK Manager como administrador e verifique se o diretório não tem nenhuma flag ou nenhum arquivo apenas para leitura; veja a receita 1.12 para mais detalhes. Quando tiver concluído, feche o SDK Manager clicando no botão × no canto superior da janela.

Instalação do plugin ADT

A instalação do plugin ADT é feita via Eclipse, mas para isso você precisa executar o Eclipse a partir da conta de administrador. Utilize o atalho criado antes ou o *eclipse.exe* na pasta *eclipse*. Em qualquer um dos casos, abra o menu de contexto (geralmente por meio de um clique com o botão direito), selecione "Executar como administrador" e aceite quaisquer avisos de segurança. Quando o Eclipse tiver carregado, abra o item de menu Help e selecione Install New Software….

Na tela de instalação, digite este endereço na caixa "Work with":

https://dl-ssl.google.com/android/eclipse/

Clique no botão Add. Uma tela Add Repository aparecerá; na caixa Name digite algo significativo, como "ADT plug-in" (o endereço web mencionado antes será mostrado na caixa Location); veja a figura 1.11.

Clique no botão OK. A tela será atualizada depois de brevemente mostrar "Pending" na coluna Name da tabela.

Marque a caixa ao lado de Developer Tools. Então, selecione o botão Next na parte inferior da tela (veja figura 1.12).

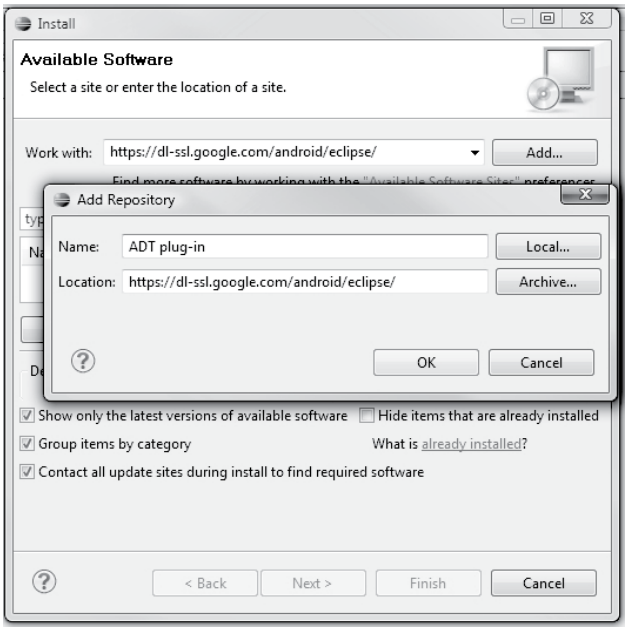


Figura 1.11 – Inclusão do repositório "ADT plug-in".

Uma lista de itens a serem instalados será mostrada. Se você receber uma mensagem de erro, verifique se o Eclipse foi executado na conta do administrador. Selecione **Next** novamente. Uma tela mostrará as licenças; certifique-se de que cada licença tenha sido aceita (selecione o botão de opção "I accept the terms of the license agreements"). Então clique no botão **Finish**. Um aviso de segurança terá de ser aceito para concluir a instalação; selecione **OK** para esse aviso (o endereço digitado antes é um endereço seguro). O Eclipse pedirá uma reinicialização. Selecione o botão **Restart Now** e o Eclipse vai fechar e recarregar. Uma caixa de diálogo "Welcome to Android Development" vai aparecer. Defina a localização do SDK na caixa **Existing Location** (uma vez que o SDK Manager já foi executado), vá até a pasta do SDK do Android (por padrão, *C:\Arquivos de Programas\Android\android-sdk*), e clique em **Next** (veja a figura 1.13).

Uma pergunta de monitoramento de uso do Google Android SDK vai aparecer; modifique a opção se necessário e clique em **Finish**. O Eclipse estará agora configurado para compilar e depurar aplicativos Android. Veja a receita 3.3 para configurar um emulador Android; depois experimente a receita 1.4 como um teste de sanidade. Conecte um dispositivo físico no computador e utilize suas configurações para ativar o **USB Debugging** (em **Development**, dentro de **Applications**).

Veja também

Receita 1.4; Receita 1.12; Receita 3.3; *http://developer.android.com/sdk/installing.html*, *http://www.eclipse.org/*; *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

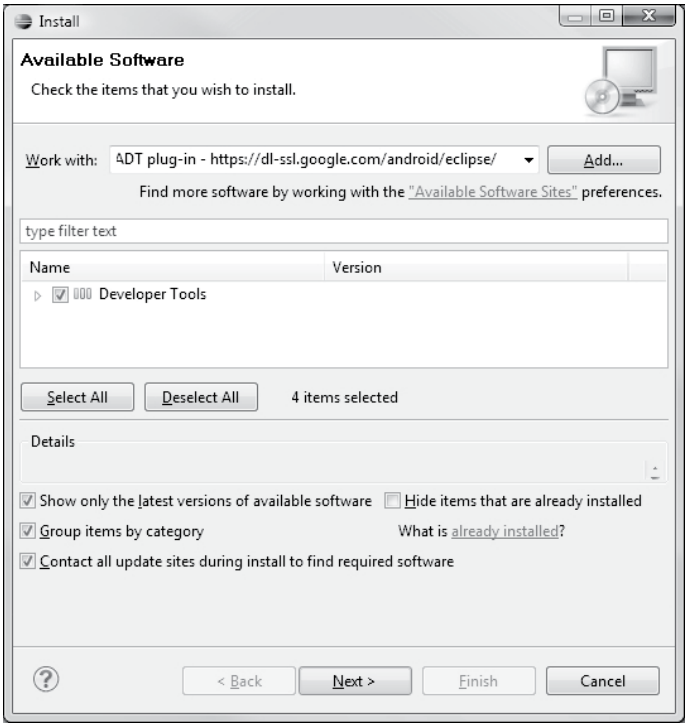


Figura 1.12 – Escolhendo o que instalar.

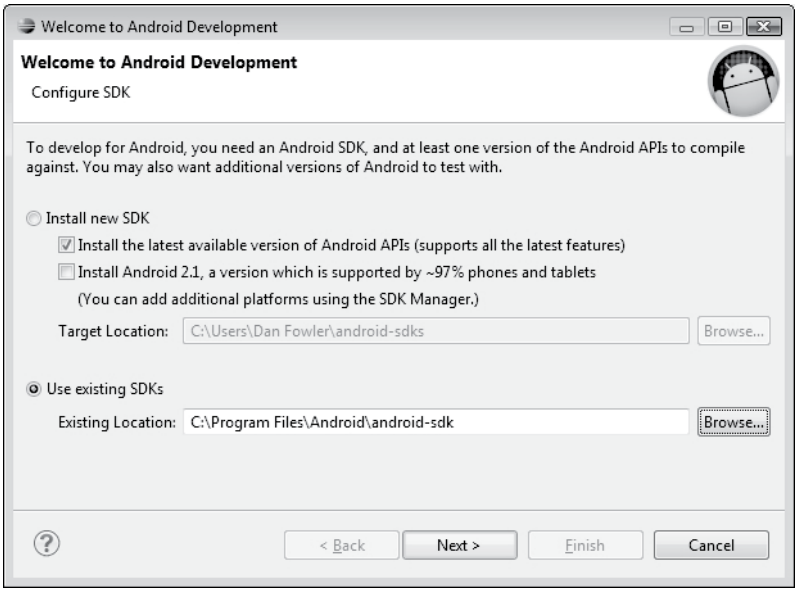


Figura 1.13 – Conexão do recém-instalado SDK ao recém-instalado plugin ADT.

1.6 Entendendo o ciclo de vida do Android

Ian Darwin

Problema

Aplicativos Android não têm um método "main"; você precisa aprender como eles começam e como param ou são parados.

Solução

A classe `android.Activity` fornece vários métodos de ciclo de vida bem definidos que são chamados quando um aplicativo é iniciado, suspenso, reiniciado e assim por diante, assim como um método que você pode chamar para marcar uma atividade como concluída.

Discussão

Seu aplicativo Android executa seu próprio processo Unix, por isso, em geral, ele não pode afetar diretamente nenhum outro aplicativo em execução. A VM Dalvik faz a interface com o sistema operacional para chamar você quando seu aplicativo inicia, quando o usuário muda para outro aplicativo e assim por diante. Há um ciclo de vida bem definido para aplicativos Android.

Um aplicativo Android tem três estados em que pode estar:

- Ativo, no qual o aplicativo está visível para o usuário e em execução.
- Pausado, no qual o aplicativo está parcialmente obscurecido e perdeu o foco de entrada.
- Parado, no qual o aplicativo está completamente oculto da visão.

Seu aplicativo percorrerá esses estados quando o Android chamar os métodos a seguir na atividade atual, no momento apropriado:

```
void onCreate(Bundle savedInstanceState)
void onStart()
void onResume()
void onRestart()
void onPause()
void onStop()
void onDestroy()
```

Você pode ver o diagrama de estado desse ciclo de vida na figura 1.14.

No caso da primeira atividade de um aplicativo, `onCreate()` é a forma como você sabe que o aplicativo foi iniciado. É aqui que você normalmente realiza trabalhos de tipo

construtor, como definir a "janela principal" com `setContentView()`, incluir ouvintes (listeners) para botões que realizem trabalhos (inclusive iniciar atividades adicionais) e assim por diante. Esse é o único método que até mesmo o mais simples dos aplicativos Android precisa.

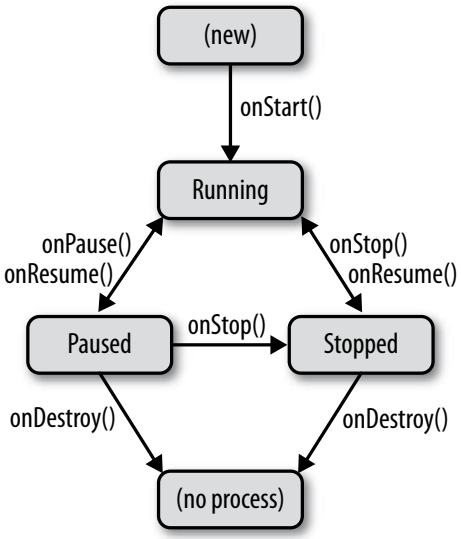


Figura 1.14 – Estados de ciclo de vida do Android.

Você pode ver os efeitos dos vários métodos de ciclo de vida criando um projeto de teste no Eclipse e sobrescrevendo todos os métodos utilizando instruções de log para fins de "depuração".

1.7 Instalação de arquivos .apk em um emulador por meio do ADB

Rachee Singh

Problema

Você tem um arquivo *.apk* de um aplicativo, e deseja instalá-lo no emulador para conferir esse aplicativo, ou porque um aplicativo que você está desenvolvendo o requer.

Solução

Utilize a ferramenta de linha de comando ADB para instalar o arquivo *.apk* no emulador em execução; você também pode utilizar essa ferramenta para instalar um arquivo *.apk* em um dispositivo Android conectado.

Discussão

Para instalar o arquivo *.apk*, siga estes passos:

1. Encontre o local em sua máquina onde você instalou o SDK do Android. No diretório do SDK do Android, vá até o diretório *tools*.
2. Procure um executável chamado *adb* no diretório *tools*. Se ele estiver presente, essa é a localização do arquivo *adb*; do contrário, deve haver um arquivo *.txt* chamado "adb has moved". O conteúdo desse arquivo meramente direciona você para a localização do binário do *adb*; o arquivo afirma que o *adb* está presente no diretório *platform-tools* em vez de estar no diretório *tools*.
3. Assim que você tiver localizado o programa *adb*, use o comando `cd` para chegar a essa localização em um terminal (Linux) ou prompt de comando (Windows).
4. Utilize o comando `adb install` *`localização do .apk que você deseja instalar`*. Se você receber "command not found" no Linux, tente utilizar "./adb" em vez de apenas "adb".

Isso deve iniciar a instalação no dispositivo que está atualmente sendo executado (seja um emulador em execução em seu desktop, ou um dispositivo físico Android conectado).

Depois de a instalação terminar, no menu do dispositivo/emulador Android, você deve ver o ícone do aplicativo que acabou de instalar (veja a figura 1.15).

Figura 1.15 – Comando da instalação.

1.8 Instalação de aplicativos em um emulador via SlideME

David Dawes

Problema

As *app stores* (em uma tradução livre, lojas de aplicativos) são uma enorme parte da atração dos smartphones modernos. O Android Market[1] do Google é a *app store* oficial, mas você também pode querer utilizar outras.

Solução

A SlideMe LLC oferece uma *app store* alternativa. A *app store* SlideME permite que você instale outros aplicativos (talvez você queira integrar com outros aplicativos), bem como que você teste a experiência de publicar e efetuar o download de seus próprios aplicativos em seu dispositivo emulado Android. A SlideME também alcança muitos usuários Android que estão fora do Android Market, incluindo pessoas com dispositivos não suportados e aqueles que não vivem em um país onde há suporte do Android Market.

Discussão

Uma alternativa ao Android Market oficial é a SlideME, uma *app store* alternativa. A SlideME pode não ter tantos aplicativos quanto o Android Market do Google, mas tem algumas vantagens, incluindo o fato de que funciona facilmente em um dispositivo Android emulado.

Vá até o site da SlideME utilizando seu dispositivo Android emulado, navegue ou busque aplicativos e clique em um aplicativo gratuito. Depois de uma pausa para download do aplicativo, abra o download (a pequena seta no canto superior esquerdo), revise a licença e inicie o arquivo *.apk* do qual você efetuou o download para instalar o aplicativo. Durante a instalação, será pedido que você revise e aceite a licença para o software.

Assim que o aplicativo da SlideME tiver sido instalado, você poderá percorrer o catálogo e instalar mais aplicativos sem utilizar o navegador. Isso é muito mais fácil do que utilizar um navegador web para efetuar o download dos aplicativos, já que a apresentação é projetada para o dispositivo Android; simplesmente escolha uma categoria, navegue dentro dela, e escolha um aplicativo para instalar. Eu tive alguns problemas de estabilidade utilizando o aplicativo em meu emulador – ele congelou em alguns casos –, mas fui capaz de instalar alguns aplicativos básicos gratuitos, como o Grocery List.

1 N.T.: Android Market era o nome anterior da loja de aplicativos do Google, que hoje, unificada às lojas de música, filmes e livros, é chamada de Google Play (fonte: Wikipédia).

Percebi, no fórum de discussão Android Invasion no *Linkedin.com*, que alguns usuários do Android ficaram desapontados ao perceber que muitos provedores de telefones celulares não incluem o Android Market oficial em suas ofertas de telefones celulares Android, e, a não ser que você se sinta à vontade em fazer o root[2] e atualizar seu telefone Android, não há como acessá-lo. A maioria dos clientes não se sente confortável em fazer o root e atualizar seus telefones, e, para eles, a SlideME oferece um modo alternativo de encontrar aplicativos gratuitos e baratos para seus telefones.

Veja também

A SlideME também permite que você publique seus aplicativos em sua *app store*; veja a página Applications no site da SlideME.

Para informações sobre o desenvolvimento de aplicativos para a SlideME, consulte *http://slideme.org/developers*.

1.9 Compartilhando classes Java de outro projeto Eclipse

Ian Darwin

Problema

Você deseja utilizar uma classe de outro projeto, mas não quer copiar e colar.

Solução

Inclua o projeto como um "projeto referenciado", e o Eclipse (e o DEX) cuidarão do trabalho.

Discussão

Você muitas vezes precisa reutilizar classes de outro projeto. Em meu programa de monitoramento GPS, o JPSTrack, a versão do Android empresta classes como o módulo de I/O de arquivos da versão Java SE. Você certamente não deseja copiar e colar classes inalteradas de um projeto para outro, pois isso torna a manutenção improvável.

No caso mais simples, quando o projeto de biblioteca contém o código-fonte das classes que você deseja importar, tudo que você tem de fazer é declarar o projeto que contém as classes necessárias (a versão Java SE neste caso) como um projeto referenciado no path de compilação. Selecione **Project->Properties->Java Build Path**, selecione **Projects**, e clique em **Add**. Na figura 1.16, eu estou incluindo o projeto SE "jpstrack" como uma dependência no projeto Android "jpstrack.android."

2 N.T.: O rooting é um processo que permite a usuários de smartphones, tablets, e outros dispositivos que executam o sistema operacional Android, obter controle privilegiado (conhecido como "acesso à raiz") dentro do subsistema do Android (fonte: Wikipédia).

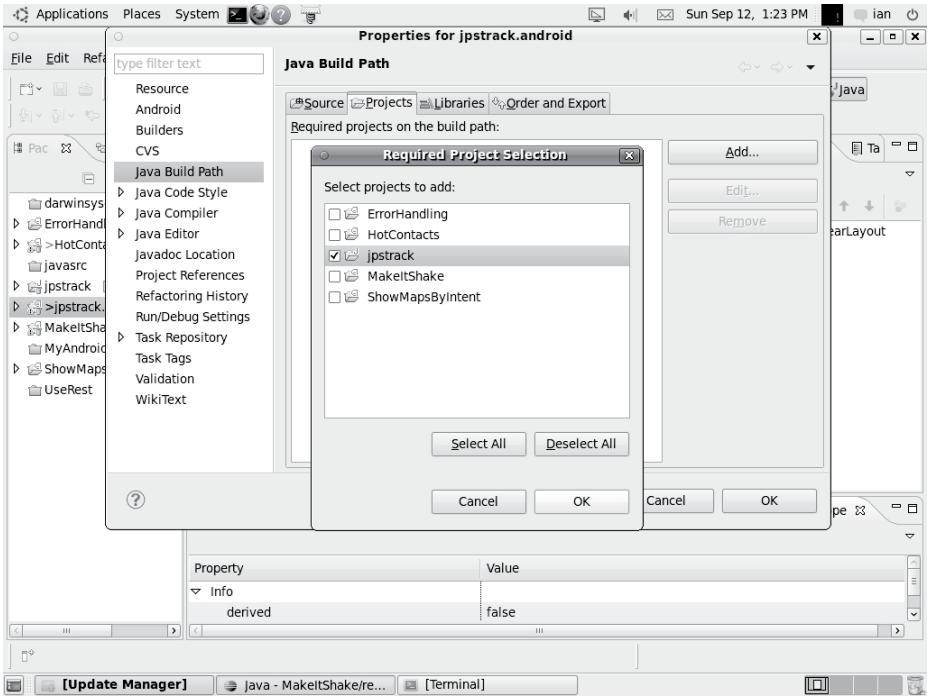


Figura 1.16 – Tornando um projeto dependente de outro – utilizando Eclipse padrão.

Desenvolvedores móveis que também criam aplicativos para outras plataformas devem notar que essa técnica não funciona se você tem o plugin BlackBerry Java atual (do final de 2011) instalado em sua instalação Eclipse. Esse é um bug no plugin BlackBerry Java; ele sinaliza incorretamente até mesmo projetos não BlackBerry como dependentes de projetos de biblioteca não BlackBerry, e marca o projeto como tendo um erro, o que impede a geração e execução correta do código. Remova o plugin defeituoso ou coloque-o em uma instalação separada do Eclipse.

Como alternativa, crie um arquivo JAR utilizando o Ant ou o assistente do Eclipse. Faça com que o outro projeto referencie esse arquivo como um JAR externo nas configurações do classpath. Ou copie fisicamente este arquivo para o diretório *libs* e faça referência a ele a partir desse local.

Um método mais novo que é frequentemente mais confiável, e agora oficialmente recomendado, mas útil apenas se ambos os projetos forem projetos Android, é declarar o projeto de biblioteca como tal na guia **Project->Properties->Android->Library** (marque o botão **Is Library**), e utilizar o botão **Add** no outro projeto, na mesma tela, para listar o projeto de biblioteca como uma dependência do projeto principal (veja a figura 1.17).

Para fãs da linha de comando, o primeiro método envolve a edição do arquivo *.classpath*, enquanto o segundo simplesmente cria entradas no arquivo *project.properties*, por exemplo:

```
# Projeto alvo
target=android-7
android.library=false
android.library.reference.1=../wheel
```

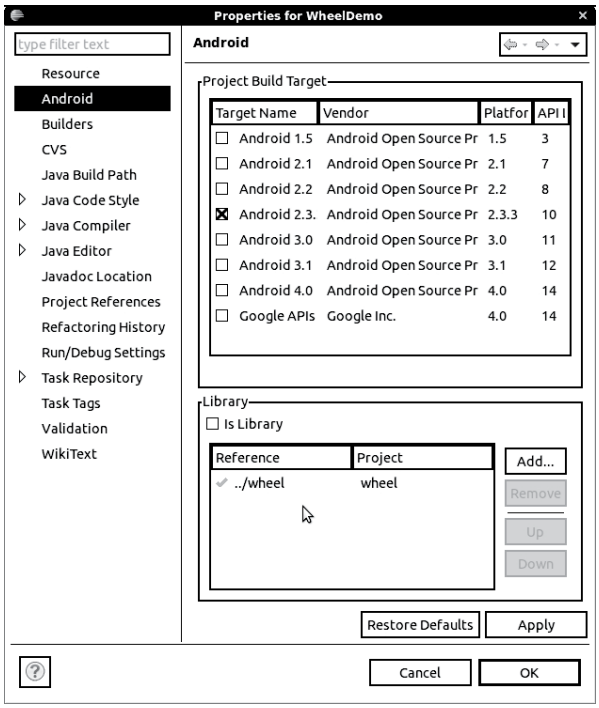


Figura 1.17 – Tornando um projeto dependente de outro – utilizando ADT.

Como você provavelmente está mantendo ambos os projetos em controle de versão (e se forem programas que em algum momento você pretende lançar, você deve!), lembre-se de "marcar" ambos os projetos quando lançar o projeto Android – um dos pontos a favor do controle de versão é que você é capaz de recriar exatamente o que entregou.

Veja também

Consulte a documentação oficial sobre projetos de biblioteca.

1.10 Referenciando bibliotecas para implementar funcionalidades externas

Rachee Singh

Problema

Você precisa referenciar uma biblioteca externa em seu código-fonte.

Solução

Obtenha o arquivo JAR da biblioteca requerida e inclua-o em seu projeto.

Discussão

Como exemplo, você pode ter de utilizar o AndroidPlot, uma biblioteca para representação de diagramas e gráficos em seu aplicativo, ou o OpenStreetMap, um projeto wiki que cria e fornece dados e mapeamento geográficos gratuitos. Se afirmativo, seu aplicativo precisa referenciar essas bibliotecas. Você pode fazer isso no Eclipse em poucos passos simples:

1. Efetue o download do arquivo JAR correspondente à biblioteca que você deseja utilizar.
2. Depois de criar seu projeto Android no Eclipse, clique com o botão direito sobre o nome do projeto e selecione **Properties** no menu (Figura 1.18).
3. Na lista do lado esquerdo, selecione **Java Build Path** e clique na guia **Libraries**.
4. Clique no botão **Add External JARs**.
5. Forneça o local onde você efetuou o download do arquivo JAR da biblioteca que você deseja utilizar.

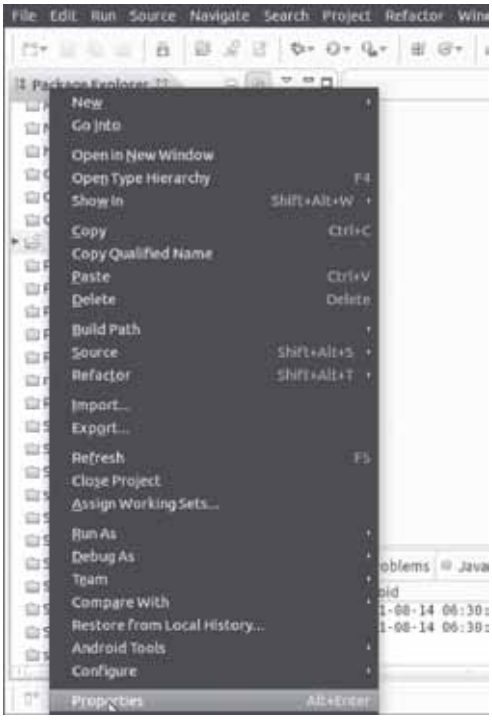


Figura 1.18 – Seleção de propriedades do projeto.

Nesse ponto, você verá um diretório *Referenced Libraries* em seu projeto. Os JARs que você incluiu vão aparecer (veja a figura 1.19).

Uma abordagem alternativa é criar uma pasta *lib* em seu projeto, copiar fisicamente os arquivos JAR para ela e incluí-los individualmente como você fez antes, mas, em vez disso, clicando no botão **Add JARs**. Isso mantém tudo em um lugar só (especialmente se seu projeto for compartilhado por meio de um sistema de controle de versão com outros que podem utilizar um sistema operacional diferente e incapaz de utilizar os JARs externos no mesmo local). No entanto, isso eleva o fardo de responsabilidade para questões de licenciamento referentes aos arquivos JAR incluídos (veja a figura 1.20).

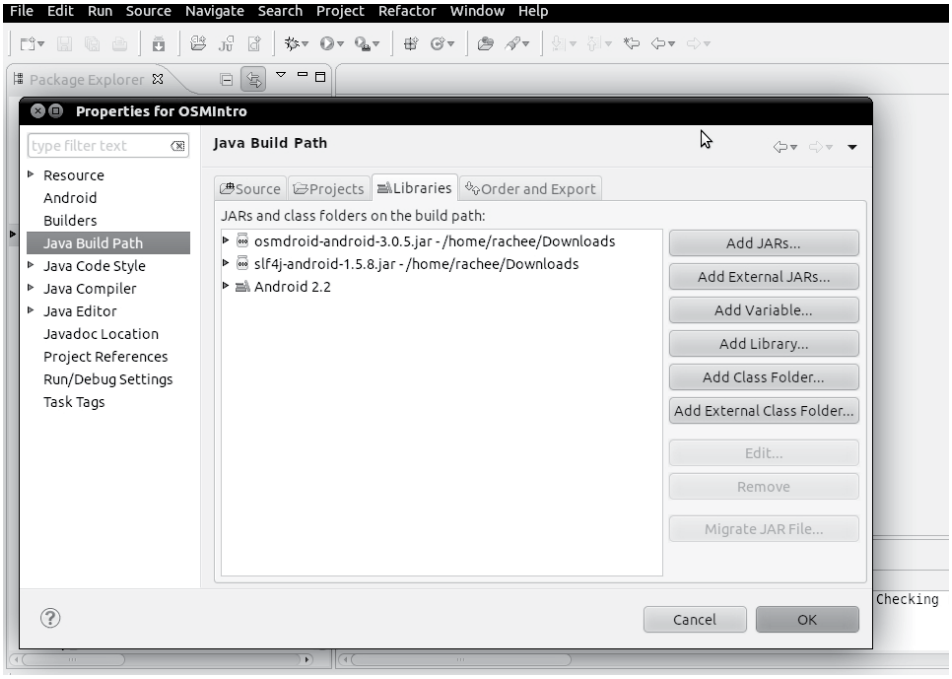


Figura 1.19 – Inclusão de bibliotecas.

Em qualquer um dos casos, se você também compilar com Ant, certifique-se de atualizar seu arquivo *build.xml*.

Qualquer que seja o modo escolhido é bastante fácil incluir bibliotecas em seu projeto.

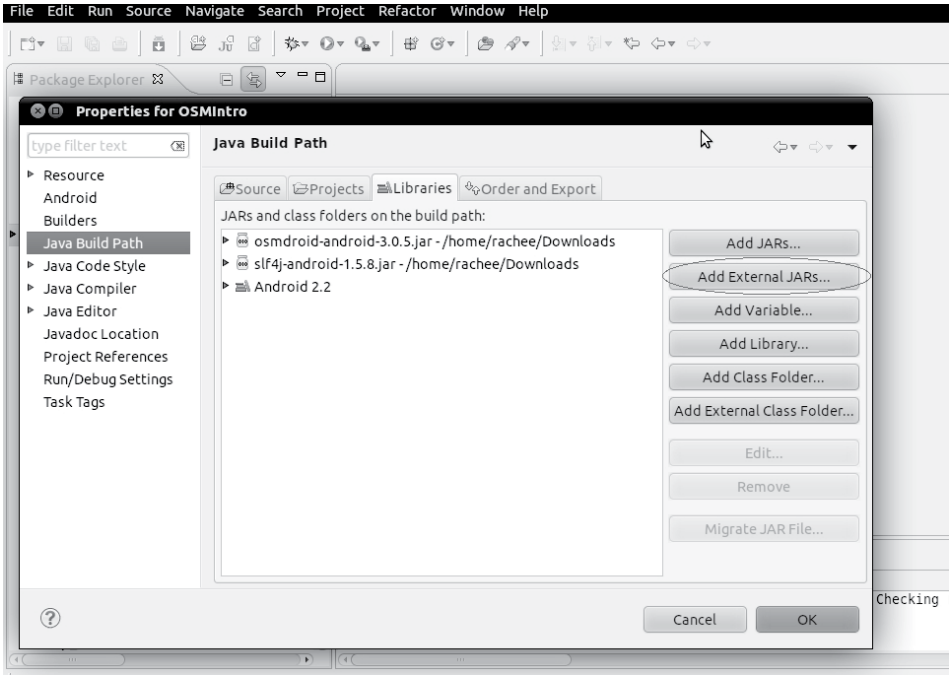


Figura 1.20 – Inclusão do arquivo JAR externo.

1.11 Uso de exemplos do SDK para ajudar a evitar situações de dúvida

Daniel Fowler

Problema

Às vezes é difícil codificar determinada funcionalidade, especialmente quando a documentação é imprecisa ou não fornece nenhum exemplo.

Solução

Analisar códigos funcionais já existentes vai ajudar. O SDK do Android tem programas de exemplo que você pode destrinchar para ver como funcionam.

Discussão

O SDK do Android vem com vários aplicativos de exemplo que podem ser úteis quando você estiver tentando codificar alguma funcionalidade. Analisar o código do exemplo pode ser esclarecedor. Assim que você tiver instalado o SDK do Android, vários exemplos se tornarão disponíveis:

- Accelerometer Play
- Accessibility Service
- API Demos
- Backup and Restore
- Bluetooth Chat
- Business Card
- Contact Manager
- Cube Live Wallpaper
- Home
- Honeycomb Gallery
- JetBoy
- Lunar Lander
- Multiple Resolutions
- Near Field Communication
- Note Pad
- RenderScript
- Sample Sync Adapter
- Searchable Dictionary
- Session Initiation Protocol
- Snake
- Soft Keyboard
- Spinner
- SpinnerTest
- StackView Widget
- TicTacToeLib
- TicTacToeMain
- USB
- Wiktionary
- Wiktionary (Simplified)
- Weather List Widget
- XML Adapters

Para abrir um projeto de exemplo a partir do Eclipse, abra o menu **File** e então selecione **Android Project** (veja a figura 1.21).

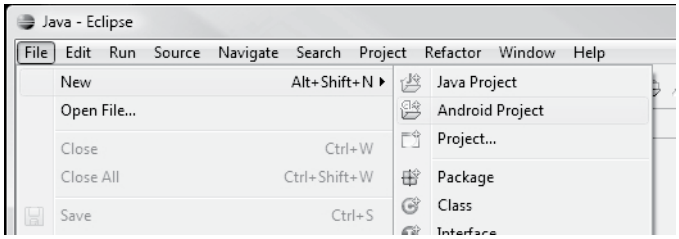
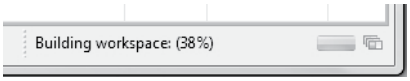


Figura 1.21 – Iniciando um novo projeto Android.

No diálogo **New Android Project**, selecione a opção **Create project from existing sample**. Clique em **Next** e selecione o alvo de compilação. Uma lista de exemplos disponíveis para o alvo selecionado será mostrada. Se o exemplo necessário não for mostrado, volte e selecione outro alvo de compilação. (O exemplo pode não estar instalado; o SDK Manager pode ser utilizado para instalar exemplos adicionais se eles estiverem faltando durante o setup do SDK.) Escolha o exemplo a ser carregado, clique em **Finish**, e o exemplo será copiado para o Workspace e compilado (com o progresso mostrado na barra de status).



Depois de pouco tempo o exemplo estará pronto para ser executado e você será capaz de explorar o código-fonte para ver como tudo é feito.

Se os exemplos foram movidos do diretório *samples* do SDK, utilize a opção "Create project from existing source" no diálogo **New Android Project** para abrir o exemplo.

Quando o exemplo for executado pela primeira vez, selecione **Android Application** no diálogo **Run As** que deve aparecer. Também pode ser necessário configurar um AVD apropriado para executar o exemplo (veja a receita 3.3). Veja a figura 1.22.

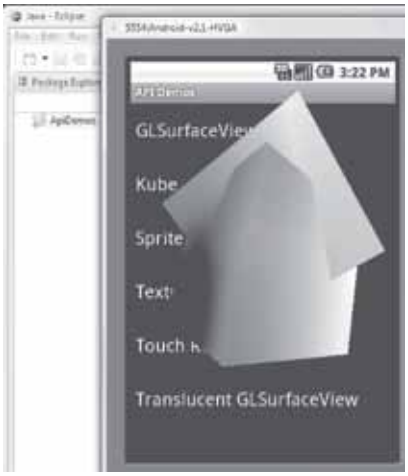


Figura 1.22 – Demonstrações da API em ação.

Veja também

O site Android Developers em *http://developer.android.com/index.html*; este cookbook, é claro.

Você também pode pesquisar a Web em busca de programas ou exemplos adicionais. Se você ainda não conseguiu encontrar o que precisa, pode buscar ajuda no Stack Overflow (*http://www.stackoverflow.com*; utilize "android" como tag) ou no canal IRC (Internet Relay Chat) `#androiddev` no `freenode`.

1.12 Mantendo o SDK do Android atualizado

Daniel Fowler

Problema

O SDK deve ser mantido atualizado para permitir que desenvolvedores de aplicativos trabalhem com as APIs mais recentes na plataforma Android que está sempre em evolução.

Solução

Utilize o programa Android SDK Manager para atualizar os pacotes SDK instalados existentes e instalar novos pacotes SDK. Isso inclui pacotes de terceiros para funcionalidades de dispositivos específicos.

Discussão

O sistema operacional (SO) Android está evoluindo constantemente, e, portanto, o mesmo vale para o SDK do Android. O desenvolvimento constante do Android é motivado por estes fatores:

- A pesquisa e desenvolvimento do Google.
- Fabricantes de telefones estarem sempre desenvolvendo dispositivos novos e melhorados.
- A resolução de problemas de segurança e possíveis *exploits*[3].
- A necessidade do suporte a novos dispositivos (por exemplo, o suporte a dispositivos tablet foi incluído com a versão 3.0).
- O suporte a novas interfaces de hardware (por exemplo, o suporte a comunicação de campo próximo foi incluído na versão 2.3).
- A correção de bugs.

3 N.T.: Um exploit, em segurança da informação, é um programa de computador, ou uma sequência de comandos que aproveita das vulnerabilidades de um sistema computacional (fonte: Wikipédia).

- Melhorias de funcionalidades (por exemplo, um novo engine JavaScript).
- Alterações no kernel Linux subjacente.
- A substituição de interfaces de programação redundantes.
- Novos usos (por exemplo, o Google TV).
- A comunidade de desenvolvimento Android como um todo.

Abordamos a instalação do SDK do Android em outro local (veja a Receita 1.5 ou *http://developer.android.com/sdk/installing.html*). Depois de o SDK ter sido instalado na máquina de desenvolvimento e de o ambiente de programação estar sendo executado sem problemas, desenvolvedores terão, de vez em quando, de verificar se há atualizações para o SDK.

Você pode manter o SDK atualizado executando o programa SDK Manager. (Em uma máquina Windows, execute o *SDK Manager.exe* na pasta *C:\Arquivos de Programas\Android\androidsdk*, ou utilize o menu **Iniciar**, e então selecione **Todos os Programas->Android SDK Tools**, e clique em **SDK Manager**). Você também pode executá-lo dentro do Eclipse (utilizando o menu **Window** e selecionando **Android SDK Manager**); veja a figura 1.23. O SDK do Android é dividido em vários pacotes. O SDK Manager procura automaticamente por atualizações para pacotes existentes e vai listar novos pacotes e aqueles fornecidos por fabricantes de dispositivos.

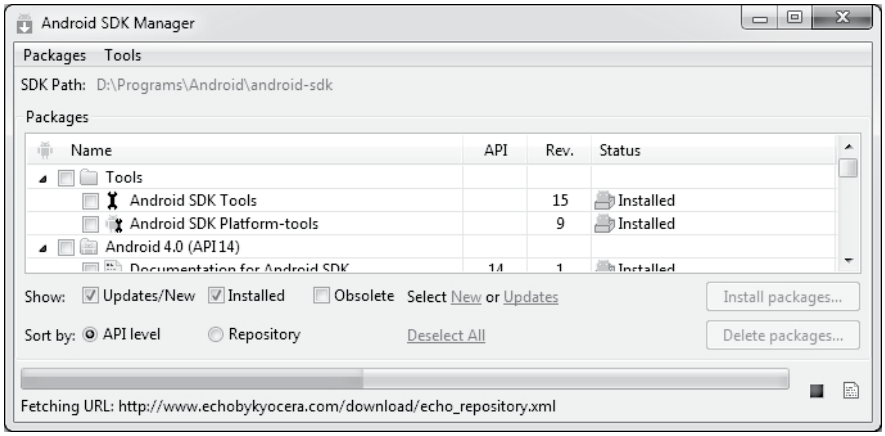


Figura 1.23 – SDK Manager do Android.

Atualizações disponíveis serão mostradas em uma lista (assim como pacotes opcionais disponíveis). Se uma atualização ou um pacote tiver termos de licença que requerem aceitação eles serão mostrados com um ponto de interrogação. Destaque cada pacote que tem um ponto de interrogação para ler os termos de licença. Você pode aceitar ou rejeitar o pacote utilizando os botões de opção. Pacotes rejeitados serão marcados com um × vermelho (veja a figura 1.24).

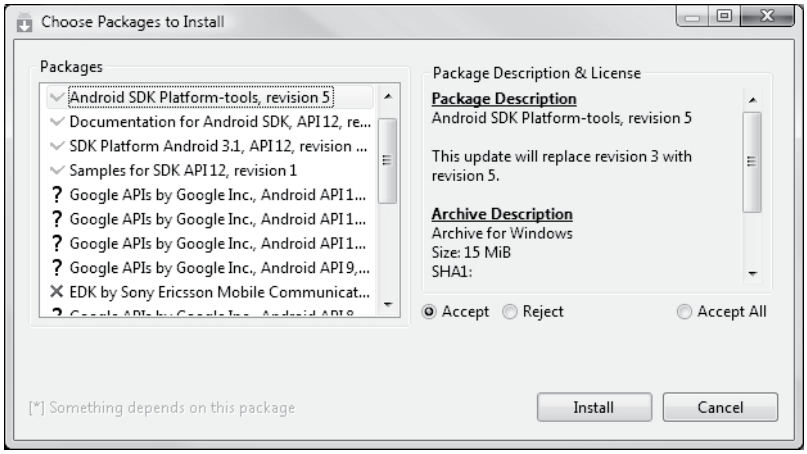


Figura 1.24 – Escolha de pacotes do SDK.

Como alternativa, clique em **Accept All** para aceitar tudo que estiver disponível. Todos os pacotes e atualizações que estiverem disponíveis para download e instalação serão mostrados com um sinal de visto verde.

Clique no botão **Install** para começar o download e a instalação; quando concluído, clique no botão **Close** (veja a figura 1.25).

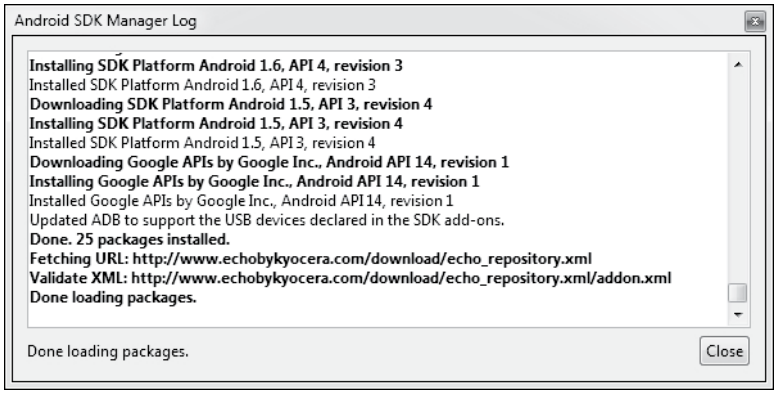


Figura 1.25 – Janela de log do SDK Manager.

Se o próprio programa SDK Manager tiver sido atualizado, você verá uma mensagem pedindo que reinicie o programa (veja a figura 1.26).

O SDK Manager também é utilizado para download de pacotes adicionais que não são parte da plataforma padrão. Esse mecanismo é utilizado por fabricantes de dispositivos para fornecer suporte a seus próprios hardwares. Por exemplo, a LG Electronics fornece um dispositivo 3D e, para oferecer suporte à capacidade 3D em aplicativos, um pacote adicional é fornecido. Ele também é utilizado pelo Google para permitir o download de APIs opcionais.

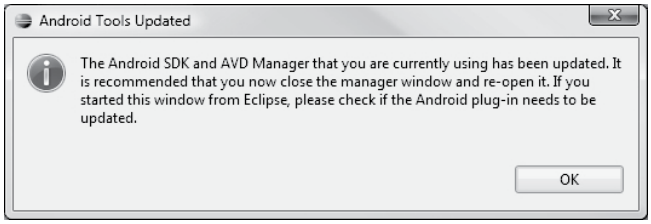


Figura 1.26 – Notificação de atualização do SDK Manager.

Na caixa de diálogo do SDK Manager, expanda e marque os pacotes necessários na lista da esquerda e então clique no botão **Install** (veja a figura 1.27). Se um pacote de terceiros não estiver listado, o URL para um arquivo *respository.xml*, fornecido pelo publicador do pacote, terá de ser digitado por meio do menu **Tools**.

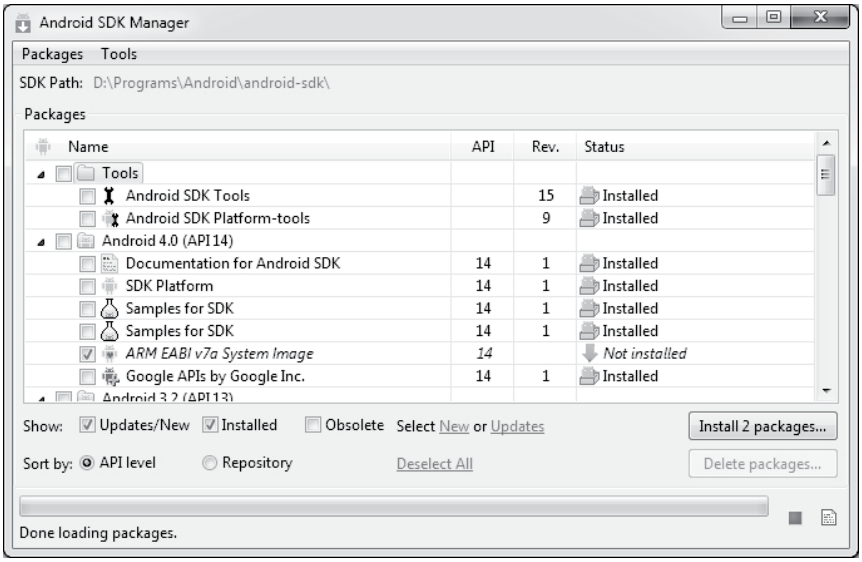


Figura 1.27 – Lista de componentes instalados e instaláveis.

Possíveis erros de atualização no Windows

Em um sistema tão complexo, há muitas coisas que podem dar errado. Esta seção discute algumas dessas coisas e suas soluções.

Execute o SDK Manager como administrador. Em uma máquina Windows, a localização padrão para o SDK é o diretório *C:\Arquivos de Programas\Android\android-sdk*. Esse é um diretório restrito e isso pode fazer com que a instalação do SDK fracasse. Um diálogo de mensagem com o título "SDK Manager: failed to install" pode aparecer (veja a figura 1.28).

Figura 1.28 – SDK Manager: falha ao instalar.

Para resolver esse erro há algumas coisas a verificar:

- Desconecte quaisquer dispositivos Android (isso pode impedir o *adb.exe* de fechar).
- Vá até *C:\Arquivos de Programas\Android\Android-sdk* e abra as propriedades para a pasta de ferramentas (selecione o menu de contexto e então **Propriedades**). Certifique-se de que a caixa seletora "Somente leitura (arquivos da pasta)" não está clicada (veja figura 1.29).

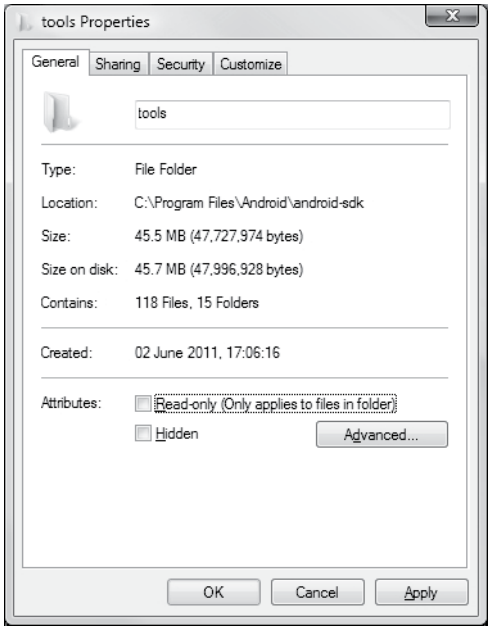


Figura 1.29 – Definição do atributo de leitura e escrita no Microsoft Windows.

Você pode ter de dar permissão para alterar os atributos (veja a figura 1.30).

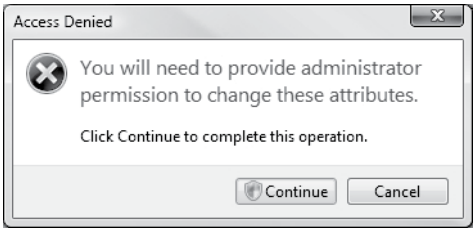


Figura 1.30 – Confirmação de permissão necessária.

Uma caixa de diálogo de confirmação das modificações de atributos vai aparecer; certifique-se de que a opção "Aplicar as alterações a esta pasta, subpasta e arquivos" está selecionada e clique em `OK`. Então, faça o seguinte:

- Reinicie o computador.
- Certifique-se de que todos os outros programas estão fechados, especialmente qualquer cópia do Gerenciador de Arquivos.
- Execute o *SDK Manager.exe* na conta de administrador. Abra o menu de contexto e selecione "Executar como administrador" (veja a figura 1.31).

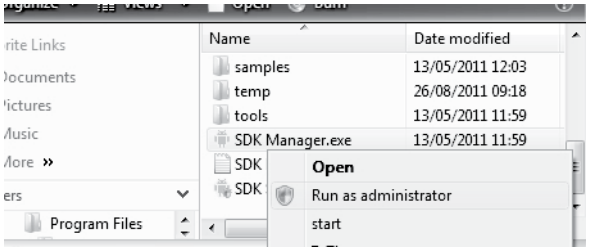


Figura 1.31 – Executar como administrador.

Feche o ADB antes de atualizar. Uma mensagem pedindo que você reinicie o ADB (o Android Debugger) pode aparecer (Figura 1.32).

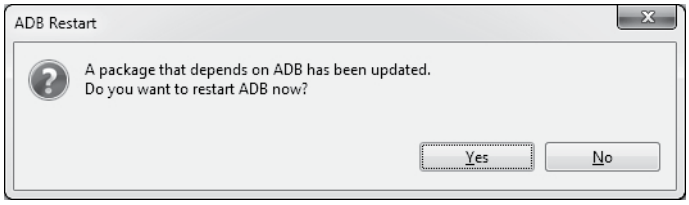


Figura 1.32 – Confirmação para reiniciar o ADB.

Idealmente, é melhor executar o SDK Manager sem que o ADB esteja sendo executado, e ele não deve estar sendo executado se o Windows tiver acabado de iniciar. Como alternativa você pode utilizar o Gerenciador de Tarefas do Windows para interromper o *adb.exe*. Se o ADB não estava sendo executado, responda `No` a esse prompt; do contrário, responda `Yes`.

O SDK Manager não pode se atualizar. Durante a instalação da atualização do SDK pode haver um erro relacionado ao programa SDK Manager (veja a figura 1.33).

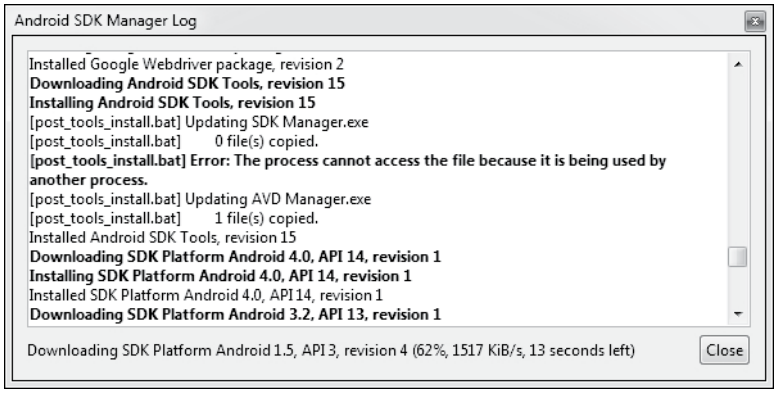


Figura 1.33 – Janela de log do SDK Manager do Android.

Para solucionar esse erro, certifique-se de que todos os programas estão fechados (inclusive o *adb.exe*). Depois copie o *SDK Manager.exe* de *C:\Arquivos de Programas\Android\android-sdk\tools\lib* para *C:\Arquivos de Programas\Android\android-sdk* (ou onde quer que o SDK esteja instalado). Então execute novamente o SDK Manager (veja a figura 1.32).

Atualização do Eclipse. Depois que você atualizar o SDK e abrir o Eclipse, uma mensagem de erro pode aparecer (veja a figura 1.34).

Figura 1.34 – Versão incorreta do SDK do Android.

No Eclipse, selecione **Help** e então **Check for Updates**. Espere até que o diálogo de progresso termine e as atualizações do Android Eclipse sejam mostradas. Clique em **Next** duas vezes, e aceite os termos de licença. Então, clique em **Finish** para começar o processo de download e atualização. Uma mensagem de aviso sobre conteúdo não assinado pode aparecer. Clique em **OK** para aceitar o aviso (faça isso apenas se você estiver atualizando via Eclipse). Reinicie o Eclipse assim que a atualização estiver concluída (uma mensagem para isso vai aparecer).

Mais informações sobre a resolução de problemas do SDK Manager e do plugin do Eclipse para Android estão disponíveis no site Android Developers.

Veja também

Receita 1.5; Instalação do SDK; Inclusão de componentes SDK; Plugin ADT para Eclipse

1.13 Captura de tela do emulador/dispositivo Android

Rachee Singh

Problema

Você deseja obter uma captura de tela de um aplicativo executado em um dispositivo Android.

Solução

Utilize a funcionalidade Device Screen Capture da visão DDMS (Dalvik Debug Monitor Server) no Eclipse.

Discussão

Para utilizar a funcionalidade Device Screen Capture siga estes passos:

1. Execute o aplicativo no Eclipse e vá até a visão DDMS (**Window menu->Open Perspective->Other->DDMS**) ou **Window menu->Show View->Other->Android->Devices**; a primeira é mostrada na figura 1.36). Note que a linha que diz "Resource...does not exist" aparece na figura 1.35 apenas porque outro projeto Eclipse foi fechado, e não afeta os passos listados aqui.

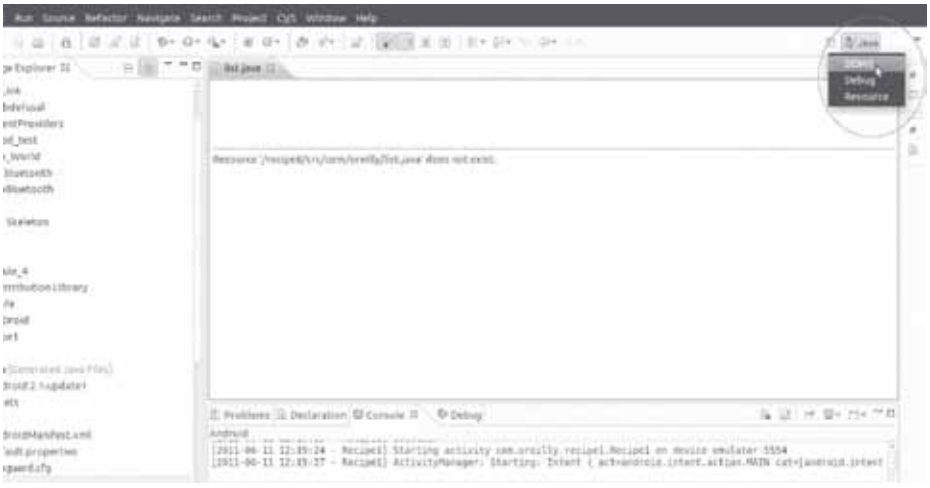


Figura 1.35 – Iniciando a visão DDMS.

2. Na visão DDMS, selecione o dispositivo ou emulador cuja tela você deseja capturar.
3. Na visão DDMS, clique no ícone Screen Capture (veja a figura 1-36).

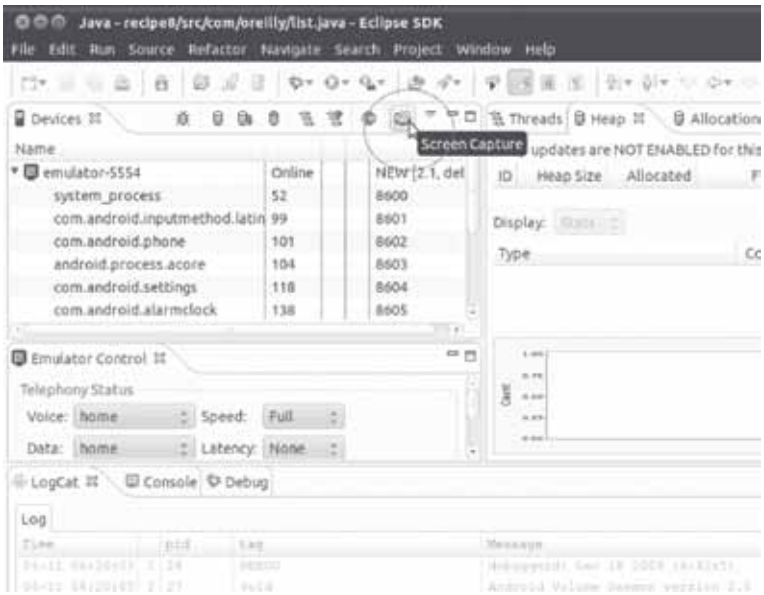


Figura 1.36 – Captura de tela do dispositivo.

4. Uma janela mostrando a tela atual do emulador/dispositivo Android será mostrada. Ela deve se parecer com a figura 1.37. Você pode até salvar a captura de tela e utilizá-la para descrever o aplicativo!

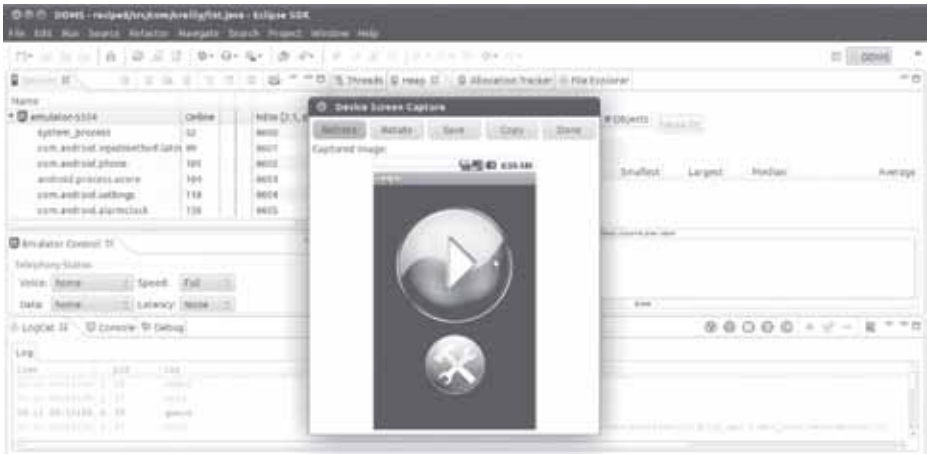


Figura 1.37 – A captura de tela.

Veja também

Algumas distribuições fornecem modos alternativos de se obter capturas de tela. O CyanogenMod 7.x fornece uma captura de tela no menu que surge quando você segura pressionado o botão de ligar/desligar. Alguns tablets HTC com suporte a caneta oferecem capturas de tela no menu Pen. O Ice Cream Sandwich (Android 4.0) fornece um mecanismo integrado para captura de tela em dispositivos reais: basta pressionar o controle de diminuição de volume ao mesmo tempo em que o botão de ligar/desligar e a imagem serão salvos em seu dispositivo e poderá ser visualizada no aplicativo de galeria.

1.14 Programa: um simples exemplo de CountDownTimer

Wagied Davids

Problema

Você deseja um simples timer regressivo, um programa que fará uma contagem regressiva a partir de um determinado número de segundos até atingir zero.

Solução

O Android vem com uma classe integrada para construção de `CountDownTimer`s. Ela é fácil de utilizar, eficiente e funciona (isso nem precisa dizer!).

Discussão

Os passos para fornecer um timer regressivo são estes:

1. Crie uma subclasse de `CountDownTimer`. O construtor dessa classe recebe dois argumentos, `CountDownTimer(long millisInFuture, long countDownInterval)`. O primeiro é o número de milissegundos, contados a partir de agora, determinando quando o intervalo deve estar concluído; nesse ponto o método `onFinish()` da subclasse será chamado. O segundo é a frequência em milissegundos da regularidade em que você deseja ser notificado de que o timer ainda está em execução, normalmente para atualizar um monitor de progresso ou comunicar o usuário de outra forma. O método `onTick()` de sua subclasse será chamado a cada passagem desse número de milissegundos.
2. Substitua os métodos `onTick()` e `onFinish()`.
3. Instancie uma nova instância em sua atividade Android.
4. Chame o método `start()` na nova instância criada!

O programa de exemplo de timer regressivo consiste em um layout XML (que pode ser visto no exemplo 1.4) e um pouco de código Java (exemplo 1.5). Quando executado, ele deve se parecer mais ou menos com a figura 1.38, ainda que os tempos sejam provavelmente diferentes.

Exemplo 1.4 – main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
        android:id="@+id/button"
        android:text="Start"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <TableLayout
        android:padding="10dip"
        android:layout_gravity="center"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TableRow>
            <TextView
                android:id="@+id/timer"
                android:text="Time: "
                android:paddingRight="10dip"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
            <TextView
                android:id="@+id/timeElapsed"
                android:text="Time elapsed: "
                android:paddingRight="10dip"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </TableRow>
    </TableLayout>
</LinearLayout>
```

Exemplo 1.5 – Main.java

```
package com.examples;

import android.app.Activity;
import android.os.Bundle;
```

```
import android.os.CountDownTimer;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class Main extends Activity implements OnClickListener
   {
       private MalibuCountDownTimer countDownTimer;
       private long timeElapsed;
       private boolean timerHasStarted = false;
       private Button startB;
       private TextView text;
       private TextView timeElapsedView;
       private final long startTime = 50 * 1000;
       private final long interval = 1 * 1000;

       /** Chamado quando a atividade é criada pela primeira vez */
       @Override
       public void onCreate(Bundle savedInstanceState)
           {
               super.onCreate(savedInstanceState);
               setContentView(R.layout.main);
               startB = (Button) this.findViewById(R.id.button);
               startB.setOnClickListener(this);

               text = (TextView) this.findViewById(R.id.timer);
               timeElapsedView = (TextView) this.findViewById(R.id.timeElapsed);
               countDownTimer = new MalibuCountDownTimer(startTime, interval);
               text.setText(text.getText() + String.valueOf(startTime));
           }
       @Override
       public void onClick(View v)
           {
               if (!timerHasStarted)
                   {
                       countDownTimer.start();
                       timerHasStarted = true;
                       startB.setText("Start");
                   }
               else
                   {
                       countDownTimer.cancel();
                       timerHasStarted = false;
                       startB.setText("RESET");
                   }
           }
```

```
        // Classe CountDownTimer
        public class MalibuCountDownTimer extends CountDownTimer
            {
                public MalibuCountDownTimer(long startTime, long interval)
                    {
                        super(startTime, interval);
                    }
                @Override
                public void onFinish()
                    {
                        text.setText("Time's up!");
                        timeElapsedView.setText("Time Elapsed: " +
                            String.valueOf(startTime));
                    }
                @Override
                public void onTick(long millisUntilFinished)
                    {
                        text.setText("Time remain:" + millisUntilFinished);
                        timeElapsed = startTime - millisUntilFinished;
                        timeElapsedView.setText("Time Elapsed: " +
                            String.valueOf(timeElapsed));
                    }
            }
    }
```

Figura 1.38 – Reset do timer.

URL de download do código-fonte

O código-fonte para esse exemplo está no repositório do Android Cookbook, em *http://github.com/AndroidCook/Android-Cookbook-Examples*, no subdiretório *CountDownTimerExample* (veja "Obtenção e uso de exemplos de código" na página 20).

1.15 Programa: Tipster, um calculador de gorjetas para o Android OS

Sunit Katkar

Problema

Quando você vai com amigos a um restaurante e deseja dividir a conta e a gorjeta, pode se envolver em muitos cálculos manuais e controvérsias. Em vez disso, você deseja utilizar um aplicativo que permita que você simplesmente some o percentual da gorjeta ao total e divida o valor pelo número de presentes. O Tipster é uma implementação disso no Android, para mostrar um aplicativo completo.

Solução

Este é um exercício simples que utiliza os elementos básicos da GUI (*graphical user interface*, ou interface gráfica de usuário), combinando-os a alguns cálculos simples e a um código de UI (*user interface*, ou interface de usuário) orientada a eventos para reunir todos os elementos. Nós utilizaremos os seguintes componentes de GUI:

`TableLayout`
: Fornece controle adequado sobre o layout da tela. Esse layout permite que você utilize o paradigma da tag HTML `Table` para posicionar os widgets.

`TableRow`
: Define uma linha no `TableLayout`. É como as tags HTML `TR` e `TD` combinadas.

`TextView`
: Essa `View` fornece um rótulo para apresentação de texto estático na tela.

`EditText`
: Essa `View` fornece um campo de texto para entrada de valores.

`RadioGroup`
: Agrupa botões de opção.

`RadioButton`
: Fornece um botão de opção.

`Button`

Esse é o botão normal.

`View`

Nós utilizaremos uma `View` para criar um separador visual com certos atributos de altura e cor.

Discussão

O Android utiliza arquivos XML para o layout de widgets. Em nosso projeto de exemplo, o plugin do Android para Eclipse gera um arquivo *main.xml* para o layout. Esse arquivo tem as definições XML dos vários widgets e de seus contêineres.

Há um arquivo *strings.xml* que tem todos os recursos de strings utilizados no aplicativo. Um arquivo *icon.png* padrão é fornecido para o ícone do aplicativo.

E há também o arquivo *R.java* que é gerado automaticamente (e atualizado quando quaisquer alterações são feitas ao *main.xml*). Esse arquivo tem as constantes definidas para cada layout e widget. Não edite esse arquivo manualmente; o plugin fará isso para você ao efetuar qualquer modificação em seus arquivos XML.

Em nosso exemplo temos *Tipster.java* como o arquivo Java principal para a `Activity`.

A receita 1.4, assim como diversos tutoriais do Google, mostra como utilizar o plugin. Utilizando o plugin do Eclipse, crie um projeto Android chamado Tipster. O resultado final será um layout de projeto que se parecerá com o da figura 1.39.

Criação do layout e posicionamento dos widgets

O objetivo final é criar um layout semelhante àquele mostrado na figura 1.39.

Para o layout dessa tela você vai utilizar estes layouts e widgets:

`TableLayout`

Fornece controle adequado sobre o layout da tela. Esse layout permite que você utilize o paradigma da tag HTML `Table` para posicionar os widgets.

`TableRow`

Define uma linha no `TableLayout`. É como as tags HTML `TR` e `TD` combinadas.

`TextView`

Essa `View` fornece um rótulo para apresentação de texto estático na tela.

`EditText`

Essa `View` fornece um campo de texto para entrada de valores.

`RadioGroup`

Agrupa botões de opção.

`RadioButton`

Fornece um botão de opção.

`Button`

Esse é o botão normal.

`View`

Nós utilizaremos uma `View` para criar um separador visual com certos atributos de altura e cor.

Familiarize-se com esses widgets já que você vai utilizá-los bastante nos aplicativos que construirá. Ao verificar os Javadocs relacionados a layouts e widgets, dê uma olhada nos atributos XML. Isso vai ajudá-lo a correlacionar o uso no arquivo de layout *main.xml* e o código Java (*Tipster.java* e *R.java*) onde eles são acessados.

Também está disponível um editor visual de layout no ADT do Eclipse, bem como uma ferramenta independente de UI chamada DroidDraw, ambas as quais permitem que você crie um layout arrastando e soltando widgets a partir de uma paleta, como qualquer ferramenta de design de formulários. No entanto, recomendo que você crie o layout à mão em XML, pelo menos nos estágios iniciais de seu aprendizado Android. Futuramente, à medida que aprender todas as nuances da API de layout XML, você poderá delegar essa tarefa a tais ferramentas.

O arquivo de layout, *main.xml*, tem as informações de layout (veja o exemplo 1.6). Um widget `TableRow` cria uma única linha dentro do `TableLayout`. Por isso você utiliza tantas `TableRows` quanto for o número de linhas que deseja. Neste tutorial, vamos utilizar oito `TableRows` – cinco para os widgets até o separador visual abaixo dos botões, e três para a área de resultados abaixo dos botões e do separador.

Exemplo 1.6 – /res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Uso de layout de tabela, para controle do tipo tabela HTML sobre o layout -->
<TableLayout
        android:id="@+id/TableLayout01"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="1"
        xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Linha 1: rótulo de texto colocado na coluna zero,
         campo de texto colocado na coluna dois e permitido estender-se
```

```
        por duas colunas. Assim, um total de 4 colunas nesta linha -->
        <TableRow>
        <TextView
                android:id="@+id/txtLbl1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:text="@string/textLbl1"/>
        <EditText ❶
                android:id="@+id/txtAmount"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:numeric="decimal"
                android:layout_column="2"
                android:layout_span="2"
                />
        </TableRow>
    <!-- Linha 2: rótulo de texto colocado na coluna zero,
         campo de texto colocado na coluna dois e permitido estender-se
         por duas colunas. Assim, um total de 4 colunas nesta linha -->
        <TableRow>
        <TextView
                android:id="@+id/txtLbl2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_column="0"
                android:text="@string/textLbl2"/>
        <EditText
                android:id="@+id/txtPeople"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:numeric="integer"
                android:layout_column="2"
                android:layout_span="3"/>
        </TableRow>
    <!-- Linha 3: esta tem apenas um rótulo de texto posicionado na coluna zero -->
        <TableRow>
        <TextView
                android:id="@+id/txtLbl3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/textLbl3"/>
        </TableRow>
    <!-- Linha 4: RadioGroup para RadioButtons posicionado na coluna zero
         com extensão de três colunas, criando, assim, um botão de opção
         por célula da linha da tabela. A quarta e última célula engloba o
```

```
        textfield para entrada de um percentual personalizado de gorjeta -->
    <TableRow>
    <RadioGroup
            android:id="@+id/RadioGroupTips"
            android:orientation="horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_span="3"
            android:checkedButton="@+id/radioFifteen">
            <RadioButton android:id="@+id/radioFifteen"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rdoTxt15"
                    android:textSize="15sp" />
            <RadioButton android:id="@+id/radioTwenty"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rdoTxt20"
                    android:textSize="15sp" />
            <RadioButton android:id="@+id/radioOther"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="@string/rdoTxtOther"
                    android:textSize="15sp" />
    </RadioGroup>
            <EditText
                    android:id="@+id/txtTipOther"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:numeric="decimal"/>
    </TableRow>
<!-- Linha para os botões Calculate e Reset. O botão Calculate
     é posicionado na coluna dois, e o Reset na coluna três -->
    <TableRow>
    <Button
            android:id="@+id/btnReset"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:text="@string/btnReset"/>
    <Button
            android:id="@+id/btnCalculate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="3"
```

```
            android:text="@string/btnCalculate"/>
        </TableRow>
    <!-- TableLayout permite que quaisquer outras visões sejam inseridas entre os
         elementos TableRow. Assim, insira uma visão em branco para criar um
         separador de linha. Essa visão separadora é utilizada para separar a
         área abaixo dos botões, a qual apresentará os resultados dos cálculos -->
        <View
            android:layout_height="2px"
            android:background="#DDFFDD"
            android:layout_marginTop="5dip"
            android:layout_marginBottom="5dip"/>
    <!-- Novamente a linha da tabela é utilizada para posicionar as TextViews de resultado
         na coluna zero e o resultado em TextViews na coluna dois -->
        <TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
        <TextView
            android:id="@+id/txtLbl4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:text="@string/textLbl4"/>
        <TextView
            android:id="@+id/txtTipAmount"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_span="2"/>
        </TableRow>
        <TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
        <TextView
            android:id="@+id/txtLbl5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:text="@string/textLbl5"/>
        <TextView
            android:id="@+id/txtTotalToPay"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_span="2"/>
        </TableRow>
        <TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
        <TextView
            android:id="@+id/txtLbl6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_column="0"
            android:text="@string/textLbl6"/>
        <TextView
            android:id="@+id/txtTipPerPerson"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_span="2"/>
        </TableRow>
    <!-- Fim de todas as linhas e widgets -->
</TableLayout>
```

TableLayout e TableRow

Depois de analisar *main.xml*, você pode perceber que `TableLayout` e `TableRow` são simples de usar. Você cria o `TableLayout` uma vez, e então insere uma `TableRow`. Agora você está livre para inserir quaisquer outros widgets, como `TextView`, `EditView` e assim por diante, dentro dessa `TableRow`.

Não deixe de dar uma olhada nos atributos, especialmente em `android:stretchColumns`, `android:layout_column` e `android:layout_span`, os quais permitem que você posicione widgets da mesma forma que você faria com uma tabela HTML comum. Recomendo que você siga os links para esses atributos e aprenda como eles funcionam em um `TableLayout`.

Controle de valores de entrada

Dê uma olhada no widget `EditText`, marcado no arquivo *main.xml* como ❶. Esse é o primeiro campo de texto para entrada do "Valor total" da conta. Queremos apenas números aqui. Podemos aceitar números decimais porque contas de restaurantes reais também podem ter valores em centavos, e não apenas números inteiros. Por isso utilizamos o atributo `android:numeric` com um valor de `decimal`. Isso permitirá valores de números inteiros, como 10, e valores decimais, como 10.12, mas impedirá qualquer outro tipo de entrada.

Essa é uma forma simples e concisa de controlar valores de entrada, economizando-nos o trabalho de escrever código de validação no arquivo *Tipster.java*, e garantindo que o usuário não digite valores errados. Essa funcionalidade do Android de limitação com base em XML é bastante poderosa e útil. Você deve explorar todos os atributos possíveis que acompanham um widget específico para extrair o máximo de benefícios desse modo XML abreviado de definir restrições. Em um lançamento futuro, a não ser que eu não tenha visto isso nesta versão, eu espero que o Android permita a entrada de intervalos para o atributo `android:numeric`, de modo que você possa definir qual intervalo de números deseja aceitar.

Como (até onde eu sei) intervalos atualmente não estão disponíveis, você verá futuramente que temos realmente de verificar a presença de certos valores, como zero ou valores vazios, para garantir que as contas de nosso calculador de gorjeta não falhem.

Análise do Tipster.java

Agora vamos analisar o arquivo *Tipster.java* que controla nosso aplicativo. Essa é a classe principal que cuida do layout, do tratamento de eventos e da lógica do aplicativo.

O plugin do Eclipse para Android cria o arquivo *Tipster.java* em nosso projeto com o código padrão que pode ser visto no exemplo 1.7.

Exemplo 1.7 – Trecho de código 1 de /src/com/examples/tipcalc/Tipster.java

```
package com.examples.tipcalc;

import android.app.Activity;

public class Tipster extends Activity {
    /** Chamado quando a atividade é criada pela primeira vez */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

A classe `Tipster` estende a classe `android.app.Activity`. Uma atividade é uma única ação concentrada que o usuário pode realizar. A classe `Activity` cuida de criar a janela e depois de apresentar a UI. Você tem de chamar o método `setContentView(View view)` para colocar sua UI na `Activity`. Por isso, pense em `Activity` como uma moldura externa que está vazia, e que você preenche com sua UI.

Agora verifique o trecho da classe *Tipster.java* que pode ser visto no exemplo 1.8. Primeiro, definimos os widgets como membros de classe. Verifique, mais especificamente, de ❶ a ❷ para referência.

Então, utilizamos o método `findViewById(int id)` para localizar os widgets. O ID de cada widget, definido em seu arquivo *main.xml*, é automaticamente definido no arquivo *R.java* quando você limpa e compila o projeto no Eclipse. (Se você configurou o Eclipse para compilar automaticamente, o arquivo *R.java* será atualizado instantaneamente quando você atualizar *main.xml*.)

Cada widget é derivado da classe `View`, e fornece funcionalidades GUI especiais. Assim, uma `TextView` fornece uma forma de colocar rótulos na UI, enquanto que o `EditText` fornece um campo de texto. Verifique de ❸ a ❻ no exemplo 1.8. Você pode ver como `findViewById()` é utilizado para localizar os widgets.

Exemplo 1.8 – Trecho de código 2 de /src/com/examples/tipcalc/Tipster.java

```
public class Tipster extends Activity {
    // Widgets do aplicativo
    private EditText txtAmount; ❶
    private EditText txtPeople;
    private EditText txtTipOther;
    private RadioGroup rdoGroupTips;
    private Button btnCalculate;
    private Button btnReset;

    private TextView txtTipAmount;
    private TextView txtTotalToPay;
    private TextView txtTipPerPerson; ❷

    // Para o id do botão de opção selecionado
    private int radioCheckedId = -1;

    /** Chamado quando a atividade é criada pela primeira vez */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Acessa os vários widgets por id em R.java
        txtAmount = (EditText) findViewById(R.id.txtAmount); ❸
        // No carregamento do aplicativo, o cursor deve estar no campo Amount
        txtAmount.requestFocus(); ❹

        txtPeople = (EditText) findViewById(R.id.txtPeople);
        txtTipOther = (EditText) findViewById(R.id.txtTipOther);

        rdoGroupTips = (RadioGroup) findViewById(R.id.RadioGroupTips);

        btnCalculate = (Button) findViewById(R.id.btnCalculate);
        // No carregamento do aplicativo, o botão Calculate está desabilitado
        btnCalculate.setEnabled(false); ❺

        btnReset = (Button) findViewById(R.id.btnReset);

        txtTipAmount = (TextView) findViewById(R.id.txtTipAmount);
        txtTotalToPay = (TextView) findViewById(R.id.txtTotalToPay);
        txtTipPerPerson = (TextView) findViewById(R.id.txtTipPerPerson); ❻

        // No carregamento do aplicativo, desabilite o campo de texto Other Tip Percentage
        txtTipOther.setEnabled(false); ❼
```

Lidando com considerações de facilidade de uso ou usabilidade

Nosso aplicativo deve tentar ser tão utilizável quanto qualquer outro aplicativo ou página web consolidado. Em resumo, incluir recursos de usabilidade resultará em uma boa experiência de usuário. Para lidar com essas considerações, dê uma olhada no exemplo 1.8 novamente.

Veja ❹, onde utilizamos o método `requestFocus()` da classe `View`. Como o widget `EditText` é derivado da classe `View`, esse método é aplicável a ele. Isso é feito para que, quando nosso aplicativo carregar, o campo de texto Total Amount receba foco e o cursor seja posicionado nele. Isso é semelhante à tela de login de aplicativos web populares onde o cursor está presente no campo de texto do nome do usuário.

Agora verifique ❺, onde o botão Calculate é desabilitado chamando o método `setEnabled(boolean enabled)` no widget `Button`. Isso é feito para que o usuário não possa clicar nele antes de digitar valores nos campos obrigatórios. Se permitíssemos que o usuário clicasse em Calculate sem digitar valores nos campos Total Amount e No. of People, teríamos de escrever um código de validação para capturar essas condições. Isso significaria mostrar um alerta em popup avisando o usuário dos valores vazios. Algo desse tipo adicionaria código e interação de usuário desnecessários. Quando o usuário vê o botão Calculate desabilitado, fica bastante óbvio que, a não ser que todos os valores sejam digitados, a gorjeta não poderá ser calculada.

Verifique ❼ no exemplo 1.8. Aqui o campo de texto Other Tip Percentage está desabilitado. Isso é feito porque o botão de opção "15% tip" é selecionado por padrão quando o aplicativo carrega. Essa seleção padrão no carregamento do aplicativo é feita por meio do arquivo *main.xml*. Dê uma olhada na linha de *main.xml* onde a instrução a seguir seleciona o botão de opção "15% tip":

```
android:checkedButton="@+id/radioFifteen"
```

O atributo `android:checkedButton` de `RadioGroup` permite que você selecione um dos widgets `RadioButton` no grupo por padrão.

A maioria dos usuários que já utilizou aplicativos populares no desktop ou na web está familiarizada com o paradigma "widgets desabilitados que são habilitados em certas condições". Incluir essas pequenas conveniências sempre torna um aplicativo mais utilizável e a experiência do usuário mais rica.

Processamento de eventos da UI

Semelhante aos populares frameworks de UI do Windows, Java Swing, Flex e outros, o Android também fornece um modelo de eventos que permite a você escutar determinados eventos na UI provocados pela interação do usuário. Vejamos como podemos utilizar o modelo de eventos do Android em nosso aplicativo.

Primeiro, vamos nos concentrar nos botões de opção da UI. Queremos saber qual botão de opção o usuário selecionou, uma vez que isso nos permitirá determinar o percentual de gorjeta em nossos cálculos. Para "escutar" botões de opção, utilizamos a interface estática `OnCheckedChangeListener()`. Isso vai nos notificar quando o estado de seleção de um botão de opção mudar.

Em nosso aplicativo, queremos habilitar o campo de texto Other Tip Percentage apenas quando o botão de opção Other estiver selecionado. Quando os botões "15% tip" e "20% tip" estiverem selecionados, nós queremos desabilitar esse campo de texto. Além disso, queremos incluir um pouco mais de lógica em nome da usabilidade. Como discutimos antes, não devemos habilitar o botão Calculate até que todos os campos obrigatórios tenham valores válidos. Em termos dos três botões de opção, queremos garantir que o botão Calculate esteja habilitado para as duas condições a seguir:

- O botão de opção Other está selecionado e o campo de texto Other Tip Percentage tem valores válidos.
- Os botões de opção "15% tip" ou "20% tip" estão selecionados e os campos de texto Total Amount e No. of People têm valores válidos.

Veja o exemplo 1.9, o qual lida com os botões de opção. Os comentários do código-fonte são bastante autoexplicativos.

Exemplo 1.9 – Trecho de código 3 de /src/com/examples/tipcalc/Tipster.java

```
/*
* Anexa um OnCheckedChangeListener ao grupo de opção
* para monitorar botões de opção selecionados pelo usuário
*/
rdoGroupTips.setOnCheckedChangeListener(new OnCheckedChangeListener() {

@Override
public void onCheckedChanged(RadioGroup group, int checkedId) {
    // Habilita/desabilita o campo Other Tip Percentage
    if (checkedId == R.id.radioFifteen || checkedId == R.id.radioTwenty) {
        txtTipOther.setEnabled(false);
        /*
        * Habilita o botão Calculate se os campos Total
        * Amount e No. of People tiverem valores válidos
        */
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
                && txtPeople.getText().length() > 0);
    }
    if (checkedId == R.id.radioOther) {
        // Habilita o campo Other Tip Percentage
        txtTipOther.setEnabled(true);
        // define o foco para este campo
        txtTipOther.requestFocus();
        /*
        * Habilita o botão Calculate se os campos Total Amount e No. of People tiverem
        * valores válidos. Também assegura que o usuário digitou um valor em Other Tip
        * Percentage antes de habilitar o botão Calculate
        */
```

```
            btnCalculate.setEnabled(txtAmount.getText().length() > 0
                    && txtPeople.getText().length() > 0
                    && txtTipOther.getText().length() > 0);
        }
        // Para determinar a escolha do percentual de gorjeta feita pelo usuário
        radioCheckedId = checkedId;
        }
    });
```

Monitorando atividade de teclas em campos de texto

Como mencionei antes, o botão Calculate não deve ser habilitado a não ser que os campos de texto tenham valores válidos. Então temos de garantir que o botão Calculate seja habilitado apenas se os campos de texto Total Amount, No. of People e Other Tip Percentage tiverem valores válidos. O campo de texto Other Tip Percentage será habilitado apenas se o botão de opção Other Tip Percentage for selecionado.

Não temos de nos preocupar com o tipo dos valores, ou seja, se o usuário digitou ou não números negativos ou letras, pois o atributo `android:numeric` foi definido para os campos de texto, limitando, dessa forma, os tipos de valores que o usuário pode digitar. Temos apenas de garantir que os valores estejam presentes.

Então utilizamos a interface estática `OnKeyListener()`. Ela vai nos notificar quando uma tecla for pressionada. A notificação chegará até nós antes de a tecla pressionada em si ser enviada para o widget `EditText`.

Verifique o código nos exemplos 1.10 e 1.11, que lidam com eventos de teclas em campos de texto. Assim como no exemplo 1.9, os comentários do código-fonte são bastante autoexplicativos.

Exemplo 1.10 – Trecho de código 4 de /src/com/examples/tipcalc/Tipster.java

```
/*
* Anexa um KeyListener aos campos de texto Tip Amount, No. of People
* e Other Tip Percentage
*/
txtAmount.setOnKeyListener(mKeyListener);
txtPeople.setOnKeyListener(mKeyListener);
txtTipOther.setOnKeyListener(mKeyListener);
```

Note que criamos apenas um ouvinte, em vez de criar ouvintes anônimos/internos para cada campo de texto. Não tenho certeza se meu estilo é melhor ou recomendado, mas sempre escrevo dessa forma se os ouvintes vão realizar ações habituais. Aqui, a preocupação comum a todos os campos de texto é que eles não devem estar vazios, e apenas quando tiverem valores é que o botão Calculate deve ser habilitado.

Exemplo 1.11 – Trecho de código 5 de KeyListener.java

```
/*
* KeyListener para os campos de texto Total Amount, No of People e Other Tip
* Percentage. Precisamos aplicar esse ouvinte de teclas para verificar estas condições:
*
* 1) Se o usuário selecionar Other Tip Percentage, o campo de texto Other Tip Percentage
* deve ter um percentual de gorjeta válido inserido pelo usuário. Habilite o
* botão Calculate apenas quando o usuário digitar um valor válido.
*
* 2) Se o usuário não digitar valores nos campos Total Amount e No. of People,
* não poderemos realizar os cálculos. Assim, habilitaremos o botão Calculate
* apenas quando o usuário digitar valores válidos.
*/
private OnKeyListener mKeyListener = new OnKeyListener() {
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event) {

    switch (v.getId()) { ❶
    case R.id.txtAmount: ❷
    case R.id.txtPeople: ❸
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
                && txtPeople.getText().length() > 0);
        break;
    case R.id.txtTipOther: ❹
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
                && txtPeople.getText().length() > 0
                && txtTipOther.getText().length() > 0);
        break;
    }
    return false;
    }
};
```

Em ❶ no exemplo 1.11, analisamos o ID da `View`. Lembre-se de que cada widget tem um ID único definido no arquivo *main.xml*. Esses valores são, então, definidos na classe *R.java* gerada.

Em ❷ e ❸, se o evento-chave ocorreu nos campos Total Amount ou No. of People, verificamos o valor digitado no campo. Estamos garantindo que o usuário não tenha deixado ambos os campos em branco.

Em ❹, verificamos se o usuário selecionou o botão de opção Other, e então garantimos que o campo de texto Other não esteja vazio. Também verificamos mais uma vez se os campos Total Amount e No. of People estão vazios.

Assim, o propósito de nosso `KeyListener` agora está claro: garantir que todos os campos de texto não estejam vazios, e apenas então habilitar o botão Calculate.

Escutando cliques em botões

Agora vamos verificar os botões **Calculate** e **Reset**. Quando o usuário clicar nesses botões, utilizaremos a interface estática `OnClickListener()` que nos informará quando um botão for clicado.

Como fizemos com os campos de texto, criamos apenas um ouvinte e, dentro dele, detectamos qual botão foi clicado. Dependendo do botão que foi clicado, o método `calculate()` ou `reset()` será chamado.

O exemplo 1.12 mostra como o ouvinte de cliques é adicionado aos botões.

Exemplo 1.12 – Trecho de código 6 de /src/com/examples/tipcalc/Tipster.java

```
/* Anexa um ouvinte aos botões Calculate e Reset */
btnCalculate.setOnClickListener(mClickListener);
btnReset.setOnClickListener(mClickListener);
```

O exemplo 1.13 mostra como detectar qual botão foi clicado verificando o ID da `View` que recebe o evento de clique.

Exemplo 1.13 – Trecho de código 7 de /src/com/examples/tipcalc/Tipster.java

```
/**
* ClickListener para os botões Calculate e Reset.
* Dependendo do botão clicado, o método correspondente será chamado.
*/
private OnClickListener mClickListener = new OnClickListener() {

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btnCalculate) {
            calculate();
        } else {
            reset();
        }
    }
};
```

Reset do aplicativo

Quando o usuário clicar no botão **Reset**, os campos de texto deverão ser limpos, o botão de opção-padrão "15% tip" deverá ser selecionado, e quaisquer resultados calculados deverão ser limpos.

O exemplo 1.14 mostra o método `reset()`.

Exemplo 1.14 – Trecho de código 8 de /src/com/examples/tipcalc/Tipster.java

```
/**
* Faz o reset das visões de texto dos resultados na parte inferior da tela,
* assim como dos campos de texto e dos botões de opção.
*/
private void reset() {
    txtTipAmount.setText("");
    txtTotalToPay.setText("");
    txtTipPerPerson.setText("");
    txtAmount.setText("");
    txtPeople.setText("");
    txtTipOther.setText("");
    rdoGroupTips.clearCheck();
    rdoGroupTips.check(R.id.radioFifteen);
    // define o foco no primeiro campo
    txtAmount.requestFocus();
}
```

Validação da entrada para calcular a gorjeta

Como eu disse antes, estamos limitando o tipo de valor que o usuário pode digitar nos campos de texto. No entanto, o usuário ainda pode digitar um valor de zero nos campos de texto Total Amount, No. of People e Other Tip Percentage, provocando, dessa forma, condições de erro, como divisão por zero, em nossos cálculos de gorjetas.

Se o usuário digitar zero, devemos mostrar um popup de alerta pedindo que digite um valor diferente de zero. Lidaremos com isso utilizando um método chamado `showErrorAlert(String errorMessage, final int fieldId)`, mas discutiremos isso mais detalhadamente no futuro.

Primeiro, verifique o exemplo 1.15, o qual mostra o método `calculate()`. Note como os valores digitados pelo usuário são processados como valores do tipo `Double`.

Agora perceba ❶ e ❷, onde verificamos a presença de valores iguais a zero. Se o usuário digitar zero, mostraremos um popup de alerta para avisá-lo. Depois, verifique ❸, onde o campo de texto Other Tip Percentage é habilitado porque o usuário selecionou o botão de opção Other. Aqui também devemos verificar se o percentual de gorjeta não é zero.

Quando o aplicativo carregar, o botão de opção "15% tip" será selecionado por padrão. Se o usuário modificar a seleção, atribuiremos o ID do botão de opção selecionado à variável membro `radioCheckedId`, como vimos no exemplo 1.9, em `OnCheckedChangeListener`.

Mas se o usuário aceitar a seleção padrão, o `radioCheckedId` terá o valor padrão de `-1`. Em resumo, nunca saberemos qual botão de opção foi selecionado. Nós sabemos, é claro, qual está selecionado por padrão, e poderíamos ter codificado a lógica de modo

levemente diferente, para presumir 15% se `radioCheckedId` tiver o valor de `-1`. Mas se você consultar a API, verá que podemos chamar o método `getCheckedRadioButtonId()` no `RadioGroup` e não em botões individuais de opção. Isso ocorre porque `OnCheckedChangeListener` nos fornece prontamente o ID do botão de opção selecionado.

Apresentação dos resultados

Calcular a gorjeta é simples. Se não houver nenhum erro de validação, a flag booleana `isError` será `false`. Verifique de ❹ a ❺ no exemplo 1.15 para observar os cálculos simples da gorjeta. Na sequência, os valores calculados serão definidos nos widgets `TextView` de ❻ a ❼.

Exemplo 1.15 – Trecho de código 9 de /src/com/examples/tipcalc/Tipster.java

```
/**
* Calcule a gorjeta de acordo com os dados digitados pelo usuário
*/
private void calculate() {
    Double billAmount = Double.parseDouble(txtAmount.getText().toString());
    Double totalPeople = Double.parseDouble(txtPeople.getText().toString());
    Double percentage = null;
    boolean isError = false;
    if (billAmount < 1.0) { ❶
        showErrorAlert("Enter a valid Total Amount.", txtAmount.getId());
        isError = true;
    }
    if (totalPeople < 1.0) { ❷
        showErrorAlert("Enter a valid value for No. of People.", txtPeople.getId());
        isError = true;
    }

    /*
    * Se o usuário nunca modificar sua seleção de opção,
    * significa que a seleção padrão de 15% está em efeito. Mas é mais seguro verificar
    */
    if (radioCheckedId == -1) {
        radioCheckedId = rdoGroupTips.getCheckedRadioButtonId();
    }
    if (radioCheckedId == R.id.radioFifteen) {
        percentage = 15.00;
    } else if (radioCheckedId == R.id.radioTwenty) {
        percentage = 20.00;
    } else if (radioCheckedId == R.id.radioOther) {
        percentage = Double.parseDouble(
            txtTipOther.getText().toString());
        if (percentage < 1.0) { ❸
```

```
            showErrorAlert("Enter a valid Tip percentage", txtTipOther.getId());
            isError = true;
        }
    }
    /*
    * Se todos os campos estiverem preenchidos com valores válidos, prossiga
    * para o cálculo das gorjetas
    */
    if (!isError) {
        Double tipAmount = ((billAmount * percentage) / 100); ❹
        Double totalToPay = billAmount + tipAmount;
        Double perPersonPays = totalToPay / totalPeople; ❺
        txtTipAmount.setText(tipAmount.toString()); ❻
        txtTotalToPay.setText(totalToPay.toString());
        txtTipPerPerson.setText(perPersonPays.toString()); ❼
    }
}
```

Apresentação dos alertas

O Android fornece a classe `AlertDialog` para mostrar popups de alerta. Isso nos permite mostrar uma caixa de diálogo com até três botões e uma mensagem.

O exemplo 1.16 mostra o método `showErrorAlert`, o qual utiliza esse `AlertDialog` para mostrar as mensagens de erro. Perceba que informamos dois argumentos a esse método: `String error Message` e `int fieldId`. O primeiro argumento é a mensagem de erro que queremos mostrar ao usuário. O `fieldId` é o ID do campo que causou a condição de erro. Depois que o usuário dispensar o diálogo de alerta, esse `fieldId` nos permitirá solicitar o foco para esse campo, de modo que o usuário saiba qual campo contém o erro.

Exemplo 1.16 – Trecho de código 10 de /src/com/examples/tipcalc/Tipster.java

```
/**
* Mostra a mensagem de erro no diálogo de alerta
*
* @param errorMessage
*       String a ser mostrada pela mensagem de erro
* @param fieldId
*       o Id do campo que provocou o erro. Isso é necessário para que o foco possa ser definido
*       nesse campo assim que o diálogo for dispensado.
*/
private void showErrorAlert(String errorMessage,
    final int fieldId) {
    new AlertDialog.Builder(this).setTitle("Error")
    .setMessage(errorMessage).setNeutralButton("Close",
            new DialogInterface.OnClickListener() {
```

```
                @Override
                public void onClick(DialogInterface dialog,
                    int which) {
                        findViewById(fieldId).requestFocus();
                    }
            }).show();
    }
```

Quando reunimos tudo isso, o resultado deve ficar como a figura 1.39.

Figura 1.39 – Tipster em ação.

Conclusão

O desenvolvimento para o OS Android não é tão diferente do desenvolvimento para qualquer kit de ferramentas de UI, incluindo Microsoft Windows, X Windows, Java Swing ou Adobe Flex. O Android tem, é claro, suas diferenças e, em geral, um design muito bom. O paradigma de layout em XML é muito interessante e útil para criação de UIs complexas utilizando XML simples. Além disso, o modelo de tratamento de eventos é simples, rico em funcionalidades e intuitivo de se utilizar no código.

URL de download do código-fonte

Você pode efetuar o download do código-fonte para esse exemplo em *http://www.vidyut.com/sunit/android/tipster.zip*.

URL de download do binário

Você pode efetuar o download do código executável para esse exemplo em *http://www.vidyut.com/sunit/android/tipster.zip*.