Tornando tudo mais fácil!

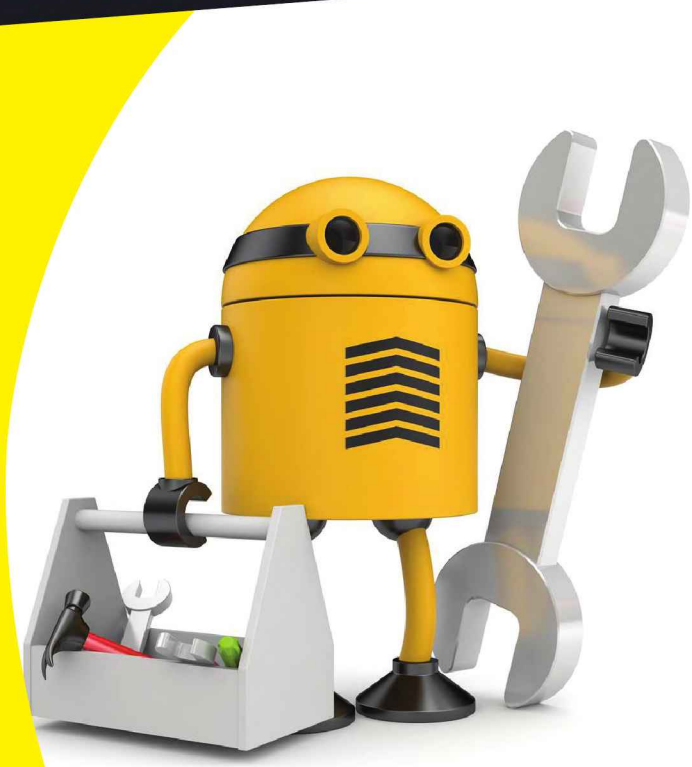
Tradução da 2ª edição

Desenvolvimento de Aplicativos Android™ para Leigos®

For Dummies®

Aprenda a:

- Criar aplicativos surpreendentes para os últimos smartphones e tablets Android
- Baixar e instalar o SDK e começar a trabalhar com as ferramentas JDK
- Adaptar seus aplicativos para uso em tablets Android
- Publicar seus aplicativos na Google Play Store



Michael Burton
Donn Felker

Desenvolvimento de Aplicativos Android Para Leigos

Folha de Cola

Sugestões Úteis para o Desenvolvimento de Aplicativos Android

- Não tem certeza sobre como resolver um determinado problema? Visite os fóruns em StackOverflow.com e use a tag Android.
- As tarefas mais comuns já foram feitas para você. Uma lista de tarefas e instruções comuns é fornecida pelo site do desenvolvedor Android.
- Se você não estiver certo sobre o que um pacote ou uma classe faz, coloque seu cursor sobre o objeto ou classe quando estiver no Eclipse para exibir a janela instantânea da documentação. Se não existir nenhuma dica da ferramenta, você poderá exibir a documentação on-line na Referência do Desenvolvedor Android.
- Entenda os ciclos de vida das atividades e dos fragmentos. Você irá usá-los durante sua carreira de desenvolvimento e uma compreensão completa poderá ajudar a evitar problemas.
- Lembre-se que você é responsável por salvar e restaurar os estados de instância nas atividades e nos fragmentos. Se você criar uma variável membro em uma atividade ou fragmento, salve-a em `onSaveInstanceState( )` e restaure-a usando o bundle em `onCreate( )`. Falhar em fazer isso poderá fazer sua atividade ou fragmento parecer funcionar na maioria das vezes, porém, falhando aleatoriamente em outras situações (como, por exemplo, quando um usuário gira a tela).
- Para tornar mais legíveis as mensagens em fluxo na perspectiva DDMS, crie um filtro que se aplica especificamente ao registro de seu aplicativo.
- Ao digitar no Eclipse, algumas vezes, você sabe o nome da propriedade de destino, método ou classe que gostaria de criar. Contudo, ela não existe ainda. Digite o nome no componente e o Eclipse irá informar se o componente não pode ser encontrado. Agora, selecione-o, e então pressione F2. Isso abrirá uma pequena janela instantânea que permitirá a criação através do clique de um botão.
- Para navegar rapidamente por um arquivo de classe, pressione Ctrl+O e comece a digitar o nome do membro no qual está interessado. Selecione-o na lista instantânea e pressione Enter.

Atalhos do Teclado Úteis no Eclipse para Desenvolver Aplicativos Android

Ação	Atalho do teclado
Criar um novo arquivo no pacote atual	Alt+Shift+N
Organizar as instruções de importação	Ctrl+Shift+O
Navegar para a definição da origem	F3
Renomear um objeto	Alt+Shift+R
Pesquisar os arquivos Java	Ctrl+H
Saltar para um determinado tipo	Ctrl+Shift+T
Encontrar declarações	Ctrl+G
Encontrar referências para um objeto selecionado	Ctrl+Shift+G
Navegar à esquerda	Alt+seta para esquerda
Navegar à direita	Alt+seta para direita
Navegar entre as guias	Shift+Page Up ou Shift+Page Down
Executar seu aplicativo	Ctrl+Shift+F11

Desenvolvimento de Aplicativos Android Para Leigos

Folha de Cola

Uso Comum das Intenções do Android

A seguinte tabela mostra os códigos das intenções Android básicas.

Intenção	*Código*
Iniciar uma atividade	`startActivity(new Intent(this, Destination.class));`
Criar um seletor	`Intent.createChooser(youIntent, "Please Select");`
Abrir o navegador web	`Intent i = newIntent(Intent.ACTION_VIEW, Uri.parse("http://example.org"));startActivity(i);`
Iniciar atividade para um resultado	`startActivityForResult(yourIntent,YOUR_REQUEST_CODE);`

Desenvolvimento de Aplicativos Android para Diversos Tamanhos de Tela

Um dos desafios que você encontrará ao ser um desenvolvedor de aplicativos Android é desenvolver para diversos tamanhos de tela.

- As exigências de tamanho para cada ícone em cada densidade variam para cada tipo de ícone. Você encontrará ícones de inicialização, de menu, da barra de status, de guias e muito mais. São todos construídos de modo diferente para cada densidade da tela. Ao construir esses ícones, consulte as Diretrizes de Construção de Ícones do Android.
- Tente sempre usar a unidade de medida pixel independente da densidade (dip ou dp) para as exibições e use pixel independente da escala (sip ou sp) para os tamanhos da fonte ao definir sua interface de usuário. Isso ajuda seu aplicativo a se dimensionar em diferentes dispositivos. O pixel independente da densidade é um pixel virtual que se dimensiona proporcionalmente para cada densidade de tela.
- Forneça o elemento de telas e suportes para o arquivo `AndroidManifest.xml` para ajudar a loja Android a determinar se seu aplicativo é compatível com os vários tamanhos de tela.
- Forneça gráficos com densidades extra-alta, alta, média e baixa para os dispositivos. Embora isso possa aumentar o tempo de desenvolvimento e design, melhora muito a utilização e a aparência de seu aplicativo.
- Use fragmentos de forma abundante para facilitar o suporte aos dispositivos tablet.

Para Leigos: A série de livros para iniciantes que mais vende no mundo.

Desenvolvimento de Aplicativos Android™

PARA

LEIGOS®

Tradução da 2ª Edição

A compra deste conteúdo não prevê o atendimento e fornecimento de suporte técnico operacional, instalação ou configuração do sistema de leitor de ebooks. Em alguns casos, e dependendo da plataforma, o suporte poderá ser obtido com o fabricante do equipamento e/ou loja de comércio de ebooks.

Desenvolvimento de Aplicativos Android™ PARA LEIGOS®

Tradução da 2ª Edição

de Michael Burton e Donn Felker

ALTA BOOKS
EDITORA
Rio de Janeiro, 2014

Desenvolvimento de Aplicativos Android Copyright © 2014 da Starlin Alta Editora e Consultoria Eireli.
ISBN: 978-85-7608-848-6

Translated from original Android™ Application Development For Dummies©, 2nd edition, 2012 by John Wiley & Sons, Inc., ISBN 978-1-118-38710-8. *This translation is published and sold by permission John Wiley & Sons, Inc., the owner of all rights to publish and sell the same. PORTUGUESE language edition published by Starlin Alta Editora e Consultoria Eireli, Copyright © 2014 by Starlin Alta Editora e Consultoria Eireli.*

Todos os direitos reservados e protegidos por Lei. Nenhuma parte deste livro, sem autorização prévia por escrito da editora, poderá ser reproduzida ou transmitida.

Erratas: No site da editora relatamos, com a devida correção, qualquer erro encontrado em nossos livros. Procure pelo título do livro.

Marcas Registradas: Todos os termos mencionados e reconhecidos como Marca Registrada e/ou Comercial são de responsabilidade de seus proprietários. A Editora informa não estar associada a nenhum produto e/ou fornecedor apresentado no livro.

Impresso no Brasil — 1ª Edição, 2014

Vedada, nos termos da lei, a reprodução total ou parcial deste livro.

Produção Editorial
Editora Alta Books

Gerência Editorial
Anderson Vieira

Editoria Para Leigos
Marcelo Vieira

Supervisão Gráfica
Angel Cabeza

Supervisão de Qualidade Editorial
Sergio Luiz de Souza

Supervisão de Texto
Jaciara Lima

Design Editorial
Auleriano Messias
Aurélio Corrêa

Captação e Contratação de Obras
Cristiane Santos
Marco Pace
J. A. Rugeri
autoria@altabooks.com.br

Vendas Atacado e Varejo
Daniele Fonseca
Viviane Paiva
comercial@altabooks.com.br

Marketing e Promoção
marketing@altabooks.com.br

Ouvidoria
ouvidoria@altabooks.com.br

Equipe Editorial
Claudia Braga
Hannah Carriello
Letícia Vitoria
Livia Brazil
Marcelo Vieira
Mayara Coelho
Milena Lepsch
Natália Gonçalves
Raquel Ferreira
Rodrigo Araujo
Thiê Alves

Tradução
Eveline Vieira Machado

Copidesque
Marcus Victor Martins

Revisão Gramatical
Thamiris Leiroza

Diagramação
Joyce Matos

Dados Internacionais de Catalogação na Publicação (CIP)

B974d Burton, Michael.
Desenvolvimento de aplicativos Android para leigos / por Michael Burton e Donn Felker. – Rio de Janeiro, RJ : Alta Books, 2014.
408 p. : il. ; 24 cm. – (Para leigos)

Inclui índice.
Tradução de: Android application development for dummies (2.ed.).
ISBN 978-85-7608-848-6

1. Android (Recurso eletrônico). 2. Software de aplicação - Desenvolvimento. 3. Smartphones - Programação. 4. Computadores Tablet - Programação. I. Felker, Donn. II. Título. II. Série.

CDU 004.42:621.395.6
CDD 005.1

Índice para catálogo sistemático:
1. Software de aplicação : Smartphones 004.42:621.395.6

(Bibliotecária responsável: Sabrina Leal Araujo – CRB 10/1507)

ALTA BOOKS
EDITORA

Rua Viúva Cláudio, 291 – Bairro Industrial do Jacaré
CEP: 20970-031 – Rio de Janeiro – Tels.: (21) 3278-8069/8419
www.altabooks.com.br – e-mail: altabooks@altabooks.com.br
www.facebook.com/altabooks – www.twitter.com/alta_books

Sobre os Autores

Michael Burton é o engenheiro Android chefe na Groupon. Ele criou os aplicativos Groupon, Digg, Triplt e OpenTable Android, entre outros. Lançou um projeto no Ônibus Espacial; falou sobre o desenvolvimento de aplicativos Android em conferências em Londres, Boston, Vale do Silício, Rio de Janeiro e outros lugares. Também é autor do RoboGuice, a estrutura de injeção de dependência com fonte aberta usada pelo Google, Facebook e outros. Siga Michael no Twitter (@roboguice) ou verifique o RoboGuice em `http://roboguice.org`.

Donn Felker é um reconhecido consultor e líder no desenvolvimento de software de ponta nas áreas móveis e web. Ele é consultor independente com mais de 10 anos de experiência profissional em vários mercados que incluem entretenimento, saúde, varejo, seguros e mercados financeiro e mobiliário. Viciado em tecnologia móvel, é empreendedor serial e inovador criativo em todas as coisas móveis e da web. Ele é fundador da Agilevent, uma firma inovadora de desenvolvimento criativo que fez trabalhos para pequenas empresas iniciantes, assim como empresas Fortune 500. É um profundo conhecedor do Microsoft ASP, especialista MCTS para o .NET 2.0 e Aplicativos Web 3.5, e possui certificação ScrumMaster. Dá palestras sobre assuntos que incluem Android, .NET e arquitetura de software. Ele é autor da série de vídeos TekPub.com Introduction to Android. É escritor, apresentador e consultor em vários assuntos, variando desde a arquitetura até o desenvolvimento em geral, práticas ágeis, padrões e atividades. Siga Donn no Twitter (@donnfelker) ou leia seu blog em `http://blog.donnfelker.com`.

Dedicatória

A BugDroid.

Agradecimentos do Autor

Obrigado a Donn Felker por escrever a versão inicial deste livro e colocar o projeto no meu caminho. Espero que trabalhemos em muitos projetos bem-sucedidos no futuro!

Um grande obrigado à extensa comunidade de fonte aberta Android, inclusive a Carlos Sessa, Manfred Moser, Don e Jake Wharton, entre outros, que contribuíram com seu código, conhecimento e revisões deste livro.

Obrigado à minha grande equipe no Groupon, Chris, Alex, Robyn, Eric, Aubrey e David, que me levaram a entender a plataforma Android mais profundamente do que eu teria conseguido sozinho.

E, finalmente, obrigado a meus amigos e família, que me deram apoio nas noites que passei trabalhando neste projeto. O filhotinho emprestado e os tratamentos por capítulo foram tudo que eu precisava para conseguir vencer os longos finais de semana!

Sumário Resumido

Introdução 1

Parte I: Os Componentes Básicos do Android 7

Capítulo 1: Desenvolvendo Aplicativos Android Espetaculares 9
Capítulo 2: Preparando o Quartel-general de Desenvolvimento 27

Parte II: Construindo e Publicando Seu Primeiro Aplicativo Android 53

Capítulo 3: Seu Primeiro Projeto Android 55
Capítulo 4: Projetando a Interface do Usuário 91
Capítulo 5: Codificando Seu Aplicativo 113
Capítulo 6: Compreendendo os Recursos Android 151
Capítulo 7: Transformando Seu Aplicativo em um Componente da Tela Inicial 161
Capítulo 8: Publicando Seu Aplicativo na Google Play Store 185

Parte III: Criando um Aplicativo Cheio de Recursos 205

Capítulo 9: Projetando o Aplicativo Task Reminder 207
Capítulo 10: À la Carte com Seu Menu 231
Capítulo 11: Lidando com a Entrada do Usuário 239
Capítulo 12: Sendo Persistente com o Armazenamento de Dados 261
Capítulo 13: Alertando o Usuário com AlarmManager 289
Capítulo 14: Atualizando a Barra de Status do Android 303
Capítulo 15: Trabalhando com a Estrutura de Preferências do Android 313

Parte IV: Tablets 327

Capítulo 16: Desenvolvendo para Tablets 329
Capítulo 17: Portando Seu Aplicativo para os Tablets Android 337
Capítulo 18: Indo Além do Google 351

Parte V: A Parte dos Dez 361

Capítulo 19: Dez Aplicativos de Amostra Gratuitos e SDKs 363
Capítulo 20: Dez Ferramentas para Simplificar Sua Vida no Desenvolvimento 367

Índice 371

Sumário

Introdução 1

Sobre Este Livro 1
Convenções Usadas Neste Livro 2
Penso que... 2
Como Este Livro Está Organizado 3
Parte I: O Básico do Android 3
Parte II: Construindo e Publicando Seu Primeiro Aplicativo Android 3
Parte III: Criando um Aplicativo Cheio de Recursos 3
Parte IV: Tablets 4
Parte V: A Parte dos Dez 4
Ícones Usados Neste Livro 4
De Lá para Cá, Daqui para Lá 5

Parte I: Os Componentes Básicos do Android 7

Capítulo 1: Desenvolvendo Aplicativos Android Espetaculares 9

Por que Desenvolver para o Android? 9
Participação no mercado 10
Colocação no mercado 10
Plataforma aberta 10
Compatibilidade cruzada 11
Capacidade mashup 11
O Básico do Desenvolvimento Android 12
Java: Sua linguagem de programação Android 13
Atividades 13
Intenções 13
Controles sem cursor 15
Exibições 15
Chamadas assíncronas 15
Serviços em segundo plano 16
Recursos Honeycomb, Ice Cream Sandwich e Jelly Bean 17
Fragmentos 17
Carregadores 18
Biblioteca de suporte do Android 18
Barra de ação 18
Holo 20
Componentes, notificações, desempenho 21
Ferramentas de Hardware 21
Tela de toque 22
GPS 23
Acelerômetro 23

Cartão SD 23
Ferramentas de Software 24
Internet 24
Suporte de áudio e vídeo 24
Contatos 24
Segurança 25
APIs do Google 25

Capítulo 2: Preparando o Quartel-general de Desenvolvimento 27

Desenvolvendo o Desenvolvedor Android Dentro de Você 28
Montando Seu Kit de Ferramentas 28
Kernel do Linux 2.6 28
Estrutura do Android 29
Estrutura do aplicativo 30
Bibliotecas Open Handset Alliance 31
Conhecimento do Java 32
Ajustando Seu Hardware 33
Sistema operacional 33
Hardware do computador 33
Instalando e Configurando Suas Ferramentas de Suporte 34
Obtendo o Kit de Desenvolvimento Java 35
Adquirindo o SDK do Android 36
Download do SDK do Android 36
Seguindo e definindo o caminho de suas ferramentas 38
Obtendo o Eclipse Total 41
Instalando o Eclipse 41
Configurando o Eclipse 43
Navegando o SDK do Android 47
Visando as Plataformas Android 48
Usando Ferramentas SDK para o Desenvolvimento Diário 49
Dizendo olá para o emulador 49
Tornando físico com um dispositivo Android real 49
Depurando seu trabalho 51
Experimentando a API e as amostras SDK 51
Testando os demos da API 52

Parte II: Construindo e Publicando Seu Primeiro Aplicativo Android 53

Capítulo 3: Seu Primeiro Projeto Android 55

Iniciando um Novo Projeto no Eclipse 55
Desconstruindo Seu Projeto 60
Respondendo às mensagens de erro 60
Compreendendo as Definições Build Target e Min SDK Version 62
Configurando um Emulador 63
Criando as Configurações de Inicialização 66
Executando o Aplicativo Hello Android 70

Executando a aplicativo no emulador 70
Verificando o status da implementação 76
Compreendendo a Estrutura do Projeto 77
Navegando as pastas do aplicativo 77
Exibindo o arquivo manifest do aplicativo 86
Exibindo o arquivo project.properties 88
Fechando Seu Projeto 89

Capítulo 4: Projetando a Interface do Usuário 91

Criando o Aplicativo Silent Mode Toggle 92
Layout do Aplicativo 93
Usando o arquivo de layout XML 94
Usando as ferramentas de layout do SDK do Android 96
Usando o designer visual 97
Desenvolvendo a Interface do Usuário 100
Exibindo os atributos do layout XML 101
Trabalhando com as exibições 101
Adicionando uma Imagem ao Seu Aplicativo 102
Colocando uma imagem na tela 102
Adicionando a imagem ao layout 104
Criando um Ícone de Inicialização para o Aplicativo 107
Projetando um ícone de inicialização personalizado 107
Adicionando um ícone de inicialização personalizado 108
Adicionando uma Exibição do Botão de Alternância 109
Visualizando o Aplicativo no Designer Visual 110

Capítulo 5: Codificando Seu Aplicativo 113

Compreendendo as Atividades 113
Trabalhando com métodos, pilhas e estados 114
Controlando o ciclo de vida de uma atividade 115
Criando Sua Primeira Atividade 118
Iniciando com onCreate 118
Informando ao Android para exibir a interface do usuário 119
Lidando com a entrada do usuário 119
Escrevendo sua primeira sub-rotina de eventos 120
Trabalhando com as Classes da Estrutura Android 123
Obtendo um bom serviço 124
Alternando o modo silencioso com AudioManager 125
Instalando Seu Aplicativo 129
Executando seu aplicativo em um emulador 129
Instalando em um dispositivo Android físico 131
Uh-Oh! (Respondendo aos Erros) 133
Usando o servidor do monitor de depuração Dalvik 134
Usando o depurador do Eclipse 139
Pensando Além dos Limites do Aplicativo 147
Interagindo com seu aplicativo 148
Testando se seu aplicativo funciona 149

Capítulo 6: Compreendendo os Recursos Android 151
Compreendendo os Recursos 151
Dimensões 152
Estilos 153
Temas 153
Valores 153
Menus 154
Cores 154
Trabalhando com Recursos 154
Movendo strings para recursos 154
Lutando com a imagem 156
Tornando seus aplicativos globais com recursos 157

Capítulo 7: Transformando Seu Aplicativo em um Componente da Tela Inicial 161
Trabalhando com os Componentes de Aplicativo no Android 162
Trabalhando com exibições remotas 163
Usando AppWidgetProviders 164
Trabalhando com Intenções Pendentes 165
Compreendendo o sistema de intenções do Android 165
Compreendendo os dados da intenção 167
Avaliando as intenções 168
Usando intenções pendentes 169
Criando o Componente da Tela Inicial 170
Implementando a AppWidgetProvider 170
Comunicando-se com o componente do aplicativo 172
Construindo o layout do componente do aplicativo 173
Fazendo o trabalho dentro de uma AppWidgetProvider 174
Trabalhando com os metadados do componente de aplicativo 179
Registrando seus novos componentes no manifest 181
Colocando Seu Componente na Tela Inicial 183

Capítulo 8: Publicando Seu Aplicativo na Google Play Store 185
Criando um Arquivo de Distribuição 185
Revendo o arquivo manifest 186
Escolhendo suas ferramentas 187
Assinando digitalmente seu aplicativo 187
Criando o arquivo APK 189
Criando um Perfil de Desenvolvedor Google Play 192
Colocando Preço em Seu Aplicativo 195
Escolhendo o modelo pago 196
Escolhendo o modelo gratuito 196
Obtendo Capturas de Tela para Seu Aplicativo 197
Fazendo o Upload de Seu Aplicativo para a Google Play Store 198
Observando o Número de Instalações Decolar 202

Parte III: Criando um Aplicativo Cheio de Recursos.....205

Capítulo 9: Projetando o Aplicativo Task Reminder.....207

Revisando os Requisitos Básicas.....207
Agendando um script de lembrete (É alarmante!).....208
Armazenando dados.....208
Distraindo o usuário (gentilmente).....208
Criando as Telas do Aplicativo.....209
Iniciando o novo projeto.....209
Criando a ReminderListActivity.....211
Criando o ReminderListFragment.....212
Usando uma atividade para criar e editar lembretes.....214
Adicionando um fragmento à atividade.....217
Criando o layout para adicionar/editar fragmentos.....220
Completando Seu Fragmento de Lista.....223
Ficando cheio com dados falsos.....224
Lidando com os eventos de clique do usuário.....225
Identificando Sua Intenção.....227
Iniciando novas atividades com intenções.....227
Criando um seletor.....228

Capítulo 10: À la Carte com Seu Menu.....231

Compreendendo os Menus de Opções e Contextuais.....232
Criando Seu Primeiro Menu.....233
Definindo o arquivo XML.....233
Lidando com as ações do usuário.....235
Criando uma tarefa de lembrete.....236
Criando um Menu Contextual.....236
Criando o arquivo XML do menu.....236
Carregando o menu.....237
Lidando com as seleções do usuário.....237

Capítulo 11: Lidando com a Entrada do Usuário.....239

Criando a Interface de Entrada do Usuário.....239
Criando uma exibição EditText.....239
Exibindo um teclado na tela.....241
Sendo Exigente com Datas e Horas.....242
Criando botões para selecionar.....242
Criando o seletor de data.....243
Criando o seletor de hora.....247
Criando uma Caixa de Diálogo de Alerta.....252
Vendo por que você deve trabalhar com caixas de diálogo.....253
Escolhendo a devida caixa de diálogo para uma tarefa.....254
Criando sua própria caixa de diálogo de alerta.....255
Validando a Entrada.....258
Informando o usuário.....259
Usando outras técnicas de validação.....259

Capítulo 12: Sendo Persistente com o Armazenamento de Dados 261
Encontrando Lugares para Colocar os Dados 262
Examinando suas opções de armazenamento 262
Escolhendo uma opção de armazenamento 264
Criando o SQLite ContentProvider de Seu Aplicativo 264
Compreendendo como o SQLite ContentProvider funciona 264
Criando um ContentProvider para manter o código do banco de dados 265
Definindo os principais elementos de um banco de dados 265
Visualizando a tabela SQL 266
Criando a tabela do banco de dados 268
Determinando as URLs do ContentProvider 269
Criando e Editando Tarefas com o SQLite 272
Inserindo uma entrada de tarefa 272
Carregadores 282
Retornando todas as tarefas com um cursor 283
Compreendendo o SimpleCursorAdapter 287
Apagando uma tarefa 287

Capítulo 13: Alertando o Usuário com AlarmManager 289
Vendo Por que Você Precisa do AlarmManager 289
Pedindo a Permissão do Usuário 290
Vendo como as permissões afetam a experiência do usuário 290
Definindo as permissões solicitadas no arquivo AndroidManifest.xml 291
Ativando um Processo com AlarmManager 292
Criando a classe ReminderManager 293
Criando a classe OnAlarmReceiver 294
Criando a classe WakeReminderIntentService 296
Criando a classe ReminderService 298
Reinicializando os Dispositivos 299
Criando um receptor de inicialização 299
Verificando o receptor de inicialização 302

Capítulo 14: Atualizando a Barra de Status do Android 303
Desconstruindo a Barra de Status 303
Usando o Gerenciador de Notificações 307
Criando uma notificação 307
Exibindo o fluxo de trabalho 310
Adicionando recursos de string 310
Atualizando uma Notificação 311
Limpando uma Notificação 311

Capítulo 15: Trabalhando com a Estrutura de Preferências do Android ... 313
Compreendendo a Estrutura de Preferências do Android 314
Compreendendo a Classe PreferenceActivity 315
Mantendo os valores de preferência 316
Layout das preferências 316
Criando Sua Tela de Preferências 317
Construindo o arquivo de preferências 318

Adicionando recursos de string 319
Trabalhando com a Classe PreferenceActivity 320
Abrindo a classe PreferenceActivity 322
Lidando com as seleções de menu 322
Trabalhando com as Preferências em Suas Atividades Durante a Execução 323
Recuperando os valores de preferência 323
Definindo os valores de preferência 325

Parte IV: Tablets *327*

Capítulo 16: Desenvolvendo para Tablets 329
Considerando a Diferença entre Telefones e Tablets 329
Ajustando o Aplicativo Task Reminder para os Tablets 330
Antecipando o tamanho da tela com um layout de fluxo 331
Adicionando mais fragmentos 333
Criando diferentes layouts para diferentes dispositivos 334
Usando a barra de ação 334
Usando a Biblioteca de Suporte e a ActionBarSherlock 336

Capítulo 17: Portando Seu Aplicativo para os Tablets Android 337
Configurando um Emulador Tablet 337
Atualizando o Arquivo AndroidManifest 339
Programando Atividades para os Tablets 339
Criando ReminderListAndEditorActivity 339
Escolhendo a atividade certa 340
Criando o layout da atividade 342
Trabalhando com Fragmentos nos Aplicativos do Tablet 343
Comunicando-se entre os fragmentos 344
Adicionando transações de fragmento 349

Capítulo 18: Indo Além do Google 351
Trabalhando com os Recursos do Google 351
Configurando Seu Kindle Fire ou Emulador 352
Criando um emulador do tipo Kindle 353
Configurando o ADB (Mac) 356
Configurando o ADB (Windows) 356
Publicando na Amazon Appstore para Android 357

Parte V: A Parte dos Dez *361*

Capítulo 19: Dez Aplicativos de Amostra Gratuitos e SDKs 363
Aplicativo Google I/O 2012 363
LOLcat Builder 364
Amazed 364
API Demos 364
HoneycombGallery 365
K-9 Mail 365

Agit 365
SDK do Facebook para Android 365
Replica Island 366
Notepad Tutorial 366

Capítulo 20: Dez Ferramentas para Simplificar Sua Vida no Desenvolvimento 367

droid-fu e ignition 367
RoboGuice 367
Kit de Ferramentas do Tradutor 368
Draw 9-patch 368
Hierarchy Viewer 368
UI/Application Exerciser Monkey 369
zipalign 369
layoutopt 369
Git 369
Paint.NET e GIMP 370

Índice 371

Introdução

Bem-vindo ao *Desenvolvimento de Aplicativos Android Para Leigos, Tradução da 2ª Edição!*

Quando o Android foi adquirido pelo Google em 2005 (sim, o Android foi uma empresa iniciante em algum ponto), muitas pessoas não tinham muito interesse nele porque o Google não tinha entrado ainda no espaço móvel. Avance rapidamente alguns anos, quando a Google anunciou seu primeiro telefone Android: G1. Foi o início de algo enorme.

O G1 foi o primeiro dispositivo Android lançado publicamente. Ele não correspondia ao conjunto cheio de recursos do iPhone na época, mas muitas pessoas acreditaram na plataforma. Assim que o Donut (Android 1.6) foi lançando, ficou evidente que o Google estava esforçando-se em relação ao produto. Imediatamente depois da versão 1.6 ser lançada, a versão 2.0 já estava no horizonte.

Quando o livro foi escrito, estávamos na versão 4.1 da plataforma Android, sem sinais de que as coisas estão perdendo velocidade. Sem dúvida alguma, é um momento de entusiasmo no desenvolvimento Android.

Sobre Este Livro

O *Desenvolvimento de Aplicativos Android Para Leigos, Tradução da 2ª Edição,* é o guia do iniciante para desenvolver aplicativos Android. Você não precisa de experiência em desenvolvimento de aplicativos Android para iniciar. Você pode encarar este material como ponto de partida, porque a plataforma Android executa vários mecanismos usando diferentes paradigmas que a maioria dos programadores não está acostumada a usar — ou desenvolver — diariamente.

A plataforma Android é uma plataforma *independente do dispositivo*, significando que você pode desenvolver aplicativos para vários dispositivos. Esses dispositivos incluem, mas não estão limitados a, telefones, leitores de e-book, netbooks, televisões e dispositivos GPS.

Descobrir como desenvolver para a plataforma Android abre uma grande variedade de opções de desenvolvimento para você. Este livro filtra centenas, se não milhares, de páginas da documentação Android, dicas, truques e

tutoriais em um formato resumido e digerível que o permite saltar para seu futuro como desenvolvedor Android. Este livro não é uma receita, mas fornece o conhecimento básico para montar as várias peças da estrutura Android para criar aplicativos interativos e atraentes.

Convenções Usadas Neste Livro

No livro, você usará as classes do framework Android e irá criar classes Java e arquivos XML.

Os exemplos de código neste livro aparecem em uma fonte mono espaçada, portanto, eles se destacam do texto no livro. Isto significa que o código que você verá é assim:

```
public class MainActivity
```

O Java é uma linguagem de programação de alto nível que leva em conta as letras maiúsculas e minúsculas, portanto, digite o texto no editor exatamente como o vê no livro, pois ele segue as convenções do Java. Portanto, você pode transitar facilmente entre os exemplos do livro e o código de exemplo fornecido pelo Kit de Desenvolvimento de Software (SDK) do Android. Todos os nomes da classe, por exemplo, aparecem no formato `Courier New` e todas as variáveis no nível da classe começam com `m`.

Todas as URLs no livro aparecem também com fonte mono espaçada:

```
http://d.android.com
```

Penso que...

Para começar a programar com o Android, você precisa de um computador que execute um dos seguintes sistemas operacionais:

- Windows XP (32 bits), Vista (32 ou 64 bits), Windows 7 ou 8 (32 ou 64 bits)
- O Mac OS X (Intel) 10.5.8 ou posterior (x86 apenas)
- Linux (i386)

Você também precisa fazer o download do SDK do Android (que é gratuito) e do Kit de Desenvolvimento Java (ou JDK, que também é gratuito), se já não os tiver em seu computador. O Capítulo 2 descreve todo o processo de instalação de todas as ferramentas e estruturas.

Como os aplicativos Android são desenvolvidos na linguagem de programação Java, você precisa entender a linguagem. O Android também usa muito o XML para definir os vários recursos dentro do aplicativo, portanto, você deve entender o XML também. Contudo, você não tem que ser especialista nessas linguagens.

Você não precisa de um dispositivo Android físico porque todos os aplicativos construídos neste livro funcionam em um emulador.

Como Este Livro Está Organizado

O livro *Desenvolvimento de Aplicativos Android Para Leigos, Tradução da 2ª Edição*, tem cinco partes, descritas nas seguintes seções.

Parte I: O Básico do Android

A Parte I apresenta as ferramentas e os frameworks que você usa para desenvolver os aplicativos Android. Também apresenta os vários componentes do SDK e mostra como eles são usados no ecossistema Android.

Parte II: Construindo e Publicando Seu Primeiro Aplicativo Android

A Parte II apresenta a construção de seu primeiro aplicativo Android: o aplicativo Silent Mode Toggle. Depois de construir o aplicativo inicial, você criará um widget para o aplicativo que pode ser colocado na tela inicial de um dispositivo Android. Então, você publicará seu aplicativo na Google Play Store.

Parte III: Criando um Aplicativo Cheio de Recursos

A Parte III leva suas habilidades de desenvolvimento a um novo nível guiando-o na construção do aplicativo Task Reminder, que permite aos usuários criarem várias tarefas com lembretes. Você implementará um provedor de conteúdo SQLite nesse aplicativo com diversas telas. Também verá como usar a barra de status Android para criar notificações que podem ajudar a aumentar a utilização de seu aplicativo.

Parte IV: Tablets

A Parte IV pega o aplicativo de telefone construído na Parte II e ajusta-o para funcionar em um tablet Android. Você também descobrirá como trazer seus aplicativos para os dispositivos Android não Google, tais como o Amazon Kindle Fire.

Parte V: A Parte dos Dez

A Parte V faz um tour pelos aplicativos de amostra que provam ser os principais pontos de partida para seus aplicativos Android e pelas úteis bibliotecas Android que podem facilitar muito sua carreira de desenvolvimento Android.

Ícones Usados Neste Livro

Este ícone indica uma diretriz útil que você não deve pular.

Este ícone representa um lembrete amistoso sobre um ponto vital que você deve ter em mente ao passar por uma seção particular do capítulo.

Este ícone significa que a explicação complementar pode ser informativa, mas não é essencial para entender o desenvolvimento do aplicativo Android. Sinta-se à vontade para pular estes trechos, se quiser.

Este ícone alerta-o sobre problemas em potencial que você pode encontrar pelo caminho. Leia e lembre-se dessas coisinhas para evitar possíveis problemas.

De Lá para Cá, Daqui para Lá

É hora de explorar a plataforma Android! Se você estiver um pouco nervoso, deixe-me assegurar que você não tem que se preocupar; você deve ficar nervoso apenas porque está entusiasmado.

Este livro inclui material extra on-line:

- Não quer digitar todo o código do livro? Você pode fazer download dele no site do livro original em `www.dummies.com/go/androidappdevfd2e`, material que também está disponível no site da Alta Books (procure pelo nome do livro).
- Se houver alguma atualização para este livro, você poderá encontrá-la em `www.dummies.com/go/androidappdevfdupdates2e`.

Parte I

Os Componentes Básicos do Android

A 5ª Onda Por Rich Tennant

"Um aplicativo picada de cobra é o que devemos desenvolver — eu disse. Mas não, você insistiu em um aplicativo de identificação de borboletas."

Nesta parte...

A Parte I apresenta a plataforma Android e descreve o que torna espetacular um aplicativo Android. Você irá explorar as várias partes do kit de desenvolvimento de software (SDK) do Android e entenderá como poderá usá-las em seus aplicativos. Você irá instalar as ferramentas e os frameworks necessários para desenvolver os aplicativos Android.

Capítulo 1

Desenvolvendo Aplicativos Android Espetaculares

Neste Capítulo

- Vendo os motivos para desenvolver aplicativos Android
- Iniciando com o básico do desenvolvimento Android
- Trabalhando com o hardware
- Familiarizando-se com o software

O Google arrasou! Ele adquiriu o projeto Android em 2005 (veja a seção complementar "As raízes do Android", posteriormente neste capítulo) para assegurar que um sistema operacional (SO) móvel pudesse ser criado e mantido em uma plataforma aberta. O Google continua a injetar tempo e recursos no projeto Android, o que já provou ser benéfico. Embora os dispositivos tenham ficado disponíveis apenas em outubro de 2008, quando o livro foi escrito, cerca de um milhão de dispositivos Android são ativados diariamente. Em apenas alguns anos, o Android já causou um *enorme* impacto.

Nunca foi tão fácil para os desenvolvedores Android ganharem dinheiro desenvolvendo aplicativos. Os usuários Android confiam no Google. Como seu aplicativo reside na Google Play Store — controlado pelo conglomerado — muitos usuários Android supõem que seu aplicativo é confiável também.

Por que Desenvolver para o Android?

A pergunta real é: "por que *não* desenvolver para o Android?". Se você quiser que seu aplicativo esteja disponível para milhões de usuários em todo o mundo, se deseja publicar aplicativos assim que terminar de escrevê-los e testá-los, ou se quiser desenvolver em uma plataforma aberta, já tem a sua resposta. Mas, caso ainda esteja indeciso, continue lendo.

Participação no mercado

Como desenvolvedor, você tem uma oportunidade de criar aplicativos para um mercado bem novo — e próspero. Muitos analistas acreditam que o número de dispositivos Android em uso é maior que o número de dispositivos em todos os outros sistemas operacionais móveis combinados. A Google Play Store coloca seu aplicativo direta e facilmente nas mãos dos usuários (ou, mais precisamente, em seus dispositivos). Os usuários não têm que pesquisar na Internet para encontrar um aplicativo para instalar — eles podem simplesmente ir para a Google Play Store, pré-instalada em seus dispositivos, e ter acesso a todos os seus aplicativos. Como a Google Play Store vem pré-instalada na maioria dos aplicativos Android (veja o Capítulo 19 para obter algumas exceções), em geral, os usuários pesquisam-na para todas as suas necessidades de aplicativos. É comum ver o número de downloads de um aplicativo decolar em apenas alguns dias.

Colocação no mercado

Por causa de todas as interfaces de programação do aplicativo (APIs) que vêm no Android, você pode desenvolver facilmente aplicativos cheios de recursos em um intervalo de tempo relativamente curto. Depois de se registrar como desenvolvedor na Google Play Store, simplesmente faça upload de seus aplicativos e os publique. Diferentemente de outros mercados móveis, a Google Play Store não tem nenhum processo de aprovação de aplicativos. Tudo que você tem a fazer é criar os aplicativos e publicá-los.

Embora qualquer pessoa possa publicar qualquer tipo de aplicativo (tecnicamente falando), mantenha sua reputação — e a compatibilidade com os termos de serviço Google — produzindo aplicativos que não sejam ofensivos ou de mau gosto. O Android tem usuários em diversas áreas do mundo e de todas as idades.

Plataforma aberta

O sistema operacional Android é uma *plataforma aberta*: qualquer fabricante ou provedor de hardware pode fabricar ou vender os dispositivos Android. Como você pode imaginar, a abertura do Android permitiu conquistar uma participação no mercado rapidamente. Sinta-se à vontade para examinar o código-fonte do Android — em `http://source.android.com` — para ver como certas tarefas são lidadas, por exemplo. Usando o código fonte aberto, os fabricantes podem até criar interfaces do usuário (IUs) personalizadas e adicionar recursos predefinidos a certos dispositivos.

As raízes do Android

Embora a maioria das pessoas não saiba, o Google não iniciou o projeto Android. A versão inicial do sistema operacional Android foi criada pela Android, Inc., uma pequena empresa startup no Vale do Silício que foi comprada pelo Google em julho de 2005.

Os fundadores (que trabalhavam em várias empresas de tecnologia da internet, tais como a Danger, Wildfire Communications, T-Mobile e WebTV) tornaram-se parte da equipe Google que ajudou a criar o que é, agora, o sistema operacional móvel Android completo.

Compatibilidade cruzada

O Android possui *compatibilidade cruzada (cross-compatibility)*: ele pode ser executado em dispositivos com tamanhos de tela e resoluções muito diferentes, inclusive telefones e tablets. E mais, o Android vem com ferramentas para ajudá-lo a desenvolver aplicativos com compatibilidade cruzada. Entretanto, a Google permite que os aplicativos sejam executados apenas nos dispositivos compatíveis com o Android. Se seu aplicativo requerer uma câmera frontal, por exemplo, apenas os telefones com câmeras frontais poderão "ver" seu aplicativo na Google Play Store — uma solução conhecida como *detecção do recurso* (feature detection). Para obter mais informações sobre como publicar seus aplicativos na Google Play Store, veja o Capítulo 8.

Capacidade mashup

Um *mashup* combina dois ou mais serviços para criar um aplicativo. Você pode criar um mashup usando a câmera e os serviços de localização Android, por exemplo, para tirar uma foto com o local exato exibido na imagem. Ou pode usar a API de um mapa com a lista Contatos, por exemplo, para mostrar todos os contatos em um mapa. Você pode criar facilmente os aplicativos combinando os serviços ou as bibliotecas de incontáveis maneiras novas e interessantes. Outros tipos de mashups que podem ajudar seu cérebro a começar a produzir ideias.

- **Geolocalização e rede social:** suponha que você queira escrever um aplicativo que twitta o local atual de um usuário a cada 10 minutos durante o dia. Usando os serviços de localização Android e uma API Twitter de terceiros (tais como, a iTwitter), você poderá fazer isso facilmente.
- **Geolocalização e jogos:** os jogos baseados na localização, cada vez mais populares, são um modo útil de injetar jogadores no centro do jogo. Um jogo pode executar um serviço em segundo plano para verificar o local atual de um jogador e compará-lo com os locais dos outros jogadores na mesma área. Se um segundo jogador estiver

dentro de uma distância específica, o primeiro poderá ser notificado para desafiá-lo para uma batalha. Tudo isso é possível por causa da tecnologia GPS em uma plataforma forte, tal como o Android.

- **Contatos e Internet:** com todas as APIs úteis à sua disposição, você pode criar facilmente aplicativos cheios de recursos combinando a funcionalidade de duas ou mais APIs. Você pode combinar a Internet e os nomes da lista Contatos para criar um aplicativo de cartão comemorativo, por exemplo. Ou você pode simplesmente querer adicionar um modo fácil para os usuários o contactarem a partir de um aplicativo ou permitir que eles enviem o teu aplicativo para os seus amigos veja "APIs do Google", posteriormente neste capítulo para obter mais informações.

Os desenvolvedores podem fazer com que o Android realize praticamente qualquer coisa que quiserem, portanto, use seu melhor julgamento ao criar e publicar os aplicativos para o consumo em massa. Só porque você deseja um papel de parede dinâmico para destacar sua versão de dança Hula , não significa que alguém queira vê-la.

O Básico do Desenvolvimento Android

Ainda bem que você não tem que ser um membro do Mensa para desenvolver aplicativos Android! Desenvolver no Android é simples porque sua linguagem padrão é o Java. Embora escrever aplicativos Android seja bem fácil, desenvolver sozinho pode ser uma tarefa difícil.

Se você nunca desenvolveu aplicativos antes, este livro não é o melhor lugar para começar a ler sobre o desenvolvimento de aplicativos. Pegue uma cópia do *Começando a Programar em Java Para Leigos*, de Barry Burd (Alta Books) para aprender os truques. Depois de obter uma compreensão básica do Java, você deverá estar pronto para lidar com este livro.

Embora o sistema operacional Android consista basicamente em código Java, pequenas partes da estrutura não estão incluídas. O Android usa a linguagem XML, assim como scripts Apache Ant básicos para construir os processos. Você precisa cimentar sua compreensão básica do XML antes de mergulhar neste livro.

Se você precisar de uma introdução para o XML, verifique o *XML For Dummies*, de Lucinda Dykes e Ed Tittel (Wiley).

Se você já sabe como usar o Java e o XML, parabéns — você tem uma vantagem.

Java: Sua linguagem de programação Android

Os aplicativos Android são escritos em Java — não a versão completa do Java, que é familiar para os desenvolvedores que usam a plataforma Java Enterprise Edition (J2EE), mas um subconjunto das bibliotecas Java que são específicas do Android. Esse subconjunto do Java exclui as classes que não são adequadas para os dispositivos móveis. Se você tiver experiência no Java, deverá sentir-se em casa ao desenvolver aplicativos no Android.

Mesmo com um livro de consulta Java à mão, você sempre poderá pesquisar em `www.google.com` para encontrar informações sobre os tópicos que não compreende. Como o Java não é uma linguagem nova, você pode encontrar muitos exemplos na Web que demonstram como fazer realmente qualquer coisa.

Nem toda classe disponível para os programadores Java está disponível também no Android. Verifique se está disponível antes de começar a tentar usá-la. Se não, provavelmente uma alternativa própria do Android pode funcionar para suas necessidades.

Atividades

Um aplicativo Android pode consistir em apenas uma atividade ou várias. Uma atividade serve como um contêiner para a interface do usuário e o código que a executa. Você pode considerar as atividades como *páginas* de seu aplicativo — uma página em seu aplicativo corresponde a uma atividade. As atividades serão analisadas com mais detalhe nos Capítulos 3 e 5.

Intenções

As *intenções* compõem o sistema de mensagens básico executado no Android. Uma intenção é composta por dois elementos:

- **Uma ação:** a ação geral a ser realizada (tal como exibir, editar ou discar) quando a intenção é recebida.
- **Dados:** as informações nas quais a ação opera, tais como o nome de um contato.

As intenções são usadas para iniciar as atividades e comunicar-se entre as várias partes do sistema operacional Android. Um aplicativo pode transmitir ou receber uma intenção.

Enviando mensagens com intenções

Quando você transmite/difunde via internet, envia uma mensagem informando ao Android para fazer algo acontecer. A intenção pode informar ao Android para iniciar uma nova atividade de dentro de um aplicativo ou outro aplicativo.

Registrando os receptores da intenção

Enviar uma *intent* não faz algo acontecer automaticamente. Você tem que registrar um *intent receiver* que escuta a *intent* e então informa ao Android o que fazer — seja a tarefa iniciar uma nova atividade, ou iniciar outro aplicativo. Se mais de um receptor puder aceitar determinada intenção, um seletor (crosser) poderá ser criado para permitir ao usuário escolher qual aplicativo usar para completar a atividade — por exemplo, como o aplicativo YouTube permite ao usuário escolher se quer assistir aos vídeos no aplicativo YouTube ou em um navegador.

Vários receivers registrados, tais como, os aplicativos Gmail e Messaging, lidam com as intenções para compartilhar imagens por padrão. Quando você encontra mais de um receptor de intenção possível, um seletor é aberto com uma lista de opções a escolher e pergunta o que fazer: usar o e-mail, transferir mensagens ou outro aplicativo, como mostrado na Figura 1-1.

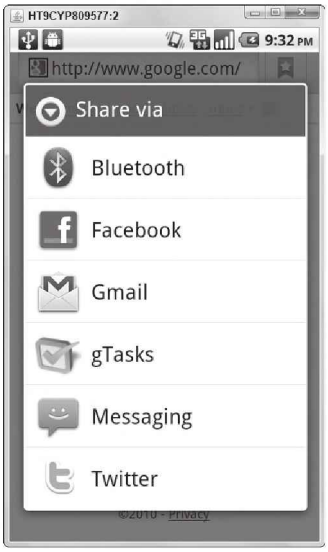


Figura 1-1: Um crosser.

Siga as boas práticas e crie seletores para as intenções que não visam outras atividades dentro de seu aplicativo. Se o sistema Android não puder encontrar uma correspondência para uma intenção que foi enviada e se um seletor não for criado manualmente, o aplicativo irá paralisar depois de ter uma exceção de execução (run-time) — um erro não tratado pelo aplicativo (o Android espera que os desenvolvedores saibam o que estão fazendo).

Controles sem cursor

Diferentemente do PC, onde você manipula o mouse para mover o cursor, um dispositivo Android permite usar seus dedos para fazer praticamente qualquer coisa que você pode fazer com um mouse. Ao invés de clicar com o botão direito, no Android você pressiona por um tempo em um elemento, até seu menu contextual aparecer.

Como desenvolvedor, você pode criar e manipular os menus contextuais. Você pode permitir que os usuários utilizem dois dedos em um dispositivo Android, ao invés de um único cursor do mouse, por exemplo. Existem vários tamanhos de dedos, portanto, projete a interface do usuário em seus aplicativos de acordo. Os botões devem ser grandes o bastante (e ter espaço suficiente) para que até os usuários com os maiores dedos possam interagir facilmente com seus aplicativos, estejam eles usando um telefone ou um tablet.

Exibições

Uma *exibição*, que é um elemento básico da interface do usuário Android, é uma área retangular da tela responsável pelos desenhos e pelo tratamento de eventos. As exibições são o bloco de construção básico das interfaces do usuário Android, da mesma forma que as tags de parágrafo `<p>` ou âncora `<a>`, são blocos de construção de uma página HTML. Algumas exibições comuns que você pode usar em um aplicativo Android podem ser um `TextView`, `ImageView`, `Layout` e `Button`, mas existem dúzias por aí que você pode explorar.

Muito mais exibições estão prontas para o uso. Para obter detalhes completos sobre as exibições, verifique os pacotes `android.widget` e `android.view` na documentação Android em `http://developer.android.com/reference/android/widget/package-summary.html` — conteúdo em inglês.

Chamadas assíncronas

Você usa a classe `AsyncTask` no Android para executar diversas operações ao mesmo tempo sem ter que gerenciar um thread separado por si mesmo. A classe `AsyncTask` não só permite que você inicie um novo processo sem ter que limpar depois, como também retorna o resultado para a atividade que o iniciou — criando um modelo de programação limpo para o processamento assíncrono. Em geral, usamos carregadores (loaders) neste livro, ao invés de `AsyncTask`, mas é útil conhecer as `AsyncTasks` para os casos em que um carregador não fará o que você deseja.

Um *thread* é um processo executado separadamente, mas simultaneamente, a tudo mais que está acontecendo.

Você usa o processamento assíncrono para as tarefas que podem levar mais do que uma pequena fração de segundo, tais como, a comunicação de rede (Internet), ler, gravar, armazenar ou fazer o processamento de mídia. Quando os usuários tiverem que esperar que sua tarefa termine, use uma chamada assíncrona e um elemento na interface do usuário para notificá-los sobre algo que está acontecendo.

Falhar em usar um modelo de programação assíncrono pode fazer com que os usuários de seu aplicativo acreditem que ele está com erros. Fazer download das últimas mensagens Twitter via Internet leva tempo, por exemplo. Se a rede ficar lenta e você não estiver usando um modelo assíncrono, o aplicativo irá travar e o usuário provavelmente irá supor que algo está errado porque o aplicativo não está respondendo à sua interação. Se o aplicativo falhar em responder dentro de um período de tempo razoável (definido pelo sistema operacional Android), o usuário verá a caixa de diálogo Application Not Responding (ANR – Aplicativo Não Respondendo), como mostrado na Figura 1-2. Então, o usuário poderá escolher se é para fechar o aplicativo ou aguardar que ele se recupere.

Figura 1-2: A caixa de diálogo ANR

Para seguir a melhor prática, execute um código de uso intenso da CPU ou de longa execução dentro de outro thread, como descrito na página Designing for Responsiveness (Projetando para a Responsividade) no site do desenvolvedor Android (`http://developer.android.com/guide/practices/design/responsiveness.html` — conteúdo em inglês).

Serviços em segundo plano

Se você for um usuário Windows, pode já saber o que é um *serviço*: um aplicativo executado em segundo plano que não tem necessariamente uma interface do usuário. Um exemplo clássico é um aplicativo antivírus que geralmente é executado em segundo plano como um serviço. Mesmo que você não o veja, sabe que está sendo executado.

A maioria dos reprodutores de música que podem ser baixados da Google Play Store, por exemplo, são executados como serviços em segundo plano. Então, os usuários podem ouvir música enquanto verificam o e-mail ou executam outras tarefas que requerem o uso da tela.

Recursos Honeycomb, Ice Cream Sandwich e Jelly Bean

O Android 3.0, apelidado de Honeycomb, apresentou ao mundo o tablet Android. O Honeycomb e suas versões 3.1 e 3.2 trouxeram muitas mudanças para suportar esta nova classe do dispositivo. O Android 4.0 Ice Cream Sandwich e o 4.1 Jelly Bean aprimoraram as ideias introduzidas no Honeycomb para os tablets e levaram-nas para os telefones, permitindo que os desenvolvedores usassem o mesmo código para suportar os telefones e os tablets em uma única base de código.

As seguintes seções apresentarão alguns dos recursos nessas três versões (que serão cobertos completamente neste livro).

Fragmentos

Cada "página" em um aplicativo Android é uma atividade separada. Nas antigas versões do Android, você colocava qualquer elemento que queria exibir na tela diretamente na classe de atividade. Essa organização funcionava bem quando exibida na tela pequena de um telefone, na qual você geralmente não pode ver muita informação de uma só vez. Você pode conseguir ver uma lista de tarefas ou uma tarefa que está editando, mas comprimir os dois elementos na tela ao mesmo tempo é impossível.

Porém, em um tablet, você está em um contexto mais amplo. Não apenas faz sentido permitir que os usuários vejam uma lista de tarefas e editá-las na mesma página, mas também parece bobo não deixá-los fazer isso. O tamanho da tela em um tablet é simplesmente grande demais para preencher com uma única lista longa de itens ou muito espaço vazio.

O Android não permite colocar facilmente duas atividades na tela ao mesmo tempo. O que fazer? A resposta é o fragmento (fragment).

Usando fragmentos, um único fragmento de lista pode ocupar metade da tela e um fragmento de edição pode ocupar a outra metade. Você poderá descobrir como usar os fragmentos no aplicativo de seu telefone no Capítulo 9 e como dimensionar seu aplicativo para os tablets no Capítulo 17.

Você pode considerar os fragmentos como atividades em miniatura. Como todo fragmento tem seu próprio ciclo de vida, você sabe quando ele está sendo criado e destruído, entre outras informações. Os fragmentos podem ficar dentro das atividades.

Carregadores

Um fragmento é geralmente usado para exibir dados para o usuário. Por exemplo, você pode listar algumas tarefas carregando a lista a partir do banco de dados. Contudo, é importante nunca realizar operações de E/S (entrada e saída) no thread principal da interface do usuário. Se você realizar uma operação do banco de dados no thread principal da interface do usuário, o usuário poderá ver a (temida) caixa de diálogo Aplicativo Não Responsivo, que é invasiva e confusa, geralmente parecendo uma paralisação para muitos usuários.

Um *carregador* (loader) fornece um modo fácil de carregar dados em um thread em segundo plano para que você não atrase o thread da interface do usuário (IU) e congele seu aplicativo. Você pode descobrir mais sobre os carregadores no Capítulo 10.

Biblioteca de suporte do Android

Os fragmentos e os carregadores são maneiras eficientes de adicionar utilidade aos seus aplicativos Android 3.x e 4.x. Porém, você pode precisar suportar antigos dispositivos que usam o Android 1.x e 2.x, que não têm esses novos recursos.

Felizmente, o Android fornece uma solução. Você pode usar a biblioteca de suporte do Android para tornar os fragmentos e os carregadores compatíveis com os dispositivos da Idade da Pedra do Android (cerca de 2009 d.C).

Além de fornecer fragmentos e carregadores, a biblioteca de suporte adiciona vários outros excelentes recursos aos antigos dispositivos, tais como:

- **ViewPager:** corta as páginas à esquerda e à direita
- **GridLayout:** um novo modo de fazer o layout das exibições
- **ShareCompat:** para compartilhar atividades com seus amigos

Visite `http://developer.android.com/tools/extras/support-library.html` (conteúdo em inglês) para ver a lista completa de recursos na biblioteca de suporte do Android.

Barra de ação

O botão Menu é um elemento importante em qualquer aplicativo que usa o Android 1.x ou 2.x. Todos os telefones Android têm (diferentemente de outro tipo popular de smartphone) o botão de Menu no hardware, que pode ser usado para acessar as funções que não são mostradas na tela.

Ou melhor, todos os telefones Android *tinham* esse botão. Começando com o Android 3.0, o Android retirou o botão Menu. Ele ainda aparece em alguns dispositivos, tais como no Samsung Galaxy S III, mas para a maior parte, é uma relíquia do passado. Geralmente falando, os elementos colocados no menu Android não eram fáceis de encontrar e os usuários ainda tendiam a esquecer que eles estavam lá.

No lugar do menu nos dispositivos que usam o Android 3.x e posterior, a *barra de ação (action bar)* está quase sempre presente na parte superior da tela — e é, portanto, extremamente difícil *não* notar. Veja a Figura 1-3 para ter um exemplo da barra de ação do aplicativo YouTube.

Figura 1-3: A barra de ação do YouTube para um divertido vídeo de gatos.

Verifique estes elementos na barra de ação:

- **Botão Up, logotipo do aplicativo:** toque no botão Up ou no logotipo do aplicativo na barra de ação para subir um nível.

  Note a distinção sutil entre o botão Up e o botão Back: pressionar o botão Back retorna o usuário para a atividade anterior, independentemente de qual aplicativo está sendo usado; pressionar o botão Up retorna o usuário à atividade anterior *no aplicativo atual*, mesmo que ele não estivesse utilizando tal atividade.

  Suponha que você esteja visualizando uma página Web no navegador Chrome e toque em um link para abrir o aplicativo YouTube. Pressionar o botão Back irá retorná-lo ao Chrome, pressionar o botão Up irá levá-lo à home page do aplicativo YouTube.

- **Página:** ao lado do ícone do aplicativo na barra de ação, está o título da página atual. Se seu aplicativo permitir que você filtre os dados na página atual, você poderá adicionar um menu suspenso para permitir que os usuários mudem o filtro.

- **Guia:** você pode colocar guias (ao invés do título da página) na barra de ação para permitir que os usuários troquem as guias na atividade atual.

- **Ação:** como você pode ver, há na extremidade direita da barra de ação várias ações que o usuário pode realizar. No aplicativo YouTube mostrado na Figura 1-3, o usuário pode adicionar o vídeo a uma lista, compartilhar o vídeo ou pesquisar mais vídeos. As ações podem

assumir a forma de texto, ícones (como mostrado na figura) ou ambos. Você pode adicionar tantas ações quanto desejar. As ações que não cabem na tela são colocadas em um submenu excedente na extremidade direita.

- **Barra de ação contextual:** a barra de ação pode mudar para mostrar o que o usuário está fazendo. Por exemplo, se um usuário escolher vários itens em uma lista, você poderá substituir a barra de ação padrão por uma barra de ação *contextual* para permitir que os usuários escolham as ações com base nesses itens. Por exemplo, se você quiser permitir exclusões em lote, poderá fornecer um botão Delete Items (Apagar Itens) na barra de ação contextual.

Visite `http://developer.android.com/guide/topics/ul/actionbar.html` (conteúdo em inglês) para obter mais informações sobre a versatilidade que este elemento da interface do usuário pode adicionar ao seu aplicativo.

A barra de ação não existe no Android 2.x e anterior! Não é suportada pela biblioteca de suporte também. Qualquer barra de ação adicionada ao seu aplicativo não aparecerá nessas versões do Android. Mas não desamine, as ações colocadas em sua barra de ação ainda aparecerão no botão Menu para esses dispositivos, portanto, os usuários ainda irão encontrá-las.

Se você estiver interessado em colocar a barra de ação em um aplicativo desenvolvido em uma versão anterior do Android (em um telefone mais antigo ou no Kindle Fire, por exemplo), experimente o ActionBarSherlock em `http://actionbarsherlock.com`.

Holo

O Android 3.0 adiciona três temas holográficos para ajudar a criar belos aplicativos Android:

- Holo Escuro
- Holo Claro
- Holo Claro com barras de ação escuras

Estes temas um pouco mais escuros podem causar algum estranhamento no início, mas são muito mais claros e mais consistentes do que os temas Android 2.x. Os temas Holo são também menos amontoados visualmente, deixando mais espaço para as informações importantes que você deseja que seu aplicativo exiba.

A melhor qualidade do Holo é sua consistência (de longo prazo) em todos os dispositivos Android e fabricantes — um fabricante não pode modificar o tema Holo para fazer sua versão do Android ficar diferente.

Componentes, notificações, desempenho

A lista de novos recursos no Android 3.0, 4.0 e 4.1 parece infinita. Eis uma pequena descrição de alguns destaques:

- **Componentes (widgets):** os componentes estão muito melhores nas versões posteriores do Android. Eles são mais fáceis de encontrar, agora que foram movidos para a lista Applications (Aplicativos). Você pode até adicionar *visualizações de lista* aos componentes para lidar com o corte limitado e a paginação, e pode redimensionar os componentes para ocupar mais ou menos espaço. Na verdade, esses componentes amigáveis redimensionam-se automaticamente quando são arrastados na tela. Essas alterações fazem com que pareçam muito mais vivos e responsivos do que nas versões anteriores.
- **Notificações (notifications):** o Android 4.1 leva novas opções estilosas para o sistema de notificação Android, antes sério. Como as notificações agora podem ser expandidas e reduzidas, o usuário pode ver mais informações sobre elas. Por exemplo, se sua mãe enviar uma foto de seu novo filhotinho em uma mensagem de texto, você poderá vê-la diretamente na notificação sem ter que abrir o aplicativo. Uma notificação sobre uma nova mensagem de e-mail pode mostrar uma visualização do texto da mensagem para que ele possa ser lido diretamente.

  E mais, uma notificação também permite agora que o usuário tome uma ação nela diretamente a partir de qualquer aplicativo sendo usado. Para responder a um e-mail de aniversário da vovó, por exemplo, simplesmente toque no botão Reply (Responder) na notificação para inicializar o Gmail com um editor para que você possa agradecer-lhe.
- **Desempenho:** o Android 4.1 traz importantes melhorias de desempenho para a plataforma. Você não tem que fazer nada especial para aproveitar uma interface mais rápida e mais suave em seu aplicativo — ela será executada suavemente nos dispositivos que executam o Android 4.1 ou posterior.

Ferramentas de Hardware

O Google dá aos desenvolvedores (até aos independentes) as ferramentas necessárias para criar aplicativos móveis excelentes e cheios de recursos. O Google também simplifica explorar e usar todo o hardware disponível em um dispositivo.

Para criar um aplicativo Android espetacular, você deve aproveitar tudo que o hardware tem a oferecer. Não nos entenda mal – se você tiver uma ideia para um aplicativo que não precisa de assistência do hardware, tudo bem, também.

Os dispositivos Android vêm com vários recursos de hardware que você pode usar para construir aplicativos. A Tabela 1-1 descreve os recursos de hardware disponíveis na maioria dos dispositivos Android.

Tabela 1-1 Hardware do dispositivo Android

Recurso de hardware do Android	*O que faz*
Acelerômetro	Indica se o telefone está movendo-se
Rádio Bluetooth	Indica se um fone de ouvido está conectado
Bússola predefinida	Indica em qual direção o usuário está indo
Câmera	Grava vídeo
Rádio GPS	Indica o local do usuário
Sensor de proximidade	Indica se o dispositivo está voltado para cima ou para baixo

A maioria dos dispositivos Android é lançada com o hardware analisado nas quatro seções a seguir, mas nem todos os dispositivos são iguais. O Android é gratuito para os fabricantes de hardware distribuírem, portanto, é usado em uma grande faixa de dispositivos, inclusive alguns feitos por pequenos fabricantes no exterior (e é comum que alguns desses dispositivos não tenham um recurso ou outro).

Os dispositivos Android têm todas as formas e tamanhos: telefones, tablets e leitores de e-book. Você encontrará muitas outras implementações do Android no futuro, tais como a Google TV (um aparelho doméstico acionado pelo Android) e carros com computadores predefinidos com tela de toque e acionados no Android. Os engenheiros por trás do Android fornecem ferramentas que permitem implementar facilmente os aplicativos em diversos tamanhos de tela e resoluções. Não se preocupe — a equipe Android fez todo o trabalho pesado para você. O Capítulo 4 cobre o básico dos tamanhos e das densidades da tela.

Tela de toque

A tela de toque Android oferece um milhão de possibilidades para aperfeiçoar a interação do usuário com seus aplicativos. Os usuários podem cortar (fazer swipe), mover, arrastar ou beliscar para fazer o zoom, por exemplo, movendo um dedo na tela de toque. Você pode até fornecer gestos personalizados em seu aplicativo, o que abre até mais possibilidades.

O Android também suporta a capacidade *multitoque*, que permite a um usuário tocar na tela inteira com mais de um dedo ao mesmo tempo.

Os botões de hardware estão ultrapassados. Você pode colocar botões com qualquer forma em qualquer lugar na tela para criar a interface do usuário mais adequada para seu aplicativo.

GPS

Combinar o sistema operacional Android com o rádio GPS em um dispositivo permite ao desenvolvedor acessar e rastrear o local de um usuário a qualquer momento. O aplicativo de rede social Foursquare é um bom exemplo – ele usa o recurso GPS para determinar o lugar do usuário, então, acessa a Web para determinar as atrações próximas ao usuário.

Outro exemplo útil é a habilidade do aplicativo Maps em apontar o local de um usuário em um mapa e fornecer instruções para o destino dessa pessoa. Combinar o Android com o hardware GPS dá acesso ao local GPS exato do usuário. Muitos aplicativos usam essa combinação para mostrar aos usuários onde está localizado o posto de gasolina mais próximo, cafeteria ou até banheiro.

Acelerômetro

Um *acelerômetro* é um dispositivo que mede a aceleração e o Android vem com o suporte do acelerômetro. O acelerômetro informa se o dispositivo de um usuário está sendo movido ou balançado, e até em qual direção está sendo virado. Então, você usa essas informações como um modo de controlar seu aplicativo.

Você pode usar o acelerômetro para realizar tarefas simples, tais como determinar quando o dispositivo foi virado de cabeça para baixo, então, completar uma ação. Por exemplo, você pode mergulhar os usuários no jogo fazendo-os sacudir seu dispositivo para rolar dados. Este nível de utilidade está separando os dispositivos móveis dos computadores pessoais típicos.

Cartão SD

O Android fornece as ferramentas necessárias para acessar (salvar e carregar) arquivos no cartão *SD* do dispositivo — uma mídia de armazenamento portátil que você pode inserir nos telefones, tablets e computadores compatíveis. Começando com o Android versão 2.2 (Froyo), se um dispositivo estiver equipado com um cartão SD, você poderá usá-lo para armazenar e acessar os arquivos necessários por seu aplicativo. Para evitar inchar seu aplicativo com os recursos extras requeridos e monopolizar a memória predefinida limitada, você poderá fazer download de alguns ou todos os recursos de seu aplicativo a partir do seu host Web e salvá-los no cartão SD do dispositivo (o que torna menos provável que os usuários desinstalem seu aplicativo quando precisarem abrir espaço em seus dispositivos).

Nem todo dispositivo tem um cartão SD pré-instalado, embora a maioria tenha. Sempre assegure-se de que um dispositivo tenha um cartão SD instalado e que o espaço adequado esteja disponível antes de tentar salvar arquivos nele.

Ferramentas de Software

Várias ferramentas do Android estão à sua disposição enquanto você está escrevendo os aplicativos Android. As seguintes seções descreverão algumas das ferramentas mais populares para usar em seu processo diário de desenvolvimento Android.

Internet

Graças às capacidades da Internet dos dispositivos Android, os usuários podem encontrar informações em tempo real na Internet, tais como, a próxima exibição de um novo filme ou a próxima chegada de uma linha de trem. Como desenvolvedor, você pode fazer com que seus aplicativos usem a Internet para acessar os dados atualizados em tempo real, tais como, o tempo, notícias e resultados dos esportes ou (como o Pandora e o YouTube) para armazenar os ícones e os gráficos de seu aplicativo.

Você pode até se livrar dos processos mais intensos de seu aplicativo utilizando um servidor Web sempre que for apropriado para economizar tempo de processamento ou aperfeiçoar o aplicativo. Nesta arquitetura de software bem estabelecida, conhecida como *computação cliente-servidor*, o cliente usa a Internet para fazer uma solicitação para um servidor que está pronto para realizar algum trabalho para seu aplicativo. O aplicativo Maps predefinido é um exemplo de cliente que acessa o mapa e os dados GPS a partir de um servidor Web.

Suporte de áudio e vídeo

Incluir áudio e vídeo em seus aplicativos é moleza no sistema operacional Android. Muitos formatos padrão de áudio e vídeo são suportados e adicionar o conteúdo multimídia aos seus aplicativos — tais como, efeitos sonoros, vídeos instrutivos, música de fundo, streaming (reprodução) de vídeo e áudio da Internet — não poderia ser mais fácil. Seja tão criativo quanto quiser. O céu é o limite.

Contatos

Seu aplicativo pode acessar a lista Contatos de um usuário, que é armazenada no dispositivo, para exibir as informações de contato de um modo novo ou

diferente, ou você pode criar sua própria lista Contatos. Pode até escrever um aplicativo que une as informações de contato com o sistema GPS para alertar o usuário sempre que ele estiver próximo do endereço de um contato.

Não use as informações da lista Contatos de uma maneira maliciosa. Use sua imaginação, mas seja responsável (veja a próxima seção "Segurança").

Segurança

Suponha que alguém lance um aplicativo que envia a lista de contatos inteira de um usuário para um servidor com finalidades maliciosas. Por isto, a maioria das funções que modificam o dispositivo Android de um usuário ou acessam seu conteúdo protegido precisa de permissões especificas. Por exemplo, se você quiser fazer o download de uma imagem na Web, precisará de permissão para usar a Internet para que possa baixar o arquivo para seu dispositivo e precisará de uma permissão separada para salvar o arquivo de imagem em um cartão SD. Quando seu aplicativo estiver sendo instalado, o usuário será notificado sobre as permissões que seu aplicativo está solicitando e poderá decidir se é para prosseguir. Embora pedir permissão não seja opcional, é tão fácil quanto implementar uma linha de código no arquivo manifest de seu aplicativo (os arquivos manifest são descritos no Capítulo 3).

APIs do Google

Os usuários do sistema operacional Android não estão limitados a fazer chamadas, organizar contatos ou instalar aplicativos. Como desenvolvedor, você tem um grande poder em suas mãos — pode até integrar mapas em seu aplicativo, por exemplo. Para tanto, você usará as APIs Maps que contêm os componentes de mapas.

Apontando locais em um mapa

Talvez, você queira escrever um aplicativo que exiba o local atual de um usuário para os amigos. Você pode passar centenas de horas desenvolvendo um sistema de mapeamento ou pode usar a API Maps do Android, que a Google fornece para usar em seus aplicativos. Você pode incorporar a API em seu aplicativo e não ter que investir centenas de horas de desenvolvimento ou mesmo um único centavo. Usando a API Maps, você pode encontrar praticamente qualquer coisa que tenha um endereço. As possibilidades são infinitas — o local de um amigo, a mercearia mais próxima ou seu posto de gasolina favorito, por exemplo.

O princípio KISS

A tarefa mais difícil ao desenvolver os aplicativos é lembrar o princípio KISS: mantenha simples, estúpido (em inglês, Keep It Simple, Stupid). Um modo de complicar desnecessariamente o código criado é mergulhar no desenvolvimento antes de entender o papel das APIs predefinidas. Escolher essa rota pode levar mais tempo do que simplesmente explicar a documentação Android; você não tem que memorizar a documentação, mas faça um favor a si mesmo e, pelo menos, passe os olhos nela. Então, poderá ver como é possível usar facilmente a funcionalidade predefinida — e quanto tempo poderá economizar. Você pode escrever facilmente diversas linhas de código para completar uma tarefa de uma linha. Mudar o volume do media player ou criar um menu, por exemplo, é um processo simples, mas se você não souber como usar as APIs, poderá causar mais problemas tendo que reescrevê-las. Outro modo de bagunçar as coisas é adicionar uma funcionalidade desnecessária — dê aos usuários o modo mais simples de operar seus dispositivos. Por exemplo, evite projetar um layout exagerado com guias personalizadas quando alguns itens de menu seriam suficientes. E mais, o Android vem com componentes suficientes (controles predefinidos) para ajudá-lo a realizar realmente qualquer tarefa. Usar esses controles torna seu aplicativo ainda mais fácil para os usuários trabalharem porque eles já os conhecem e adoram.

Mostrar sua localização atual para os amigos é legal, mas a API Maps do Android também pode acessar a API de Navegação Google para apontar sua localização e mostrar a seus usuários como chegar até ela.

Mensagem nas nuvens

Suponha que os dados de seu aplicativo estejam armazenados na nuvem (a Internet) e você faça download de seus recursos na primeira vez em que ele é executado. Então, você percebe que uma imagem está desatualizada. Para atualizar a imagem, o aplicativo precisa saber que a imagem foi alterada. Você pode usar a estrutura Cloud Messaging da Google para enviar uma notificação da nuvem para o dispositivo (uma mensagem do servidor Web para o dispositivo) para instruir o aplicativo para atualizar a imagem. Este processo funcionará mesmo que seu aplicativo não esteja em execução. Quando o dispositivo receber a mensagem, enviará uma mensagem para iniciar o aplicativo, para que ele possa tomar a devida ação.

Capítulo 2

Preparando o Quartel-general de Desenvolvimento

Neste Capítulo

- Tornando-se um desenvolvedor de aplicativos Android
- Coletando suas ferramentas do negócio
- Fazendo download e instalando o kit de desenvolvimento de software Android
- Obtendo e configurando o Eclipse
- Trabalhando com as ferramentas de desenvolvimento Android

Todo software que você precisa para desenvolver os aplicativos Android é *gratuito*. Esta é a beleza do desenvolvimento dos aplicativos Android. Os blocos de construção básicos necessários para desenvolver aplicativos Android ricos — ferramentas, estruturas e até código-fonte — são gratuitos. Ninguém lhe dará um computador gratuito, mas você pode configurar seu ambiente de desenvolvimento e começar a desenvolver aplicativos gratuitamente e nada pode ser melhor que isso. Bem, talvez uma coisa possa: ser pago para escrever um aplicativo Android, mas você chegará a esse ponto muito em breve.

Este capítulo irá guiá-lo nas etapas necessárias para instalar as ferramentas e as estruturas para que você possa começar a construir aplicativos Android magníficos.

Desenvolvendo o Desenvolvedor Android Dentro de Você

Tornar-se um desenvolvedor Android não é uma tarefa complicada. E provavelmente é mais simples do que você pensa. Para ver do que se trata, faça a si mesmo estas perguntas:

- Eu quero desenvolver aplicativos Android?
- Eu gosto das ferramentas de desenvolvimento de software gratuitas?
- Eu gosto de não pagar taxas de desenvolvimento?
- Eu tenho um computador no qual desenvolver?

Se você respondeu sim a todas as perguntas, hoje é seu dia de sorte — você está pronto para se tornar um desenvolvedor Android. Você pode estar imaginando o que queremos dizer com *não pagar taxas de desenvolvimento*. Você está lendo a pergunta corretamente: você não pagará taxas para desenvolver os aplicativos Android.

Sempre há uma condição, certo? Você pode desenvolver gratuitamente o que bem entender, mas assim que quiser publicar seu aplicativo na Google Play Store, o local onde você faz upload e publica seus aplicativos, precisará pagar uma pequena taxa de registro nominal. Na época em que este livro foi escrito, a taxa era de $25.

Se você estiver desenvolvendo um aplicativo para um cliente, poderá publicar o aplicativo como um pacote redistribuível para lhe dar. Então, seu cliente poderá publicá-lo na Google Play Store, usando a sua conta Google, para assegurar que você não terá que pagar uma taxa pelo trabalho do cliente. Assim, você poderá ser um desenvolvedor Android legítimo e nunca terá que pagar uma taxa. Que *legal*.

Montando Seu Kit de Ferramentas

Depois de saber que está pronto para ser um desenvolvedor Android, pegue seu computador e corra para instalar as ferramentas e as estruturas necessárias para construir seu primeiro aplicativo de sucesso.

Kernel do Linux 2.6

O Android foi criado sobre o kernel do Linux 2.6, de fonte aberta. A equipe do Android escolheu usar esse kernel porque ele forneceu recursos básicos comprovados sobre os quais desenvolver o sistema operacional Android. Os recursos do kernel Linux 2.6 incluem (mas não estão limitados):

- **Modelo de segurança:** o kernel do Linux lida com a segurança entre o aplicativo e o sistema.
- **Gerenciamento da memória:** o kernel lida com o gerenciamento da memória, deixando-o livre para desenvolver seu aplicativo.
- **Gerenciamento de processos:** o kernel do Linux gerencia bem os processos, alocando os recursos para os processos quando são necessários.
- **Pilha de rede:** o kernel do Linux também lida com a comunicação da rede.
- **Modelo do driver:** o objetivo do Linux é assegurar que o aplicativo funcione. Os fabricantes do hardware podem construir seus drivers na arquitetura do Linux.

Você pode ver uma boa amostra do conjunto de recursos do Linux 2.6 na Figura 2-1.

Figura 2-1: Alguns recursos do kernel do Linux.

Estrutura do Android

Sobre o kernel do Linux 2.6, foi desenvolvida a estrutura (o framework) Android com vários recursos. Esses recursos foram obtidos de vários projetos de fonte aberta. A saída desses projetos resultou nestes elementos:

- **Execução Android:** a execução (o runtime) Android é composta de bibliotecas essenciais Java e da máquina virtual Dalvik.
- **Open GL (biblioteca de gráficos):** esta interface para a programação de aplicativos (API) multilinguagem e multiplataforma é usada para produzir gráficos em 2D e 3D.
- **WebKit:** este motor de navegação web de fonte aberta fornece a funcionalidade para exibir o conteúdo web e simplificar o carregamento de páginas.
- **SQLite:** este motor de banco de dados relacional de fonte aberta é projetado para ser incorporado em dispositivos.
- **Estruturas de mídia:** estas bibliotecas permitem reproduzir e gravar áudio e vídeo.
- **Camada de Socket Seguro (SSL):** estas bibliotecas são responsáveis pela segurança da Internet.

Veja a Figura 2-2 para obter uma lista das bibliotecas Android comuns.

Código-fonte Android

Você deve saber que o código-fonte Android completo é de fonte aberta, significando que não é apenas gratuito para usar, mas também gratuito para modificar. Se você quiser fazer download do código-fonte Android e criar uma nova versão do Android, estará livre para fazer isso. Verifique o Android Open Source Project (Projeto de Fonte Aberta Android) em `http://source.android.com` – conteúdo em inglês.

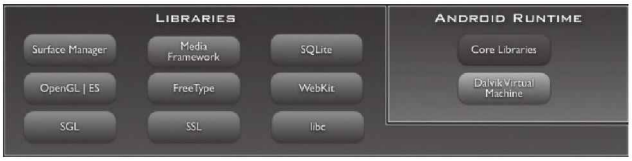


Figura 2-2: O Android e outras bibliotecas de terceiros ficam sobre o kernel do Linux 2.6.

Estrutura do aplicativo

Se você leu a seção anterior, pode dizer: "bem, isso é tudo muito bom, mas como essas bibliotecas me afetam como desenvolvedor?". É simples: todas as estruturas de fonte aberta estão disponíveis via Android. Você não tem que se preocupar sobre como o Android interage com o SQLite e o gerenciador de superfície – você usa-os como ferramentas em seu cinto de ferramentas Android. A equipe Android se baseou no conjunto conhecido de bibliotecas comprovadas, construídas em segundo plano e forneceu-as para você, tudo exposto através das interfaces Android. Essas interfaces empacotaram as várias bibliotecas e tornaram-nas úteis para a plataforma Android e para você, como desenvolvedor. Você aproveita esses recursos porque não tem que construir nenhuma funcionalidade já fornecida por elas.

- **Gerenciador de atividades:** gerencia o ciclo de vida da atividade.
- **Gerenciador de telefonia:** fornece acesso aos serviços de telefonia, assim como a certas informações do assinante, tais como os números de telefone.
- **Sistema de exibição:** lida com as exibições e os layouts que compõem a interface do usuário (IU).
- **Gerenciador de localização:** encontra o local geográfico do dispositivo.

Observe a Figura 2-3 para ver as bibliotecas que compõem a estrutura do aplicativo.

Figura 2-3: Uma visão de parte da estrutura do aplicativo Android.

Do kernel ao aplicativo, o sistema operacional Android foi desenvolvido com tecnologias comprovadas de fonte aberta. Você, como desenvolvedor, pode, portanto, construir aplicativos ricos utilizando estruturas que foram criadas pela comunidade de fonte aberta. Veja a Figura 2-4 para obter uma imagem completa de como a estrutura do aplicativo Android está organizada. A seção Applications (Aplicativos) é onde fica seu aplicativo.

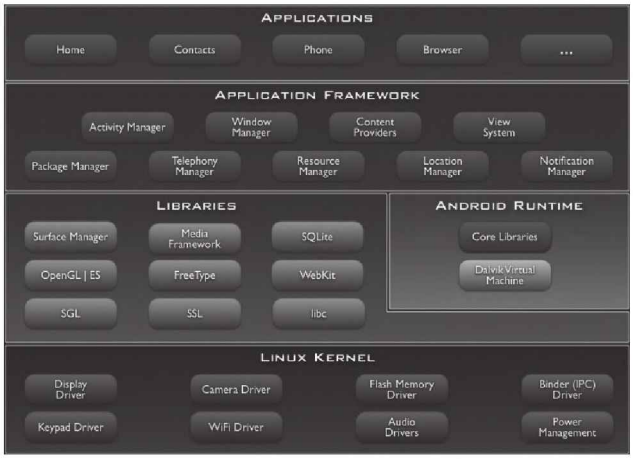


Figura 2-4: Uma visão Como a estrutura do aplicativo Android está organizada.

Algumas vezes, quando você está desenvolvendo um aplicativo Android, deseja usar o mesmo recurso do sistema Android principal. Um bom exemplo é um ícone para uma opção de menu Settings (Definições). Acessando o código-fonte do Android, você pode navegar pelos vários recursos e fazer download dos recursos necessários para seu projeto. Ter acesso ao código-fonte também permite entrar e ver exatamente como o Android faz suas tarefas. Saiba, porém, que você precisa seguir as diretrizes da marca Google ao pegar emprestado esses recursos. Descubra mais em `http://developer.android.com/distribute/googleplay/ promote/brand.html`.

Bibliotecas Open Handset Alliance

Heim? Você não se juntou a uma "aliança", portanto, o que é Open Handset Alliance? Não se preocupe — você não precisa usar A Força para lutar contra Darth Vader. Não é grande coisa e é até legal, como um monte de empresas inteligentes combinando esforços para conseguir o mesmo objetivo.

A Open Handset Alliance (OHA) foi anunciada em novembro de 2007. Na época, a aliança consistia em 34 membros, liderados pelo Google. Com 84 membros quando este livro foi publicado, esse grupo de tecnologia e empresas móveis (inclusive a T-Mobile, Sprint, LG, Motorola, HTC, NVidia e Text Instruments) juntaram-se para buscar inovação no campo móvel e tornar o mundo um lugar melhor. Seu objetivo é fornecer aos usuários aparelhos completos, atraentes e úteis. Você pode ler mais sobre a OHA em `www.openhandsetalliance.com` — conteúdo em inglês.

Você deve conhecer a OHA porque todas as bibliotecas que compõem o sistema operacional Android são baseadas no código-fonte aberto. Todos os membros contribuem de um modo especial. Os fabricantes de chip asseguram que o conjunto de chips suporte a plataforma; os fabricantes de hardware constroem dispositivos; e outras empresas contribuem com a propriedade intelectual (código e documentação, por exemplo). O objetivo é tornar o Android um sucesso comercial.

Quando esses membros contribuem, eles também começam a inovar na plataforma Android. Parte dessa inovação é incorporada no código-fonte Android e parte fica como propriedade intelectual dos membros da aliança, conforme decidido pela OHA.

Só porque um dispositivo tem algo extravagante, não significa que outro dispositivo terá. A única coisa com a qual você pode contar como desenvolvedor é com a estrutura Android principal. Os membros OHA podem ter adicionado uma biblioteca extra para ajudar a facilitar algo em um dispositivo, mas você não tem garantias de que essa biblioteca estará disponível em outro dispositivo, digamos na Turquia ou na Inglaterra. Uma exceção ocorrerá se você estiver desenvolvendo para um determinado dispositivo e apenas esse dispositivo, como, por exemplo, um leitor de e-book. Se esse hardware tiver a única função de ler livros, você poderá programá-lo para essa finalidade específica. Tal como o Barnes & Noble Nook, que utiliza o Android. Ele tem botões Avançar e Voltar especiais que os outros dispositivos Android não têm. Portanto, você programaria para esses botões porque esse dispositivo é um caso especial (se você estiver desenvolvendo para o Nook), mas não poderá esperar que esses botões sejam usados em outros dispositivos.

Conhecimento do Java

A linguagem de programação Java é uma das gloriosas ferramentas que tornam moleza a programação Android, em comparação com a programação em outras plataformas móveis. Enquanto as outras linguagens insistem que você gerencie a memória, desaloque e aloque bytes e mova os bits como um jogo de dominó, a amiguinha do Java, a Máquina Virtual Java (Java Virtual Machine, JVM), ajuda a cuidar disso para você. A JVM permite focar em escrever o código para resolver um problema comercial usando uma linguagem de programação clara e compreensível (ou construir o sensacional

jogo de tiro em primeira pessoa com o qual você vem sonhando), ao invés de focar nas estruturas internas, apenas para fazer as telas aparecerem.

É esperado que você entenda o básico da linguagem de programação Java antes de escrever seu primeiro aplicativo Android. Se você estiver sentindo-se fora de forma e precisar de um curso para lembrar o Java, poderá visitar o site de tutoriais Java em `http://docs.oracle.com/javase/tutorial` — conteúdo em inglês

Embora você encontre alguma informação Java neste livro, poderá querer passar um tempo com um bom livro, tal como o *Java All-in-One For Dummies*, de Doug Lowe (Wiley), se não tiver experiência com o Java.

Ajustando Seu Hardware

Você pode desenvolver aplicativos Android em vários sistemas operacionais, inclusive no Windows, Linux e Mac OS X. Neste livro, você encontra uma combinação do sistema operacional Windows 7 e Mac OS X, mas poderá usar o Linux também.

Sistema operacional

O Android suporta estas plataformas:

- Windows XP ou posterior
- Mac OS X 10.5.8 ou posterior (x86 apenas)
- Ubuntu Linux

Note que as distribuições Linux de 64 bits devem ser capazes de executar os aplicativos de 32 bits. Visite `http://developer.android.com/sdk/installing/index.html` (conteúdo em inglês) para obter mais detalhes.

Neste livro, alguns exemplos usam o Windows 7 Edição de 64 bits. Os caminhos do Windows ficam assim:

```
c:\path\to\file.txt
```

Outros exemplos usam o Mac OS X; um caminho Mac ou Linux fica assim:

```
/path/to/file.txt
```

Hardware do computador

Antes de começar a instalar o software requerido, certifique-se de que seu computador possa executá-lo adequadamente. Praticamente qualquer computador de mesa ou laptop fabricado nos últimos quatro anos será

suficiente. Um laptop com um processador Pentium D de 1.6 GHz e 1 GB de RAM executando o Windows XP e o Windows 7 poderá executar e depurar os aplicativos Eclipse sem problemas (o Eclipse, o software usado para desenvolver seus aplicativos, deve ser executado sem problemas em qualquer computador que você use).

Para assegurar que você poderá instalar todas as ferramentas e estruturas necessárias, certifique-se de que tenha espaço suficiente no disco rígido para acomodá-las. O site do desenvolvedor Android tem uma lista de exigências de hardware, descrevendo quanto espaço em disco rígido cada componente requer, em `http://developer.android.com/sdk/requirements.html`.

Para economizar tempo, você precisa ter cerca de 3 GB de espaço em disco rígido livre para instalar todas as ferramentas e estruturas necessárias para desenvolver os aplicativos Android.

Instalando e Configurando Suas Ferramentas de Suporte

É hora de colocar os empolgantes conceitos Android em ação, mas antes de você poder fazer isso, precisará instalar e configurar algumas ferramentas, inclusive os kits de desenvolvimento do software (SDKs):

- **JDK do Java:** estabelece a base para o SDK do Android.
- **SDK do Android:** fornece acesso às bibliotecas Android e permite que você desenvolva para o Android.
- **IDE (ambiente de desenvolvimento integrado) do Eclipse:** reúne o Java, SDK do Android e as Ferramentas de Desenvolvimento do Android (ADT), e fornece ferramentas para você escrever programas Android.
- **ADT do Android:** faz muito trabalho de base para você, tal como criar os arquivos e a estrutura requeridos para um aplicativo Android.

As seguintes seções mostram como adquirir e instalar todas essas ferramentas.

Uma vantagem de trabalhar com software de código aberto é que, na maioria das vezes, você pode obter ferramentas para desenvolver o software gratuitamente. O Android não é nenhuma exceção à regra. Todas as ferramentas que você precisa para desenvolver aplicativos Android sofisticados são gratuitas.

Obtendo o Kit de Desenvolvimento Java

Por alguma razão, as pessoas responsáveis por nomear o SDK do Java decidiram que seria mais apropriado nomeá-lo como Kit de Desenvolvimento Java ou JDK.

As seguintes etapas funcionam para as máquinas Windows, mas as etapas são parecidas para as máquinas Macs ou Linux. Siga estas etapas para instalar o JDK:

1. **Vá para** `www.oracle.com/technetwork/java/javase/downloads/index.html`.

   A página de downloads do Java SE aparecerá. Veja a Figura 2-5.

2. **Clique no botão Download para a Plataforma Java (JDK).**

   Uma nova página Java SE Downloads aparecerá, pedindo que você especifique qual plataforma (Windows, Linux ou Mac) você está usando para seu trabalho de desenvolvimento.

   A página Web mostrada na Figura 2-5 pode ser diferente no futuro. Para assegurar que você esteja visitando a página correta, visite a página System Requirements (Exigências do Sistema) do SDK do Android na documentação Android on-line para obter um link direto com a página de download do SDK do Java. Veja a página de exigências em `http://developer.android.com/sdk/requirements.html` — conteúdo em inglês.

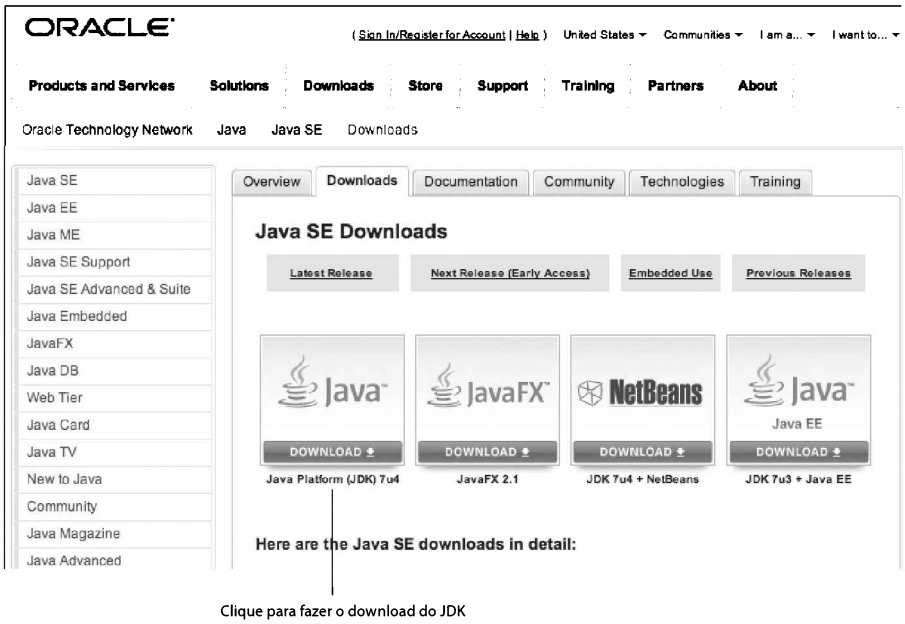


Figura 2-5: Selecione JDK

3. **Clique no link Download do determinado sistema operacional que você está usando.**

   No Windows, escolha a instalação de 32 bits. Se você estiver em uma máquina de 64 bits, poderá instalar os JDKs de 32 bits (x86) ou 64 bits (x64) se quiser, mas terá que instalar o JDK de 32 bits para desenvolver com o Android.

   O Windows poderá abrir uma caixa de mensagem com um aviso de segurança.

4. **Na caixa de diálogo Save As (Salvar Como), selecione o local onde você deseja salvar o arquivo e clique em Save (Salvar).**

5. **Quando o download estiver completo, clique duas vezes no arquivo para instalar o JDK.**

   Uma caixa de diálogo perguntará se você deseja permitir que o programa faça alterações em seu computador.

6. **Clique no botão Yes (Sim).**

   Se você clicar no botão No (Não), a instalação irá parar.

7. **Quando for solicitado, leia e aceite o acordo de licença.**

É tudo. Você instalou o JDK e está pronto para ir para a próxima fase.

Adquirindo o SDK do Android

O SDK do Android é composto por um depurador (debugger), bibliotecas Android, emulador do dispositivo, documentação, código de amostra e tutoriais. Você não pode desenvolver os aplicativos Android sem o SDK.

Download do SDK do Android

Para fazer o download do SDK do Android, siga estas etapas:

1. **Vá para** `http://developer.android.com/sdk/index.html`.

2. **Escolha a última versão do pacote de inicialização do SDK para a sua plataforma para fazer o download do SDK.**

   Você acabou de fazer o download do SDK do Android.

3. **Abra o SDK Manager (Gerenciador SDK).**

   - *Windows*: execute o SDK Installer (Instalação SDK) e instale o SDK no local padrão. Quando terminar, marque a caixa de seleção Start SDK Manager (Iniciar Gerenciador SDK) e clique em

Finish (Terminar). Se você for solicitado a aceitar a autenticidade do arquivo, clique em Yes. A caixa de diálogo Android SDK Manager (Gerenciador SDK do Android) será aberta.

- *Mac*: clique duas vezes no arquivo SDK para descompactá-lo. Mova o diretório `android-sdk-mac_x86` para um lugar seguro, tal como seu diretório Applications (Aplicativos). Abra o Terminal e digite `cd` para ir para o diretório `android-sdk-mac_x86`, então, execute tools/android. Você pode ser solicitado a instalar o Java neste ponto, se ainda não o instalou. Se for o caso, clique em Install (Instalar).

4. Selecione a caixa de seleção SDK Platform Android 4.1.

Para acompanhar este livro, selecione a versão 4.1, como mostrado na Figura 2-6. Na época em que o livro foi escrito, 4.1 era a versão mais recente e maior do Android. Você também deve selecionar as caixas de seleção para a documentação e as amostras de código que correspondem à versão 4.1 do Android (API 16).

Sempre que uma nova versão do sistema operacional Android é lançada, o Google também lança um SDK que contém o acesso à funcionalidade adicionada nessa versão. Se você quiser incluir a funcionalidade Bluetooth em seu aplicativo, por exemplo, certifique-se de que tenha o SDK versão 2.0 do Android ou posterior porque essa funcionalidade não consta nas versões anteriores.

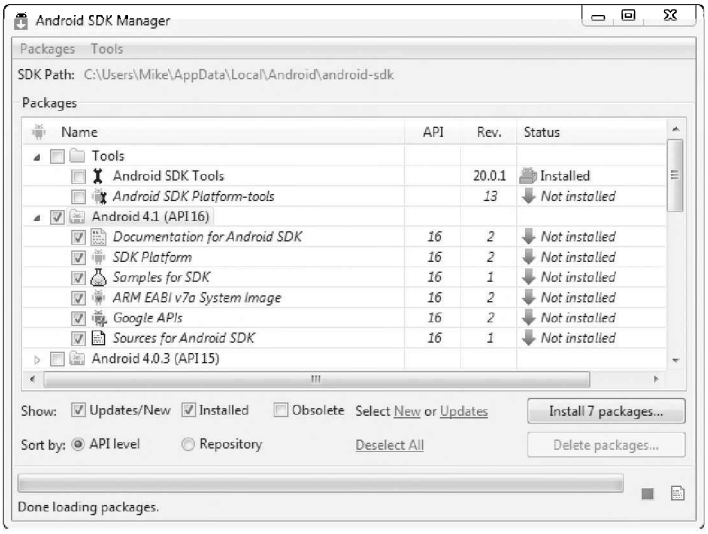


Figura 2-6: Escolha os pacotes para instalar.

5. Clique em Install packages (Instalar pacotes).

A caixa de diálogo Choose Packages to Install (Escolher Pacotes para Instalar) será aberta.

6. **Selecione o botão de rádio Accept (Aceitar) para aceitar a licença, então, clique em Install (Instalar), como mostrado na Figura 2-7.**

   A caixa de diálogo Installing Archives (Instalando Arquivos) será aberta, exibindo uma barra de progresso.

7. **Quando a instalação dos arquivos estiver completa, clique no botão Close (Fechar).**

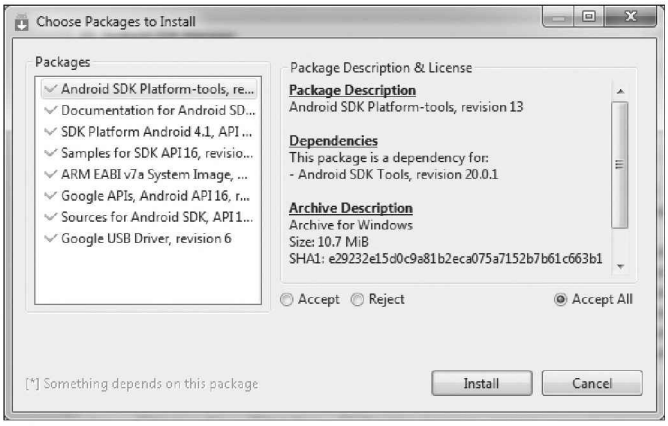


Figura 2-7: A caixa de diálogo Choose Packages to Install.

Enquanto o SDK do Android está tentando conectar os servidores para obter os arquivos, às vezes, você pode ver o erro `Failure to fetch URL` (Falha em obter URL). Se isso acontecer, navegue para Settings (Definições), selecione Force https://... Sources to Be Fetched Using http:// (Forçar https://... Fontes a Serem Buscadas Usando http://), e então tente fazer o download dos pacotes disponíveis de novo.

Seguindo e definindo o caminho de suas ferramentas

Definir o caminho das ferramentas é opcional, mas fazer isso evita ter que lembrar e digitar o caminho completo quando você estiver acessando o Android Debug Bridge (adb ou Ponte de Depuração Android) via linha de comando.

O adb permite gerenciar o estado de um emulador ou dispositivo Android para que você possa depurar seu aplicativo ou interagir com o dispositivo em um alto nível. A ferramenta adb tem muitas capacidades. Para obter informações detalhadas, veja a documentação do Android.

Adicionando o NDK opcional do Android

O conjunto de ferramentas conhecidas como Kit de Desenvolvimento Nativo (NDK) do Android é um conjunto que permite incorporar os componentes que usam o código nativo — o código que você escreveu em uma linguagem nativa, tal como C ou C++. A maioria dos desenvolvedores nem precisará do NDK para construir seus aplicativos. Se você decidir adquirir o NDK, ainda terá que fazer o download do SDK. O NDK não é um substituto do SDK. É um conjunto de funcionalidades extra que complementa o SDK.

Para adicionar as ferramentas Android à sua variável do caminho do sistema em uma máquina Windows (variável PATH), siga estas etapas:

1. **Abra o Painel de Controle e clique duas vezes no ícone para abrir System Preferences (Preferências do Sistema).**

2. **Clique no link Advanced System Settings (Definições Avançadas do Sistema) para abrir a janela Propriedades do Sistema.**

3. **Clique no botão Environment Variables (Variáveis do Ambiente) para abrir a caixa de diálogo Environment Variables.**

4. **Clique no botão Nova.**

   A caixa de diálogo Nova Variável do Sistema será aberta, como mostrado na Figura 2-8.

Figura 2-8: A caixa de diálogo Environment Variables.

5. **No campo Variable Name (Nome da Variável), digite ANDROID.**

6. **Digite o caminho completo para o diretório SDK (`c:\android\android-sdk-windows\`) no campo Variable Value.**

7. **Clique em OK.**

   A caixa de diálogo Environment Variables será aberta, como mostrado na Figura 2-9.

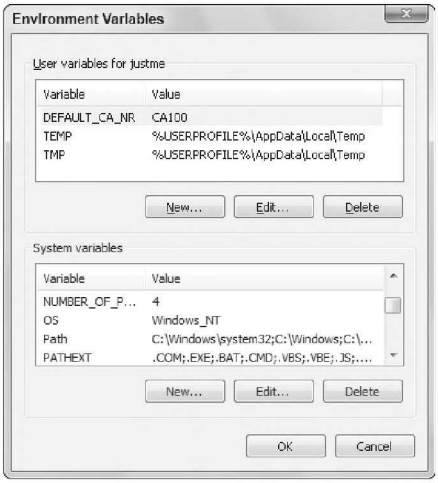


Figura 2-9: Editando a variável PATH.

8. **Na seção System Variables (Variáveis do Sistema), selecione a variável PATH.**

9. **Clique em Edit (Editar), então digite o seguinte texto no final do campo Variable Value (Valor da Variável) e clique em OK.**

```
;%ANDROID%/tools;%ANDROID%/platform-tools
```

Se você estiver em um Mac, abra o arquivo `.profile` em seu diretório principal (/Users/*<seu nomeusuário>*) usando o TextEdit. Se não puder vê-lo, pressione ⌘+Shift+ponto na caixa de diálogo Open File (Abrir Arquivo) para mostrar os arquivos ocultos. No final de `.profile`, adicione o seguinte:

```
export ANDROID=<full path to Android SDK, eg. /Appli-
cations/android-sdk-mac_x86>
export PATH=$PATH:$ANDROID/tools:$ANDROID/platform-
-tools
```

Então, salve o arquivo e reinicie o Mac.

É isso — você terminou. Agora, sempre que você acessar o diretório de ferramentas Android, simplesmente use sua variável ambiente recém-criada.

Na maioria dos sistemas operacionais, a variável `PATH` do sistema não será atualizada até que você faça logout e conecte-se de novo ao seu sistema operacional. Se você achar que os valores de sua variável `PATH` não estão valendo, tente fazer logout de seu computador e inicialize-o de novo.

Obtendo o Eclipse Total

Depois de ter o SDK, você precisará de um ambiente de desenvolvimento integrado (IDE) para usá-lo. É hora de fazer o download do Eclipse!

Instalando o Eclipse

Para fazer o download do Eclipse, navegue para a página de downloads do Eclipse em `www.eclipse.org/downloads`. Selecione Eclipse IDE for Java Developers (Eclipse IDE para Desenvolvedores Java EE funciona também) e faça o download dos arquivos zip.

Para instalar o Eclipse, extraia o conteúdo do arquivo `.zip` do Eclipse para o local escolhido, tal como `C:\Program Files\Eclipse` no Windows ou em sua pasta Applications (Aplicativos) em um Mac.

No Windows, assim que você descompactar o Eclipse, fixe um atalho em seu menu Iniciar, para que o Eclipse seja fácil de encontrar quando você precisar.

Para iniciar o Eclipse, siga estas etapas:

1. **Para executar o Eclipse, clique duas vezes no ícone Eclipse.**

   Se você estiver executando uma versão recente do Windows, na primeira vez em que executar o Eclipse, uma caixa de diálogo Security Warning (Aviso de Segurança) poderá aparecer, como mostrado na Figura 2-10. Ela informa que o fabricante não foi verificado e pergunta se você ainda deseja executar o software. Limpe a caixa de seleção Always Ask Before Opening This File (Sempre Perguntar Antes de Abrir Este Arquivo) e clique no botão Run (Executar).

Figura 2-10: O aviso de segurança do Windows.

2. **Defina seu espaço de trabalho.**

   Quando o Eclipse iniciar, a primeira coisa que você verá é a caixa de diálogo Workspace Launcher (Inicialização do Espaço de Trabalho),

como mostrado na Figura 2-11. Você pode modificar seu espaço de trabalho nela, se quiser, mas para este livro, pode ficar com o padrão:

```
c:\users\<nomeusuário>\workspace
```

no Windows ou

```
\users\<nomeusuário>\workspace
```

em um Mac.

Deixe a caixa de seleção Use This As the Default and Do Not Ask Again (Usar Isto como o Padrão e Não Perguntar Novamente) desmarcada e clique no botão OK.

Se você pretende desenvolver diversos aplicativos, use um espaço de trabalho separado para cada projeto. Se você armazenar diversos projetos em um espaço de trabalho, manter a organização será difícil e você poderá mudar facilmente um arquivo com nome parecido em um projeto diferente. Manter os projetos em seus próprios espaços de trabalho facilita encontrar o projeto quando você tem que voltar a ele para corrigir erros.

Quando o Eclipse terminar de carregar, você verá a tela de boas-vindas, mostrada na Figura 2-12.

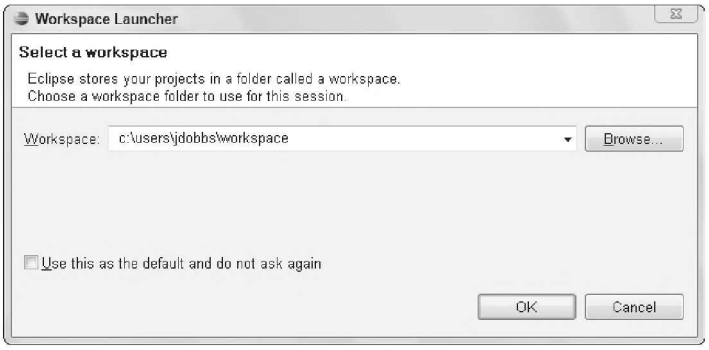


Figura 2-11: Defina seu espaço de trabalho.

Figura 2-12: A tela de boas-vindas do Eclipse.

3. **Clique no ícone de seta curva no lado direito da tela para ir para o workbench.**

   O Eclipse está instalado e tem um acesso fácil. Você irá configurá-lo na próxima seção.

Configurando o Eclipse

O plugin das Ferramentas de Desenvolvimento Android (Android development tools — ADT) adiciona funcionalidade ao Eclipse para fazer muito trabalho por você. O ADT permite

- Criar facilmente novos projetos Android.
- Começar a codificar seu aplicativo rapidamente porque ele cria todos os arquivos básicos necessários.
- Depurar seu aplicativo usando as ferramentas SDK do Android.
- Exportar um arquivo assinado de aplicativo, conhecido como Pacote Android (APK), diretamente do Eclipse, eliminando a necessidade de algumas ferramentas da linha de comando.

Os programadores costumavam precisar de vários utilitários da linha de comando para construir um APK. Embora essa tarefa não fosse difícil, era chata e, algumas vezes, frustrante. O ADT elimina esse processo frustrante guiando-o no "estilo assistente" (wizard) de dentro do Eclipse. Vá para o Capítulo 8 para exportar um APK assinado.

Configurando o Eclipse com o ADT

Para configurar o Eclipse com o ADT, siga estas etapas:

1. **Inicie o Eclipse, se ele ainda não estiver sendo executado.**

2. **Escolha Help (Ajuda)⇨Install New Software (Instalar Novo Software).**

   A janela Install (Instalar) será aberta. Você usará essa janela para instalar novos plugins no Eclipse.

3. **Clique no botão Add (Adicionar) para adicionar um novo site.**

   Um *site* é um endereço Web onde o software fica hospedado na Internet. Adicionar um site ao Eclipse facilita atualizar o software quando uma nova versão é lançada.

   A janela Add Repository (Adicionar Repositório) é aberta, como mostrado na Figura 2-13.

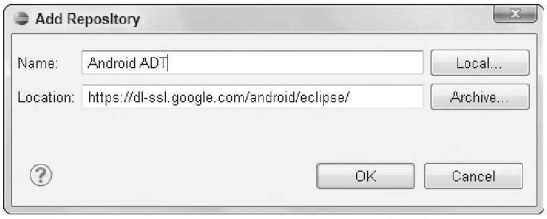


Figura 2-13: Insira o nome e o local do site.

4. **Digite um nome no campo Name (Nome).**

   Este nome pode ser qualquer coisa, mas um fácil de lembrar é Android ADT.

5. **Digite** `https://dl-ssl.google.com/android/eclipse/` **no campo Location (Local).**

6. **Clique em OK.**

   Android ADT é selecionado no menu suspenso Work With (Trabalhar Com) e as opções disponíveis são exibidas na janela Name and Version (Nome e Versão) da caixa de diálogo Install Details (Instalar Detalhes), como mostrado na Figura 2-14.

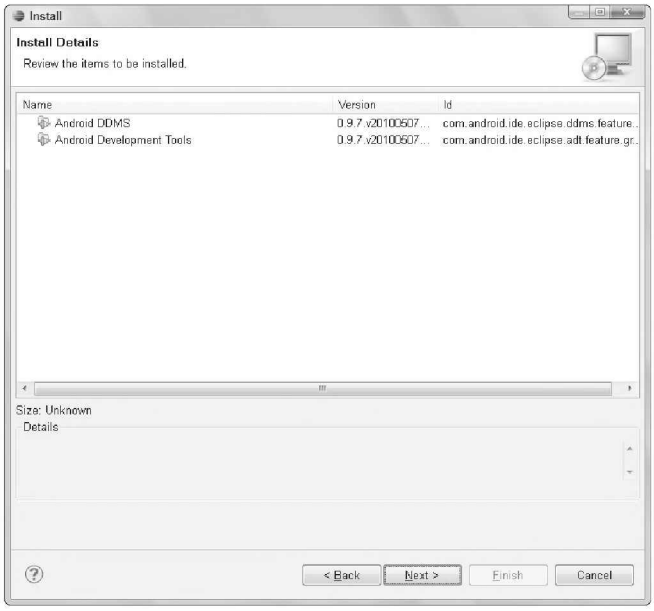


Figura 2-14: Selecione as ferramentas do desenvolvedor.

7. **Selecione a caixa de seleção ao lado de Developer Tools (Ferramentas do Desenvolvedor) e clique no botão Next (Próximo).**

   A caixa de diálogo Install Details deverá listar Android DDMS (veja "Tornando físico com um dispositivo Android real", posteriormente neste capítulo) e ADT. Veja a Figura 2-15.

8. **Clique no botão Next para rever as licenças do software.**

9. **Clique no botão Finish (Terminar).**

10. **Quando solicitado, clique no botão Restart Now (Reiniciar Agora) para reiniciar o Eclipse.**

    O plugin ADT será instalado.

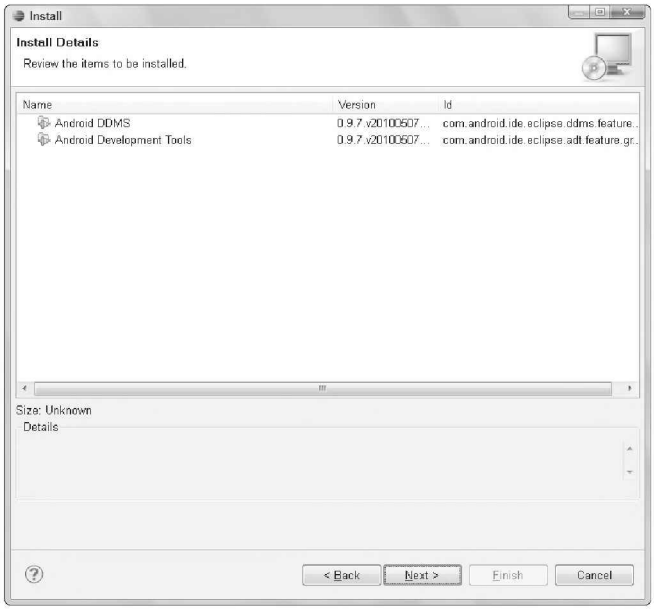


Figura 2-15: DDMS e ADT listados na caixa de diálogo Install Details.

Se você tiver dificuldade para fazer o download das ferramentas em `https://dl-ssl.google.com/android/eclipse`, tente remover o `s` de `https://`, assim: `http://dl-ssl.google.com/android/eclipse`.

Definindo o local do SDK

Esta seção guia-o no processo de configuração. Parece muita coisa a fazer, mas você está quase terminando e tem que fazer este trabalho apenas uma vez. Siga estas etapas:

1. **Escolha Window (Janela)⇨Preferences (Preferências).**

   A caixa de diálogo Preferences será aberta, como mostrado na Figura 2-16.

2. **Selecione Android no painel esquerdo.**

3. **Defina o Local do SDK para a pasta na qual você salvou o SDK do Android.**

   Se você salvou o SDK do Android em `c:\android` em seu computador, o local será `c:\android\android-dsk-windows`.

4. **Clique em OK.**

O Eclipse está configurado e você está pronto para começar a desenvolver os aplicativos Android.

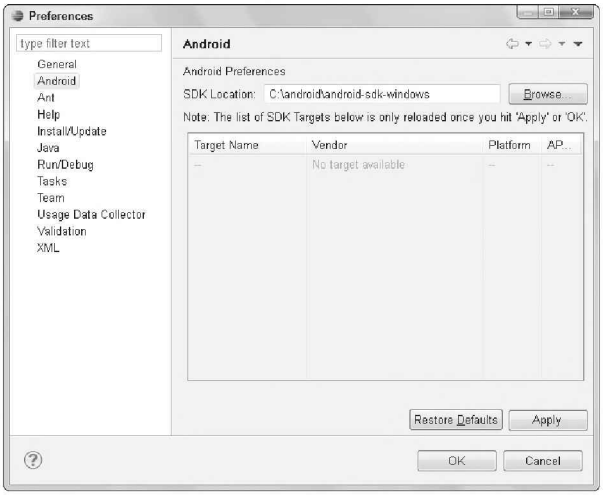


Figura 2-16: Especifique o local do SDK na caixa de diálogo Preferences.

Navegando o SDK do Android

Opa — você encontrou muitas pastas no SDK! Não se preocupe: a estrutura de pastas do SDK do Android é fácil de entender quando você a examina melhor. Você precisa entender a estrutura do SDK para conseguir dominá-la. A Tabela 2-1 resume o conteúdo de cada pasta.

Tabela 2-1 Pastas no SDK do Android

Pasta SDK	*O que contém*
`usb_driver`	Drivers para os dispositivos Android. Se você conecta seu dispositivo Android ao computador, instale esse driver para que possa exibir, depurar e enviar aplicativos ao seu telefone via ADT. A pasta `usb_driver` não fica visível até que você instale o driver USB.
`tools` e `plataform-tools`	Várias ferramentas que estão disponíveis para usar durante o desenvolvimento — tais como para a depuração, gerenciamento da exibição e construção (building).
`temp`	Um repositório temporário que o SDK utiliza quando precisa de espaço para realizar um trabalho.
`samples`	Projetos de amostra para você brincar. O código-fonte completo está incluído.

continua

Tabela 2-1	Pastas no SDK do Android (Continuação)
Pasta SDK	***O que contém***
`platforms`	As plataformas alvo quando você constrói os aplicativos Android, tais como as pastas nomeadas `android-16` (que é o Android4.1) e `android-8` (que é o Android 2.2).
`docs`	Uma cópia local da documentação SDK do Android.
`add-ons`	APIs adicionais que fornecem uma funcionalidade extra. As APIs da Google nesta pasta incluem a funcionalidade de mapeamento. Essa pasta permanece vazia até você instalar qualquer API Maps da Google.

Visando as Plataformas Android

Plataforma Android é um modo elegante de dizer *versão do Android*. Na época em que o livro foi escrito, muitas versões do Android estavam disponíveis, variando até a versão 4.1. Você pode visar qualquer plataforma escolhida.

Diversas versões do Android ainda são muito usadas em telefones. Se você quiser atingir o maior número de usuários, vise uma versão anterior. Se seu aplicativo requer uma funcionalidade que as plataformas mais antigas não podem suportar, vise a plataforma mais recente. Não faria sentido escrever um dispositivo de conexão Bluetooth visando qualquer plataforma anterior a 2.0 porque as plataformas anteriores não podem usar o Bluetooth.

A Figura 2-17 mostra a porcentagem de cada plataforma em uso desde maio de 2012. para exibir as estatísticas atuais da plataforma, visite `http://developer.android.com/resources/dashboard/platform-versions.html` — conteúdo em inglês.

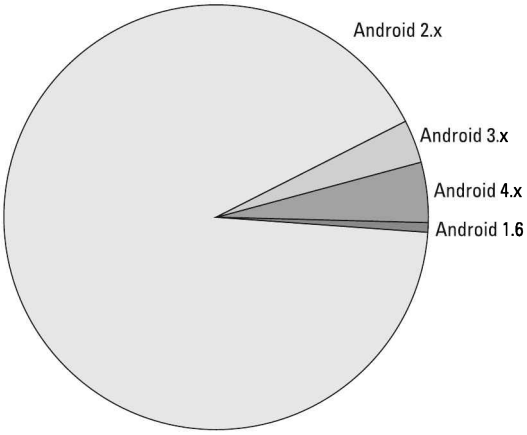


Figura 2-17: Distribuições do Android desde maio de 2012.

Usando Ferramentas SDK para o Desenvolvimento Diário

As ferramentas SDK são os blocos de construção usados ao desenvolver os aplicativos Android. Os novos recursos colocados em cada release permitem que você desenvolva para a última versão do Android.

Dizendo olá para o emulador

A Google fornece não apenas as ferramentas necessárias para desenvolver aplicativos, mas também um pequeno emulador impressionante para testar seu aplicativo. O emulador tem algumas limitações (por exemplo, ele não pode emular certos componentes de hardware, tais como o acelerômetro), mas não se preocupe — muitos aplicativos podem ser desenvolvidos e testados usando apenas um emulador.

Quando você estiver desenvolvendo um aplicativo que usa Bluetooth, por exemplo, deverá usar um dispositivo físico que tenha o Bluetooth. Se você desenvolver em um computador rápido, testar em um emulador será rápido; nas máquinas mais lentas, porém, o emulador poderá levar muito tempo para completar uma tarefa aparentemente simples. Se você estiver desenvolvendo em uma máquina mais antiga, use um dispositivo físico. Quando estiver desenvolvendo em uma máquina mais nova e rápida, use o emulador.

O emulador é útil para testar os aplicativos em diferentes tamanhos de tela e resoluções. Nem sempre é prático ou possível ter vários dispositivos conectados ao seu computador ao mesmo tempo, mas você pode executar vários emuladores com variados tamanhos de tela e resoluções.

Tornando físico com um dispositivo Android real

O emulador é impressionante, mas, algumas vezes, você precisa de um dispositivo físico para testar. O Dalvik Debug Monitor Server (Servidor do Monitor de Depuração Dalvik), DDMS, permite depurar seu aplicativo em um dispositivo real, o que é útil para desenvolver aplicativos que usam recursos de hardware que não são, ou não podem ser, emulados. Suponha que você esteja desenvolvendo um aplicativo que rastreia o local do usuário. Você pode enviar as coordenadas para o dispositivo manualmente, mas em algum ponto em seu desenvolvimento, provavelmente desejará testar o aplicativo para descobrir se ele, de fato, exibe o local correto. Usar um dispositivo real é o único modo de fazer isso.

Se você desenvolve em uma máquina Windows e quiser testar seu aplicativo em um dispositivo real, precisará instalar um driver. Se estiver em uma máquina Mac ou Linux, poderá pular esta seção porque não precisa instalar o driver USB.

Para fazer download do driver USB do Windows para os dispositivos Android, siga estas etapas:

1. **No Eclipse, escolha Window (Janela)⇨Android SDK Manager (Gerenciador SDK do Android).**

   A caixa de diálogo Android SDK Manager será aberta, como mostrado na Figura 2-18.

2. **Expanda o armazenamento Extras e selecione o pacote Google USB Driver.**

3. **Clique no botão Install Packages (Instalar Pacotes).**

   A caixa de diálogo Choose Packages to Install (Escolher Pacotes para Instalar) será aberta, como mostrado na Figura 2-19.

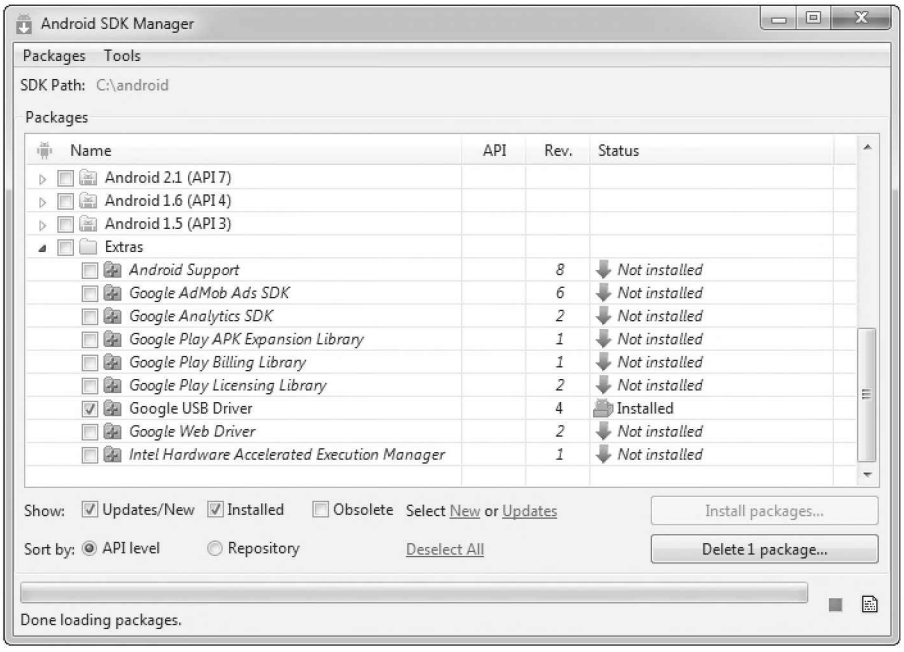


Figura 2-18: Os pacotes disponíveis.

4. **Selecione o botão de rádio Accept (Aceitar) para aceitar a licença, e então, clique no botão Install (Instalar).**

   A caixa de diálogo Installing Archives (Instalando Armazenamentos) será aberta, exibindo uma barra de progresso.

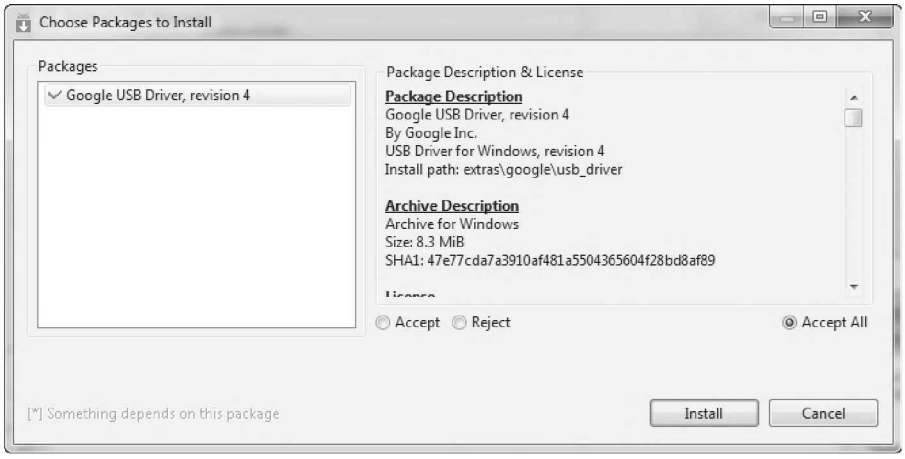


Figura 2-19: Clique no botão Install.

5. **Quando terminar o download e a instalação do pacote, clique no botão Close (Fechar).**

6. **Saia da caixa de diálogo Android SDK Manager.**

Depurando seu trabalho

O DDMS equipa-o com as ferramentas necessárias para encontrar aqueles erros desagradáveis, permitindo que você, internamente enquanto seu aplicativo está em execução, veja o estado de seu hardware, tal como o rádio sem fio. Mas espere — há mais! O DDMS também simula as ações normalmente reservadas para os dispositivos físicos, tais como enviar manualmente as coordenadas do sistema de posicionamento global (GPS), simular chamadas telefônicas ou simular mensagens de texto. Obtenha todos os detalhes do DDMS em `http://developer.android.com/guide/developing/debugging/ddms.html` — conteúdo em inglês.

Experimentando a API e as amostras SDK

As amostras da API e do SDK são fornecidas para demonstrar como usar a funcionalidade oferecida pela API e pelo SDK. Se você ficar travado e não conseguir descobrir como fazer algo funcionar, visite `http://developer.android.com/resources/samples/index.html` (conteúdo em inglês) para encontrar amostras de praticamente qualquer coisa, desde como usar o Bluetooth até criar um aplicativo de texto bidirecional ou um jogo em 2D.

Você também tem algumas amostras em seu SDK do Android. Simplesmente abra o SDK do Android e navegue para o diretório `samples`, que contém várias amostras que variam desde interagir com os serviços até manipular os bancos de dados locais. Passe um tempo brincando com as amostras — a melhor maneira de aprender a desenvolver os aplicativos Android é ver as bases de código de trabalhos existentes, então, experimentá-las no Eclipse.

Testando os demos da API

Os demos da API dentro da pasta `samples` no SDK são uma coleção de aplicativos que demonstram como usar as APIs incluídas. Você pode encontrar aplicativos de amostra com milhares de exemplos, tais como

- Notificações
- Alarmes
- Intenções
- Menus
- Pesquisa
- Preferências
- Serviços em segundo plano

Se você ficar travado ou simplesmente quiser preparar-se para escrever seu próximo aplicativo Android espetacular, verifique os detalhes completos em `http://developer.android.com/resources/samples/ApiDemos/index.html` — conteúdo em inglês.

Parte II

Construindo e Publicando Seu Primeiro Aplicativo Android

A 5ª Onda Por Rich Tennant

"Pare de trabalhar no programa Alocação de Vagas do Estacionamento. Eles querem o desenvolvimento do programa Proximidade Entre as Baias e a Jarra de Café, o mais rápido possível"

Nesta parte...

A Parte II guia-o no desenvolvimento de um aplicativo Android útil. Você começa com o básico das ferramentas Android, e então, entra no desenvolvimento das telas e dos componentes da tela inicial com os quais os usuários irão interagir. Quando o aplicativo estiver completo, você irá assiná-lo digitalmente para que possa implantá-lo na Google Play Store. Você terminará publicando seu aplicativo na Google Play Store.

Capítulo 3

Seu Primeiro Projeto Android

Neste Capítulo

- Criando um projeto novo no Eclipse
- Compreendendo os erros
- Criando um emulador
- Configurando e copiando as configurações de inicialização
- Executando seu primeiro aplicativo
- Estudando a anatomia de um projeto
- Fechando o projeto

Antes de começar a criar o próximo aplicativo Android de sucesso, iremos guiá-lo na criação de seus primeiros aplicativos para ajudar a solidificar alguns aspectos-chave no processo de criação de projetos Android. Neste capítulo, você criará um aplicativo "Hello Android" simples que não requer nenhum código. O quê? Sem código? Como é possível? Siga em frente enquanto mostramos a você.

Iniciando um Novo Projeto no Eclipse

Primeiro, o mais importante: inicie o Eclipse. Você deverá ver uma tela parecida com a mostrada na Figura 3-1. Agora, você está pronto para começar a lidar com o Android.

Se você não configurou ainda seu ambiente de desenvolvimento, volte ao Capítulo 2. Ele mostra como configurar todas as ferramentas e estruturas necessárias para desenvolver os aplicativos Android e como instalar o plugin Ferramentas de Desenvolvimento Android (ADT) do Eclipse. Ele permite a geração de novos aplicativos Android, diretamente através do menu File (Arquivo) do Eclipse.

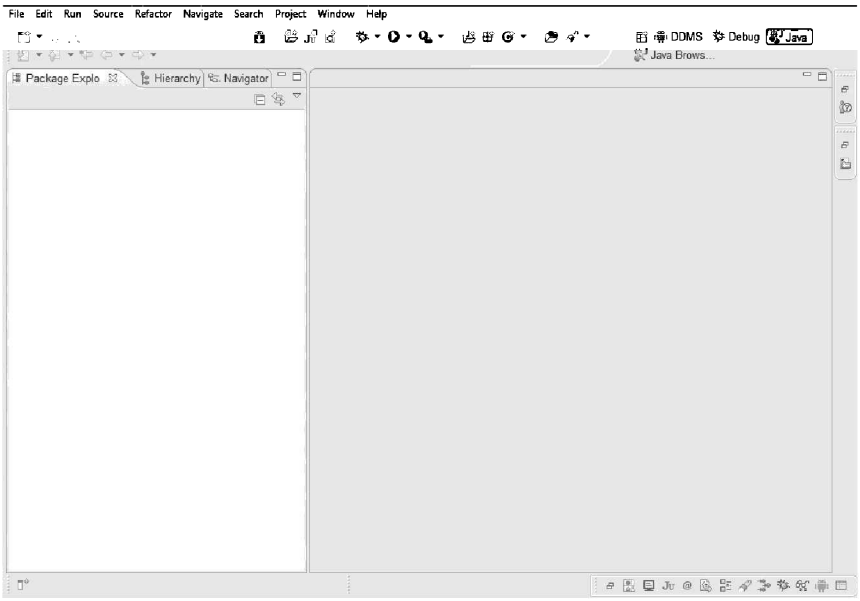


Figura 3-1: O ambiente de desenvolvimento Eclipse.

Siga estas etapas para criar seu primeiro projeto de aplicativo Android:

1. **No Eclipse, escolha File (Arquivo)⇨New (Novo)⇨Other (Outro). Selecione Android Application Project (Projeto de Aplicativo Android).**

   O New Android App Wizard (Assistente para Novo Aplicativo Android) será aberto, como mostrado na Figura 3-2.

2. **Insira** Hello Android **no nome do aplicativo.**

   O nome do aplicativo é o nome que pertence ao Android. Quando o aplicativo é instalado no emulador ou dispositivo físico, esse nome aparece na inicialização do aplicativo.

   Os nomes Project (Projeto) e Package (Pacote) devem ser completados automaticamente para você. O campo Project Name (Nome do Projeto) é importante. Este nome identifica seu projeto no espaço de trabalho Eclipse. Depois da criação do projeto, uma pasta no espaço de trabalho é criada com o mesmo nome do projeto.

   Quando você configurou o Eclipse no Capítulo 2, o sistema pediu a definição do seu espaço de trabalho padrão. O espaço de trabalho geralmente recebe como padrão o seu *diretório principal*, onde o sistema coloca os arquivos relacionados a você. O diretório principal é listado no campo Location (Local).

   Se você preferir armazenar seus arquivos em outro local que não o espaço de trabalho padrão, desmarque o quadro de seleção Create Project in Workspace (Criar Projeto no Espaço de Trabalho), o que ativa a caixa de texto Location (Local). Clique no botão Browse (Navegar) e selecione um local onde você deseja que seus arquivos sejam armazenados.

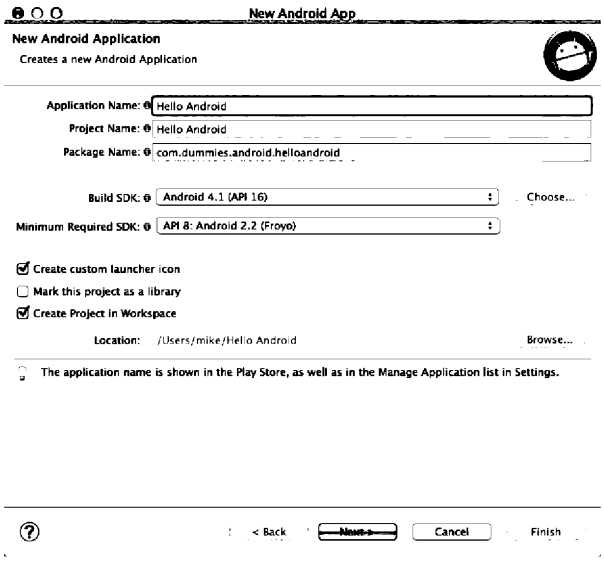


Figura 3-2: O Assistente para Novo Aplicativo Android.

3. **Na caixa Package Name (Nome do Pacote), digite** com.dummies.android.helloandroid.

   Este é nome do pacote Java (veja a seção complementar próxima "Nomenclatura do pacote Java").

Nomenclatura do pacote Java

Um *pacote Java* é um modo de organizar as classes Java em namespaces parecidos com módulos. Cada pacote deve ter um nome exclusivo para as classes que contém. As classes no mesmo pacote podem acessar os membros de acesso de pacote (package) uma das outras.

Os pacotes Java têm uma convenção de nomenclatura definida como o padrão de nomenclatura hierárquico. Cada nível da hierarquia é separado por pontos. Um nome de pacote inicia com o nome de domínio com nível mais alto da organização; então, os subdomínios são listados na ordem inversa. No final do nome do pacote, a empresa pode escolher como ela deseja chamá-lo. O nome do pacote `com.dummies.android.helloandroid` é o nome usado neste exemplo.

Note que o domínio de nível mais alto está na frente do nome do pacote (com). Os subdomínios subsequentes estão separados por pontos. O nome do pacote atravessa os subdomínios para chegar ao nome do pacote final `helloandroid`.

Um ótimo exemplo de outro uso de um pacote é ter um pacote Java para todas as suas comunicações relacionadas à web. Sempre que você precisar encontrar uma classe Java relacionada à web, poderá abrir esse pacote Java e trabalhar com tais classes. Os pacotes permitem a organização do código.

Compreendendo o versionamento no Android

Os códigos de versão não são iguais aos nomes de versão. (Heim?) O Android tem nomes de versão e códigos de versão. Cada nome de versão tem um único código de versão associado. A seguinte tabela descreve os nomes de versão com seus respectivos códigos de versão. Você também pode encontrar essas informações na seção Build Target (Destino da Construção) da caixa de diálogo New Android Project (Novo Projeto Android).

Nome da versão (Nível da plataforma)	Código da versão (Nível da API)
2.0	5
2.01	6
2.1	7
2.2	8
2.3.0 - 2.3.2	9
2.3.3 - 2.3.4	10
3.0	11
3.1	12

4. **Selecione Android 4.1 na lista suspensa Build SDK (Construir SDK) e API 8: Android 2.2 na lista suspensa Minimum Required SDK (SDK Mínimo Requerido), então clique em Next (Próximo).**

   A lista suspensa Build SDK identifica qual interface de programação do aplicativo (API) você deseja desenvolver para este projeto. Sempre defina Build Target SDK (SDK de Destino para a Construção) para a última versão na qual você testou seu aplicativo. Quando você seleciona Android 4.1, constrói e testa seu aplicativo nos dispositivos que suportam até o Android 4.1. Fazer isso permite o desenvolvimento com as APIs do Android 4.1, que incluem novos recursos, tais como o Android Beam. Se você selecionar o Android 2.2 como o destino, não será capaz de usar nenhum recurso suportado pela versão 4.1 (ou 3.1).

   Definir a versão Minimum Required SDK inferior à Build Target SDK significa que seu aplicativo ainda será executado nos antigos dispositivos, até o Android 2.2 neste caso.

   Quando você define valores diferentes para o destino da construção e os SDKs mínimos requeridos , não pode usar nenhuma API mais recente do que o SDK mínimo requerido em dispositivos mais antigos. Por exemplo, você não pode usar o Android Beam em um dispositivo Android 2.2; o aplicativo irá paralisar.

Para obter mais informações, veja a seção "Compreendendo as definições Build Target e Min SDK Version" posteriormente neste capítulo.

O aplicativo Editor de Ícones do Android aparece.

5. **(Opcional) Crie um ícone de aplicativo para seu projeto e clique em Next.**

6. **Na caixa Create Activity (Criar Atividade), escolha BlankActivity e clique em Next.**

   A tela New Blank Activity (Nova Atividade em Branco) aparecerá como mostrado na Figura 3-3.

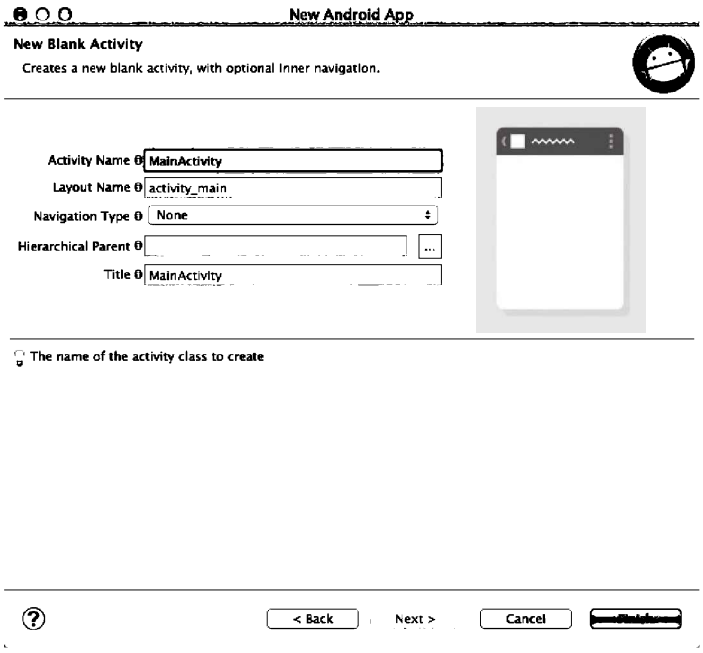


Figura 3-3: Configure sua nova atividade.

7. **Insira** MainActivity **na caixa Activity Name (Nome da Atividade).**

   A tela New Blank Activity (Nova Atividade em Branco) define como é chamada a atividade inicial – o ponto de entrada para seu aplicativo. Quando o Android executa seu aplicativo, este arquivo é o primeiro a ser acessado. Um padrão de nomenclatura comum para a primeira atividade em seu aplicativo é `MainActivity.java`. (que criativo).

8. **Clique no botão Finish (Terminar).**

   Você terminou! Você deverá ver o Eclipse com um único projeto no Package Explorer. Para este livro, o Package Explorer e o Project Explorer (que é exibido na Figura 2-3) são a mesma coisa. Nomes diferentes; mesma função.

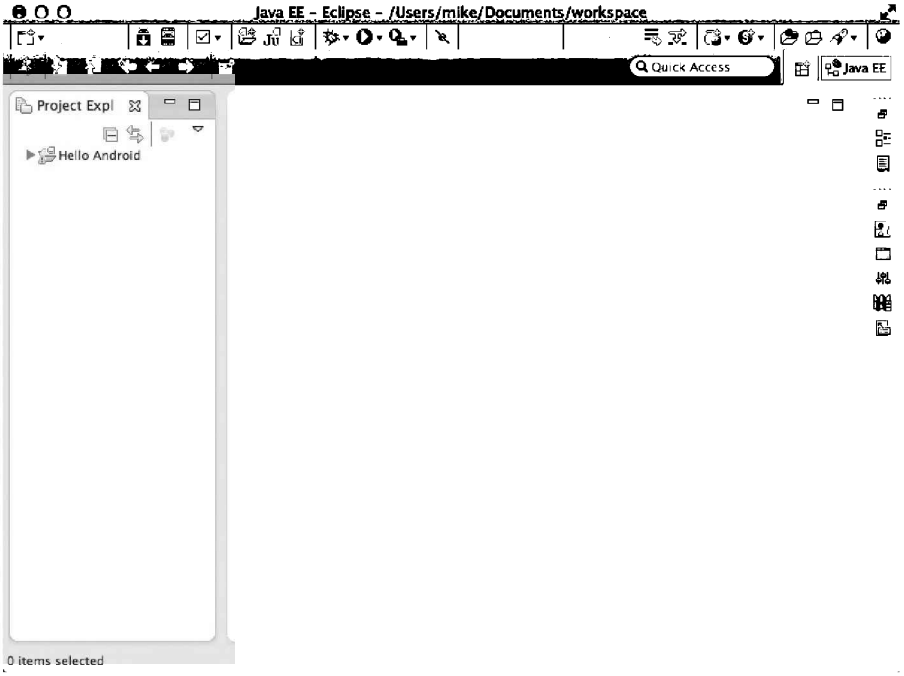


Figura 3-4: O ambiente de desenvolvimento Eclipse com seu projeto Android, Hello Android.

Desconstruindo Seu Projeto

O projeto Android gerado pelo Eclipse é um projeto novo e limpo, sem fontes binárias compiladas. Algumas vezes, é necessário um certo tempo para compreender o Eclipse. Então, você pode notar algumas pequenas esquisitices. Você também precisa entender de uma forma geral o que acontece internamente no Eclipse. É isso que você verá nas próximas seções.

Respondendo às mensagens de erro

Se você fosse rápido o bastante para ver (ou se seu computador for dos mais lentos), poderia ter notado, imediatamente após ter clicado no botão Finish (Terminar), um pequeno ícone vermelho sobre o ícone da pasta Hello Android no Package Explorer em sua janela Eclipse (veja Figura 3-5). Esse ícone é como o Eclipse comunica a você que algo está errado com o projeto no espaço de trabalho.

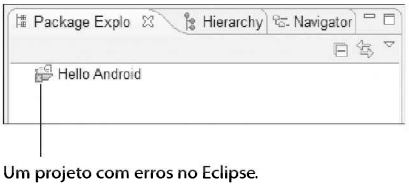


Figura 3-5: Um projeto com erros no Eclipse.

Por padrão, o Eclipse está configurado para permitir que você saiba, com esta dica visual, quando um erro é encontrado em um projeto. Como pode haver um erro neste projeto? Você acabou de criá-lo usando o New Android Project Wizard (Assistente para Novo Projeto Android) – o que está acontecendo? Internamente, o Eclipse e as Ferramentas de Desenvolvimento Android fazem algumas coisa para você:

- **Fornecem o retorno do espaço de trabalho (feedback do workspace):** Este feedback permite que você saiba quando há um problema em qualquer projeto no espaço de trabalho. Você recebe uma notificação no Eclipse via superposições de ícones, tal como mostrado na Figura 3-5. Outra superposição de ícone que você geralmente vê é um pequeno ícone amarelo, que alerta para algumas questões no conteúdo do projeto.
- **Compilam automaticamente:** Por padrão, o Eclipse autocompila os aplicativos em seu espaço de trabalho quando qualquer arquivo é salvo após uma alteração.

Se você não quiser a recompilação automática ativada, poderá desativá-la escolhendo Project (Projeto)⇨Build Automatically (Construir Automaticamente). Este comando desativa a construção automática do projeto. Se essa opção estiver desmarcada, construa seu projeto manualmente pressionando Ctrl+B sempre que você mudar seu código-fonte.

Portanto, por que você está obtendo um erro na primeira construção? Quando o projeto foi adicionado ao seu espaço de trabalho, o Eclipse assumiu o controle e, em conjunto com o ADT, determinou que o projeto no espaço de trabalho tinha um erro. O problema que gerou o erro no Eclipse foi que nenhuma pasta `gen`, nem seu conteúdo, estavam presentes (veja a seção "Compreendendo a Estrutura do Projeto", posteriormente neste capítulo, para obter informações sobre a pasta `gen`).

A pasta `gen` é gerada automaticamente pelo Eclipse e o ADT quando ocorre a compilação. O processo funciona assim:

1. Assim que o New Android Project Wizard termina, o Eclipse cria e salva o novo projeto em seu espaço de trabalho.

2. O Eclipse vê o novo projeto e diz: "Ei! Vejo alguns arquivos novos em meu espaço de trabalho. Preciso informar quaisquer erros encontrados e compilar o projeto".

3. O Eclipse informa os erros colocando um ícone de erro sobre a pasta. (Consulte a Figura 3-5.)

4. O Eclipse compila o projeto.

   Durante a etapa de compilação, a pasta gen é criada pelo Eclipse e o projeto é construído com sucesso.

5. O Eclipse reconhece que o projeto não tem mais erros e remove o ícone de erro da pasta.

   Você ficou com um espaço de trabalho limpo e um ícone de pasta limpo, como mostrado na Figura 3-6.

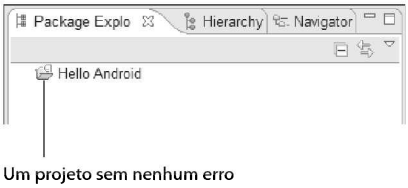


Figura 3-6: Um projeto no Package Explorer que não tem erros.

Compreendendo as definições Build Target e Min SDK Version

O *build target* (destino de construção) é o sistema operacional para o qual você escreve o código. Se você escolheu 4.1, poderá escrever o código com todas as APIs da versão 4.1. Se escolheu 2.2, poderá escrever o código apenas com as APIs que estão na versão 2.2. Você não pode usar as APIs Wi-Fi Direct na versão 2.2, por exemplo, porque elas não foram introduzidas até a versão 4.0. Se você estiver visando a 4.1, poderá escrever com as APIs Wi-Fi Direct.

Saiba qual versão você deseja como alvo antes de começar a escrever seu aplicativo Android. Identifique quais recursos Android você precisa usar para assegurar que seu aplicativo funcione como o esperado. Se você estiver certo de que precisa do suporte Bluetooth, escolha a versão 2.0 ou maior. Se não estiver certo sobre quais versões suportam os recursos que está procurando, poderá encontrar informações específicas sobre as plataformas na seção SDK de `http://d.android.com`. A página da plataforma Android 4.1 está em `http://d.android.com/sdk/android-4.1.html` (conteúdo em inglês).

As versões do sistema operacional (SO) são compatíveis com as anteriores. Se você desenvolver para o Android versão 2.2, por exemplo, seu aplicativo poderá ser executado no Android 4.x, 3.x e, claro, 2.2. A vantagem de utilizar a estrutura 2.2 é que seu aplicativo fica exposto a uma participação de mercado maior. Seu aplicativo poderá ser instalado nos dispositivos com a versão 2.2, 3.0 ou 4.0 (e nas futuras versões, supondo que nenhuma alteração com ruptura da estrutura seja introduzida nas futuras versões do SO do Android). Porém, selecionar uma versão mais antiga tem consequências. Usando uma estrutura mais antiga, você limita a funcionalidade à qual tem acesso. Com a 2.2, por exemplo, você não tem acesso às APIs de mídia social.

Códigos da versão e compatibilidade

A Min SDK Version (Versão SDK Mínima) também é usada pela Google Play Store (tratada no Capítulo 8) para ajudar a identificar quais aplicativos mostrar aos usuários com base em qual versão do Android eles estão executando. Se o dispositivo de um usuário estiver executando a versão 8 (Android 2.2), você desejará que seu aplicativo apareça se ele for compatível com a versão 3, e não com a versão 16 (Android 4.1). A Google Play Store gerencia quais aplicativos mostrar para cada usuário através da definição Min SDK Version.

Se você tiver problemas para decidir qual versão utilizar, o atual gráfico de distribuição de versões poderá ajudar a decidir: `http://developer.android.com/about/dashboards` (conteúdo em inglês).

Uma boa regra é analisar o gráfico de distribuição para determinar qual versão dará ao seu aplicativo a melhor participação de mercado. Quanto mais dispositivos você puder alcançar, maior será seu público; quanto mais instalações tiver, mais bem-sucedido será seu aplicativo.

A definição Min SDK Version (Versão SDK Mínima) é a versão mais antiga do Android que o usuário deve executar para que o aplicativo funcione apropriadamente no dispositivo. O campo não é obrigatório para a construção do aplicativo, mas se você não indicar a Versão SDK Mínima, um valor padrão 1 será usado, indicando que seu aplicativo será compatível com todas as versões do Android.

Se seu aplicativo *não* for compatível com todas as versões do Android (por exemplo, se ele usar APIs que foram introduzidas no código de versão 5 — Android 2.0) e você não declarar a Min SDK Version (Versão SDK Mínima), quando seu aplicativo for instalado em um sistema com um código de versão SDK inferior a 5, ele irá paralisar a execução quando tentar acessar as APIs indisponíveis. Como boa prática, sempre defina Min SDK Version em seu aplicativo para impedir esses tipos de paralisação.

Configurando um Emulador

Você está quase pronto para executar seu aplicativo no Eclipse. Você tem uma última coisa para configurar, e então, verá todo seu trabalho de configuração ganhar vida no aplicativo Hello Android. Para ver esse aplicativo em um estado de execução, você precisa saber como configurar um emulador através de várias configurações de inicialização.

Você precisa criar um Dispositivo Virtual Android (AVD), também conhecido como emulador. Um AVD é um dispositivo Android virtual que parece, age, anda e fala (bem, talvez não ande nem fale) como um dispositivo Android físico. Os AVDs podem ser configurados para executar qualquer versão em particular do Android, contanto que o SDK para essa versão esteja baixada e instalada.

É hora de se familiarizar com os recursos da SDK do Android e do AVD Manager (Gerenciador AVD). Siga estas etapas para criar seu primeiro AVD:

1. **Clique no ícone na barra de ferramentas Eclipse, mostrada na Figura 3-7.**

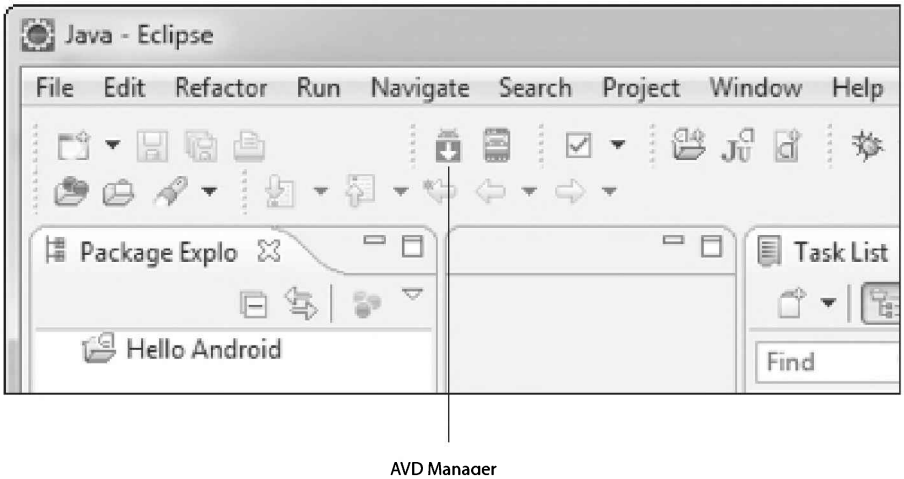


Figura 3-7: O ícone AVD Manager do Android na barra de ferramentas Eclipse.

Nomenclatura AVD

Tenha cuidado ao nomear os AVDs. O Android está disponível em muitos dispositivos no mundo real, tais como telefones, leitores de e-book e netbooks. Chegará uma hora na qual você terá que testar seu aplicativo em várias configurações; portanto, seguir uma nomenclatura comum ao criar os AVDs poderá ajudar, mais tarde, a reconhecer qual AVD é serve para qual finalidade. A seguinte linha pode ajudar a lembrar a finalidade de um AVD:

```
{TARGET_VERSION}_{SKIN}_{SCREENSIZE}[{_Options}]
```

O AVD

```
4_1_Default_WVGA
```

tem o valor `TARGET _ VERSION` do Android 4.1. O nome de versão 4.1 é transformado em `4_1`. Os sublinhados são usados no lugar de pontos para manter o nome do AVD padronizado. Criar um nome AVD como uma única palavra padronizada ajudará quando você estiver trabalhando em cenários avançados com AVDs através da linha de comando.

`SKIN` é o nome da aparência do emulador. Os emuladores podem ter várias aparências que os fazem parecer dispositivos reais. A aparência padrão é fornecida pelo SDK do Android.

O valor `SCREENSIZE` é o tamanho da tela em relação ao tamanho Video Graphics Array (VGA). O padrão é WVGA800. As outras opções são QVGA e HVGA

2. **Clique no botão New (Novo).**

   A caixa de diálogo Create New Android Virtual Device (AVD) (Criar Novo Dispositivo Virtual Android) será aberta, como mostrado na Figura 3-8.

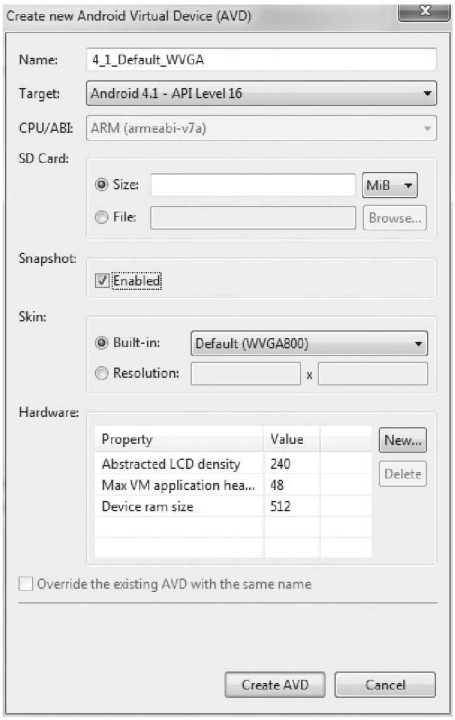


Figura 3-8: A caixa de diálogo Create New Android Virtual Device (AVD).

3. **Para este AVD, no campo Name (Nome), digite** 4_1_Default_WVGA.

   Para obter mais informações sobre a nomenclatura de seus AVDs, veja a seção complementar "Nomenclatura AVD".

4. **Na caixa Target (Destino), selecione Android 4.1 – API Level 16 (API Nível 16).**

5. **Na seção SD Card (Cartão SD), deixe os campos em branco.**

   O Cartão SD não é utilizado no aplicativo Hello Android. Você usaria a opção SD Card se precisasse salvar os dados no cartão SD. Se quiser ter um emulador no futuro, insira o tamanho do cartão SD em megabytes (MB) que você necessita. Nesse momento, um SD Card emulado será criado e colocado em seu sistema de arquivos local.

6. **Deixe a opção Skin (Aparência) definida para Default (WVGA800).**

7. **Não selecione novos recursos na seção Hardware.**

   A seção Hardware descreve os recursos de hardware que seu AVD deve emular. Você não precisa de mais configurações de hardware para o aplicativo Hello Android.

8. **Clique no botão Create AVD (Criar AVD).**

   A Figura 3-9 mostra a caixa de diálogo AVD Manager completa.

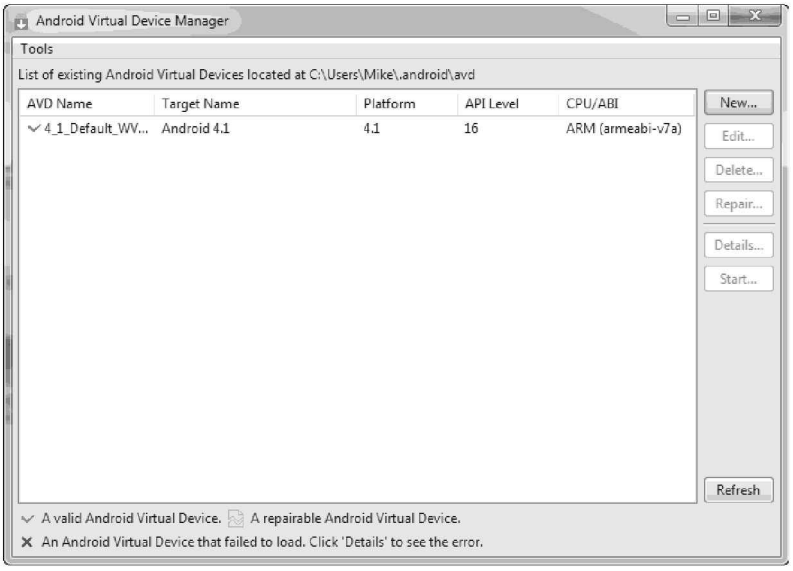


Figura 3-9: O AVD recém-criado na caixa de diálogo Android SDK e AVD Manager.

9. **Feche a caixa de diálogo Android SDK e AVD Manager.**

Você poderá ver esta mensagem de erro após criar seu AVD:

```
Android requires .class compatibility set to 5.0. Please
          fix project properties
```

Se vir, poderá corrigi-la clicando com o botão direito no projeto no Eclipse e escolhendo Android Tools (Ferramentas Android)⇨Fix Project Properties (Corrigir Propriedades do Projeto) no menu contextual.

Criando as Configurações de Inicialização

Você está quase no ponto onde poderá executar o aplicativo. Uma *configuração de execução* especifica o projeto a executar, a atividade a iniciar e o emulador ou dispositivo a conectar. Pare! É muita coisa acontecendo

rapidamente. Não se preocupe — o ADT pode ajudá-lo, automatizando as etapas principais para que você comece a trabalhar rapidamente.

O ADT do Android fornece duas opções para criar as configurações de inicialização:

- **Configuração da execução:** Você precisa executar seu aplicativo em um dado dispositivo. Você usará as configurações da execução várias vezes durante sua carreira como desenvolvedor Android.
- **Configuração da depuração:** Você depurará seu aplicativo enquanto ele estiver sendo executado em um dado dispositivo. Você poderá descobrir como depurar no Capítulo 5.

Quando você executar pela primeira vez um projeto como um aplicativo Android escolhendo Run (Executar)⇨Run, o ADT criará automaticamente uma configuração de execução para você. A opção Android Application (Aplicativo Android) fica visível quando você escolhe Run⇨Run. Depois de criada a configuração de execução, ela será a configuração padrão, usada sempre que você escolher Run⇨Run.

Se você for ambicioso e decidir que deseja criar manualmente uma configuração da execução, acompanhe aqui. Não se preocupe — seguir estas etapas é simples:

1. **Escolha Run⇨Run Configurations (Configurações da Execução).**

   A caixa de diálogo Run Configurations será aberta, como mostrado na Figura 3-10. Nessa caixa de diálogo, você poderá criar muitos tipos de configurações da execução. O lado esquerdo da caixa lista vários tipos de configurações, mas foque nisto:

   - Android Application (Aplicativo Android)
   - Android JUnit Test (Teste JUnit Android)

2. **Selecione o item Android Application e clique no ícone New Launch Configuration (Nova Configuração de Inicialização).**

   Ou clique com o botão direito em Android Application e escolha New (Nova) no menu contextual.

3. **Digite** ExampleConfiguration **no campo Name (Nome).**

Se você quiser iniciar a partir de uma configuração de inicialização existente, clique-a com o botão direito e escolha Duplicate (Duplicar) no menu contextual. Mude o nome da configuração de execução inserindo um nome no campo Name (Nome). Então, poderá mudar as várias definições para tornar a nova configuração de inicialização exclusiva.

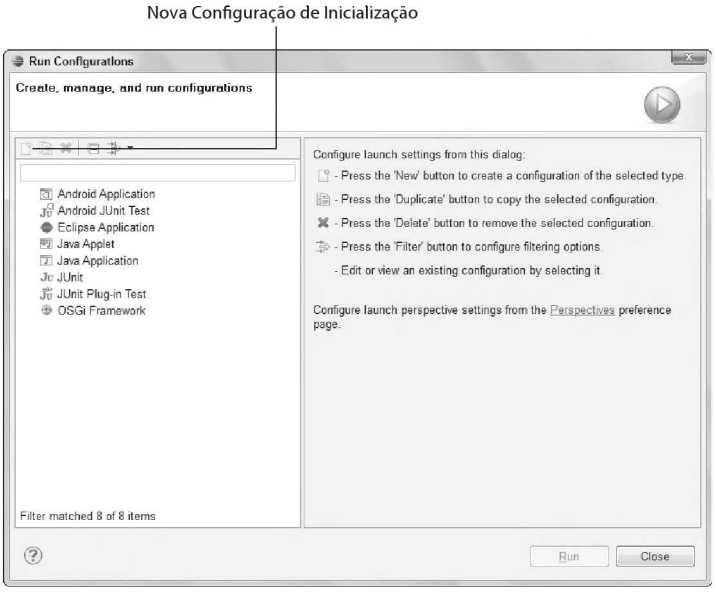


Figura 3-10: A caixa de diálogo Run Configurations (Configurações da Execução).

4. **Na guia Android, clique no botão Browse (Navegar) para selecionar o projeto para o qual você está criando esta configuração de inicialização.**

   A caixa de diálogo Project Selection (Seleção do Projeto) será aberta, como mostrado na Figura 3-11.

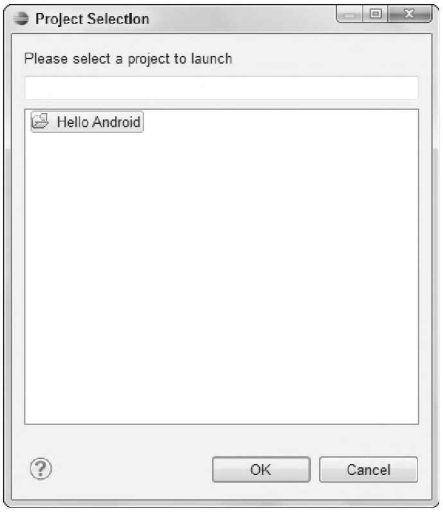


Figura 3-11: Selecionando o projeto para a nova configuração de inicialização.

5. **Selecione Hello Android e clique em OK.**

   A caixa de diálogo Run Configurations será reaberta.

6. **Na guia Android, deixe a opção Launch Action (Ação de Inicialização) definida para Launch Default Activity (Inicializar Atividade Padrão).**

   Neste caso, a atividade padrão é MainActivity, que você configurou em "Iniciando um Novo Projeto no Eclipse", anteriormente neste capítulo.

7. **Na guia Target (Destino), mostrada na Figura 3-12, deixe Automatic (Automática) selecionada.**

   Observe se um AVD está listado na seção Select a Preferred Android Virtual Device for Deployment (Selecionar um Dispositivo Virtual Android Preferido para Distribuição).

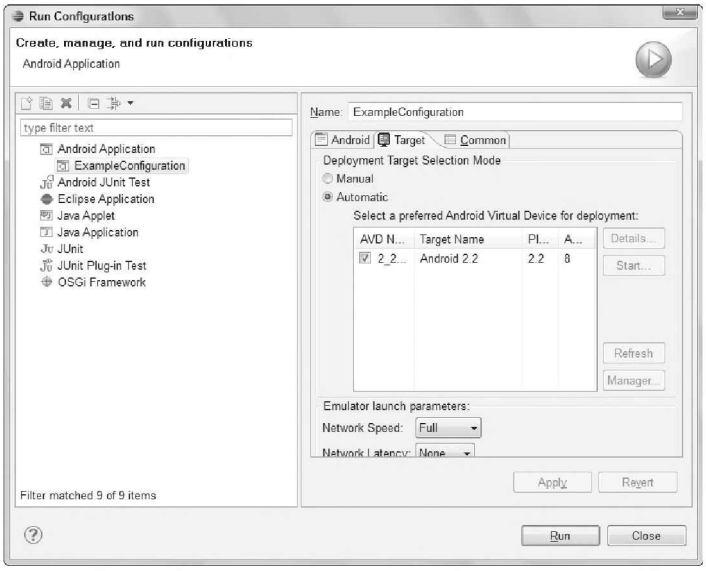


Figura 3-12: Uma nova configuração de inicialização criada manualmente.

8. **Selecione o dispositivo 4_1_Default_WVGA.**

   Este dispositivo é o AVD criado anteriormente. Selecionando-o, você está instruindo essa configuração de inicialização para iniciar esse AVD quando um usuário executar o aplicativo escolhendo Run⇨Run. Esta exibição tem opções manuais e automáticas. A opção manual permite que você escolha qual dispositivo conectar ao usar esta configuração de inicialização. A automática define o AVD predefinido para usar ao inicializar na configuração de inicialização atual.

9. **Deixe o resto das definições como estão e clique no botão Apply (Aplicar).**

   Quando você não precisar mais de uma configuração de inicialização, selecione-a no painel esquerdo e clique no botão Delete (Apagar) na barra de ferramentas, ou clique com o botão direito e escolha Delete no menu contextual.

Executando o Aplicativo Hello Android

Compreender o básico de como fazer funcionar um aplicativo Android é um processo simples, porém, detalhado. Agora, você está pronto para ver seu trabalho em ação. Você criou uma configuração de inicialização e o Dispositivo Virtual Android; agora, é hora de executar o aplicativo. Finalmente!

Executando a aplicativo no emulador

Executar o aplicativo é simples. Com suas instruções, o ADT inicializa um emulador com a configuração de inicialização padrão, construída anteriormente neste capítulo. Iniciar seu aplicativo é tão simples quanto escolher Run (Executar)⇨Run ou pressionar Ctrl+F11. Ambas ações inicializam o aplicativo em um emulador usando a configuração de inicialização padrão — neste caso, `ExampleConfiguration`. O ADT compila seu aplicativo, e então, o distribui para o emulador.

Se você não criou uma configuração de inicialização, verá a caixa de diálogo Run As (Executar Como), mostrada na Figura 3-13. Escolha Android Application (Aplicativo Android) e uma configuração de inicialização será criada para você.

Figura 3-13: A caixa de diálogo Run As aparece quando uma configuração de inicialização não está configurada.

Se você criou `ExampleConfiguration`, verá o emulador carregando, como mostrado na Figura 3-14.

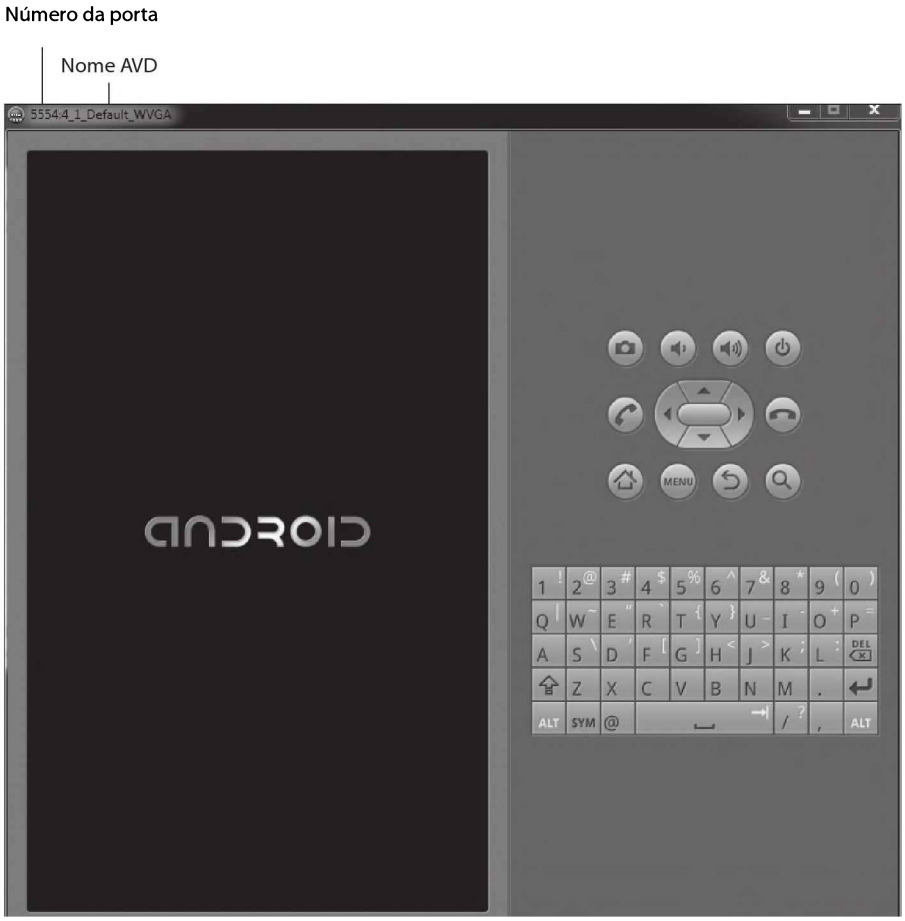


Figura 3-14: O emulador inicial em um estado de carregamento, com o número da porta na qual o emulador está sendo executado e o nome AVD na barra de título da janela.

Socorro! Se seu emulador nunca carregar e ficar preso na(s) tela(s) ANDROID, não há razão para maiores preocupações, camarada. Na primeira vez em que o emulador é executado, pode levar mais de 10 minutos para terminar o carregamento porque você está executando um sistema Linux virtual nele. O emulador passa então pelo processo de boot e inicialização. Quanto mais lento for seu computador, mais lento o emulador será em seu processo de inicialização.

A Figura 3-14 mostra a tela de inicialização do emulador. A barra de título da janela contém o número da porta na qual o emulador está sendo executado em seu computador (`5554`) e o nome AVD (`4_1_Default_WVGA`). O logotipo Android é o mesmo que os usuários do sistema operacional Android padrão veem quando inicializam seus telefones (a menos que um fabricante de dispositivo não tenha instalado suas próprias personalizações da interface de usuário, como no HTC Sense).

A terceira e última tela que você vê é o emulador carregado, mostrado na Figura 3-15.

Economize um tempo valioso deixando o emulador em execução. O emulador não tem que ser carregado sempre que você deseja executar seu aplicativo. Depois que o emulador estiver em execução, você poderá mudar seu código-fonte, e então, executar novamente seu aplicativo. O ADT encontra o emulador em execução e distribui seu aplicativo nele.

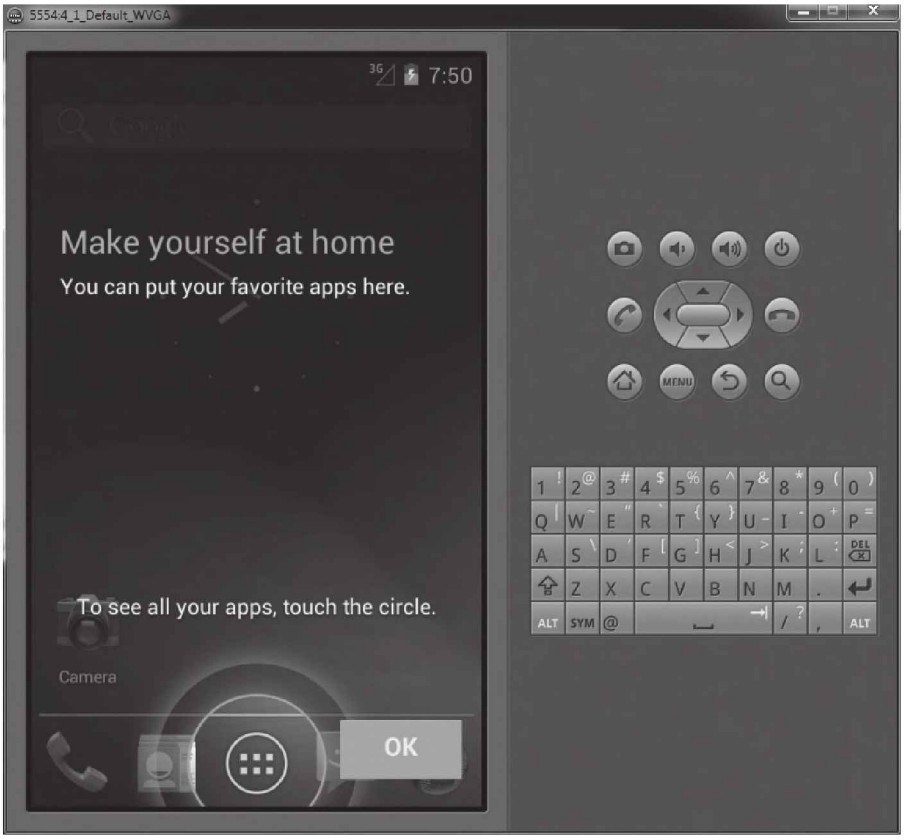


Figura 3-15: O emulador 4_1_Default _WVGA carregado.

Quando o emulador completar sua fase de carregamento, a tela inicial padrão aparecerá bloqueada, como mostrado na Figura 3-16.

Para desbloquear a tela inicial, clique e arraste o ícone cadeado em direção à borda direita da tela. Quando o ícone atingir o limite da tela Android, solte-o. Ao arrastar, o fundo do ícone ficará verde e sua etiqueta mudará para `Unlock`.

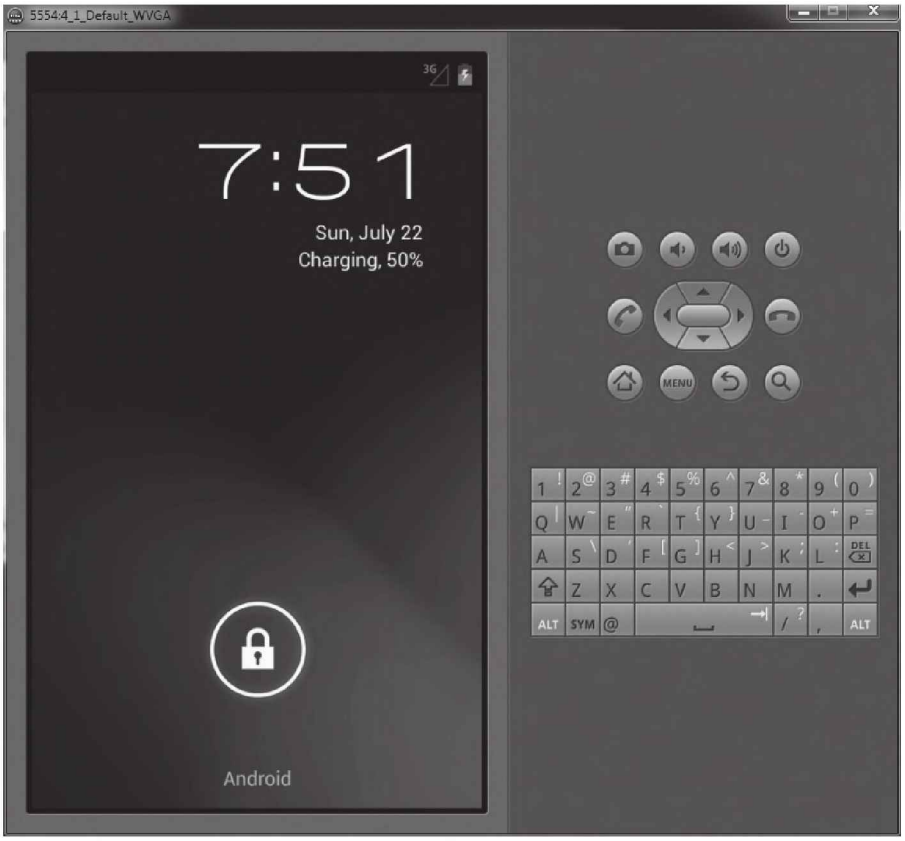


Figura 3-16: Desbloqueando uma tela inicial bloqueada.

Depois que o emulador for desbloqueado, a tela inicial aparecerá, como mostrado na figura 3-17.

Figura 3-17: A tela inicial do emulador.

Logo após isso, o ADT iniciará o aplicativo Hello Android para você. Você verá uma tela vazia com as palavras `Hello world!`, como mostrado na Figura 3-18.

Você acabou de criar e iniciar seu primeiro aplicativo Android.

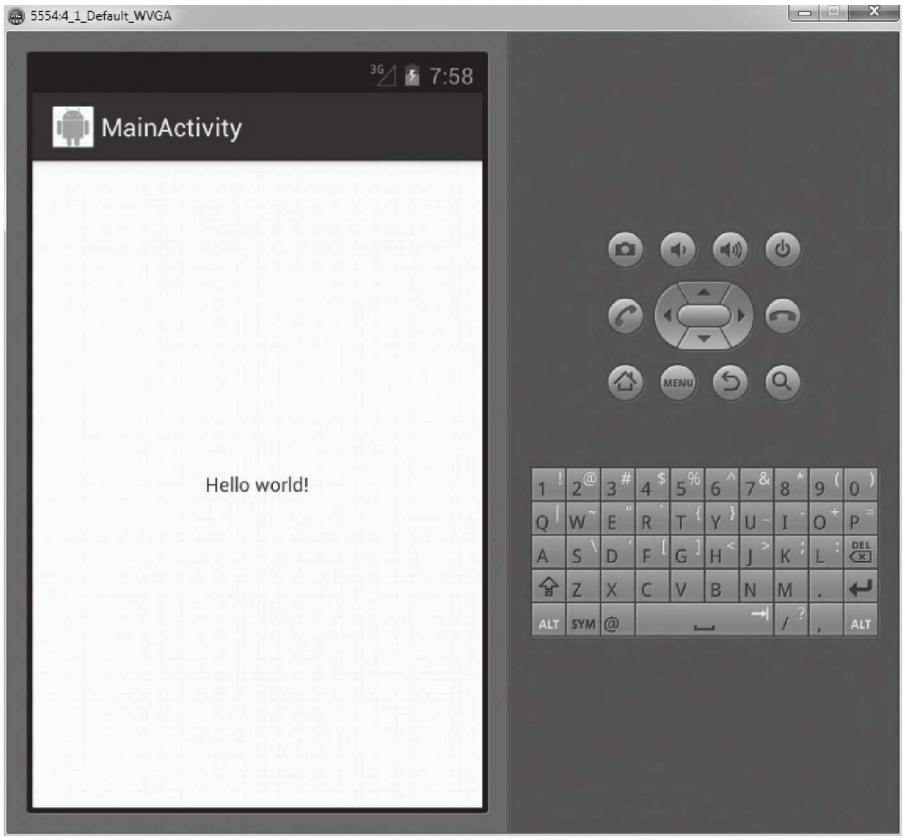


Figura 3-18: O aplicativo Hello Android no emulador.

Se você não desbloquear a tela quando o emulador iniciar, o ADT não poderá iniciar o aplicativo. Se você desbloquear a tela inicial e seu aplicativo não iniciar dentro de cinco a dez segundos, simplesmente execute o aplicativo a partir do Eclipse novamente, escolhendo Run (Executar)⇨Run. O aplicativo será redistribuído para o dispositivo e iniciará a execução.

Você pode ver o status da instalação via exibição Console no Eclipse, como mostrado na Figura 3-19.

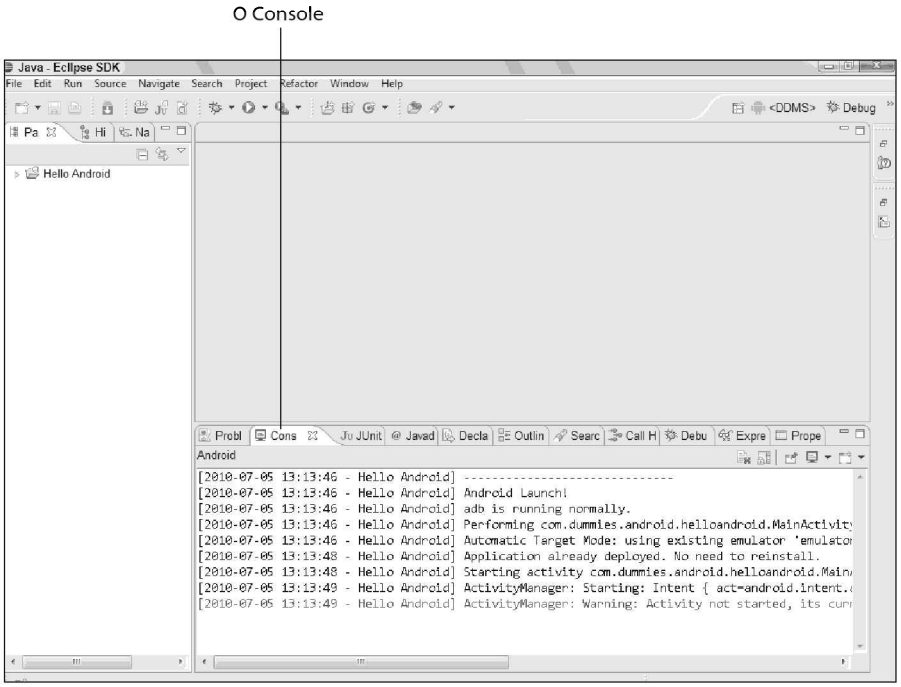


Figura 3-19: A exibição Console no Eclipse permite ver o que está acontecendo internamente enquanto seu aplicativo está sendo distribuído para um dispositivo.

Verificando o status da implementação

Na exibição Console, você pode ver as informações relacionadas ao estado da distribuição de seu aplicativo. Eis o texto completo dessas informações:

```
[2012-07-05 13:13:46 - Hello Android] ------------------------------
[2012-07-05 13:13:46 - Hello Android] Android Launch!
[2012-07-05 13:13:46 - Hello Android] adb is running normally.
[2012-07-05 13:13:46 - Hello Android] Performing com.dummies.android.
              helloandroid.MainActivity activity launch
[2012-07-05 13:13:46 - Hello Android] Automatic Target Mode: using existing
              emulator 'emulator-5554' running compatible AVD '4_1_Default_WVGA'
[2012-07-05 13:13:48 - Hello Android] Application already deployed. No need to
              reinstall.
[2012-07-05 13:13:48 - Hello Android] Starting activity com.dummies.android.
              helloandroid.MainActivity on device
[2012-07-05 13:13:49 - Hello Android] ActivityManager: Starting: Intent {
              act=android.intent.action.MAIN cat=[android.intent.category.
              LAUNCHER] cmp=com.dummies.android.helloandroid/.MainActivity }
[2012-07-05 13:13:49 - Hello Android] ActivityManager: Warning: Activity not
              started, its current task has been brought to the front
```

A exibição Console fornece informações valiosas sobre o estado da distribuição do aplicativo. Ela informa quando está inicializando uma atividade; mostra qual dispositivo o ADT está utilizando; e mostra informações de aviso, como a apresentada na última linha da exibição Console:

```
[2012-07-05 13:13:49 - Hello Android] ActivityManager: Warning:
Activity not
started, its current task has been brought to the front
```

O ADT informa que a atividade — `MainActivity`, neste caso — não foi iniciada porque já estava em execução. Sendo assim, o ADT trouxe a tarefa para o primeiro plano (a tela Android) para você ver.

Compreendendo a Estrutura do Projeto

Você criou seu primeiro aplicativo. Fez isso sem código. É bom que o ADT forneça as ferramentas para inicializar um aplicativo rapidamente, mas isso não o ajudará a criar seu próximo aplicativo de sucesso. O início deste capítulo guiou-o na criação de um aplicativo Android padrão usando o New Android Project Wizard (Assistente para Novo Projeto Android). O resto do capítulo mostrará como usar a estrutura de arquivos que o assistente Android criou para você.

As seguintes seções não devem ser somente folheadas (são vitais!) porque você passará todo seu tempo de desenvolvimento Android navegando essas pastas. Entender o que elas são e como chegaram lá é um aspecto crucial para compreender o desenvolvimento Android.

Navegando as pastas do aplicativo

No Eclipse, o Package Explorer ou o Project Explorer expande o projeto Hello Android, como mostrado na Figura 3-20.

Depois que o projeto Hello Android for expandido, a lista de subpastas incluirá

- `src`
- `gen`
- A versão Android, tal como `Android 4.1`
- `assets`
- `res`

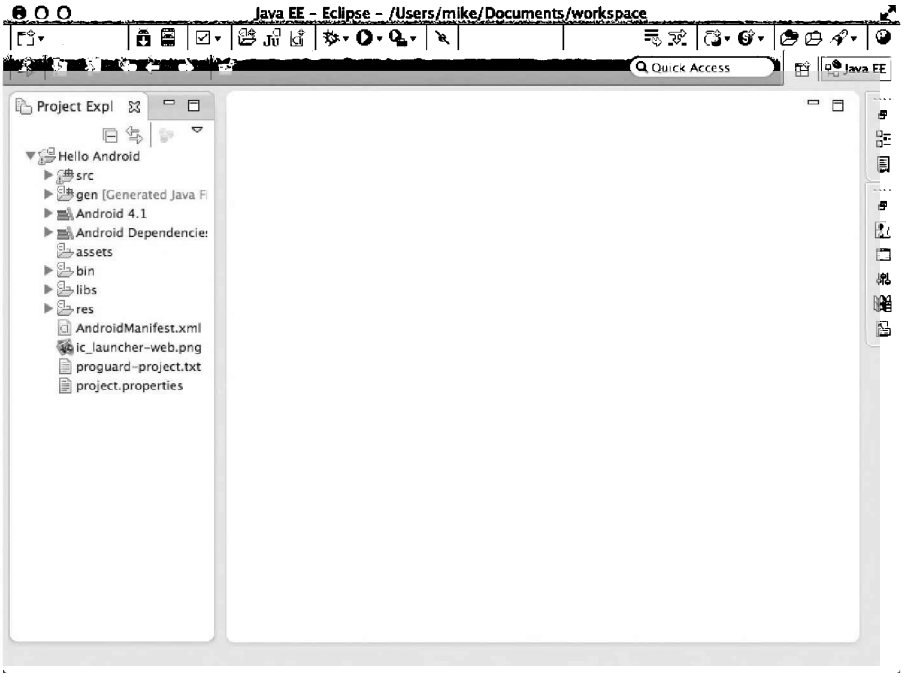


Figura 3-20: O Project Explorer com a estrutura de pastas do projeto Hello Android expandida.

Estas pastas não são as únicas que você pode ter dentro de um projeto Android, mas são as pastas criadas por padrão pelo New Android Project Wizard (Assistente para Novo Projeto Android). As outras pastas incluem `bin`, `libs` e Android Dependencies (Dependências Android).

Dois arquivos importantes no projeto são `AndroidManifest.xml` e `project.properties`. O arquivo `AndroidManifest.xml` ajuda a identificar os componentes que constroem e executam o aplicativo, ao passo que o arquivo `project.properties` ajuda a identificar os valores default das propriedades do projeto Android (tal como a versão Android).

As seguintes seções irão analisar todas as pastas e arquivos.

Pasta-fonte (src)

A pasta-fonte — conhecida como pasta `src` nos projetos Android — inclui seu arquivo de stub `MainActivity.java`, que você criou no New Android Project Wizard (Assistente para Novo Projeto Android) anteriormente neste capítulo. Para examinar o conteúdo da pasta `src`, você deve expandi-la. Siga estas etapas:

1. **Selecione a pasta `src` e clique na pequena seta à esquerda da pasta para expandi-la.**

   Você verá o pacote padrão de seu projeto: `com.dummies.android.helloandroid`.

2. Selecione o pacote padrão e expanda-o.

Esta etapa exibe o arquivo `MainActivity.java` dentro do pacote `com.dummies.android.helloandroid`, como mostrado na Figura 3-21.

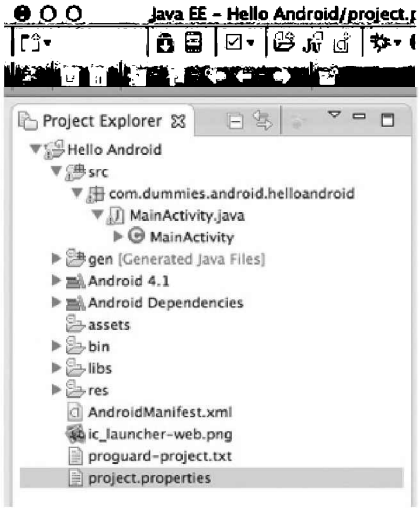


Figura 3-21: A pasta `src` expandida e mostrando o arquivo de stub `MainActivity.java` dentro do pacote Java padrão `com.dummies.android.helloandroid`.

Você não está limitado a um único pacote em seus aplicativos Android. De fato, separar em pacotes as diferentes partes da funcionalidade básica de suas classes Java é considerado uma boa prática. Um exemplo é uma classe cuja responsabilidade é comunicar-se com uma API web via linguagem de marcação extensível (XML). E mais, seu aplicativo pode ter objetos `Customer` que representam um modelo de domínio do cliente, que são recuperados via classes API web. Neste ponto, você pode ter dois pacotes Java extras que contêm as classes Java adicionais:

- `com.dummies.android.helloandroid.models`
- `com.dummies.android.helloandroid.http`

Estes pacotes conteriam seus respectivos componentes Java. `com.dummies.android.helloandroid.models` conteria as classes Java do modelo de domínio e `com.dummies.android.helloandroid.http` conteria as classes Java relacionadas ao HTTP (APIs web). A Figura 3-22 mostra um projeto Android configurado assim.

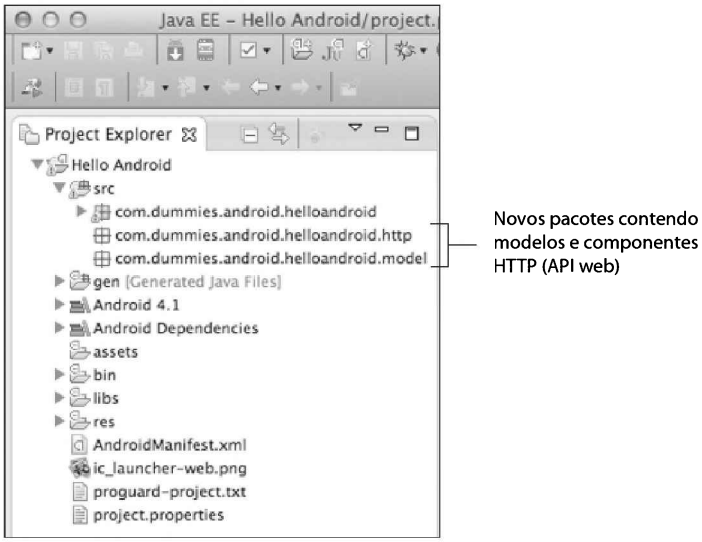


Figura 3-22: Diversos pacotes, sob a pasta `src`, que contêm suas próprias classes Java.

Pasta Target Android Library

A pasta Target Android Library (Biblioteca Android de Destino) não é uma pasta em si, porém segue mais a linha de um item no Eclipse apresentado através do ADT.

Esse item inclui o arquivo `android.jar` que seu aplicativo constrói. A versão desse arquivo foi determinada pelo destino de construção que você escolheu no New Android Project Wizard (Assistente para Novo Projeto Android). Expandir o item Android 4.1 no projeto exibirá o arquivo `android.jar` e o caminho em que ele está instalado, como mostrado na Figura 3-23.

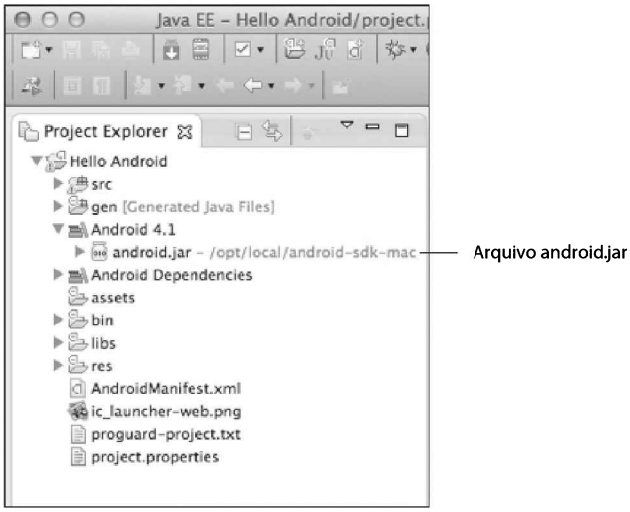


Figura 3-23: O Android versão 4.1 do arquivo `android.jar` com sua localização.

Pasta de componentes (assets)

A pasta `assets` está vazia por padrão. Essa pasta é usada para armazenar os arquivos de componentes brutos.

Um *arquivo de componentes bruto* pode ser um dos muitos componentes que você precisa para seu aplicativo funcionar. Um ótimo exemplo seria um arquivo que contém dados em um formato proprietário para ser consumido pelo dispositivo. O Android tem o Asset Manager (Gerenciador de Componentes), que pode retornar todos os componentes no diretório `assets`. Ao ler um componente, seu aplicativo pode ler os dados no arquivo. Se você criar um aplicativo que tem seu próprio dicionário para as pesquisas de palavras (para o preenchimento automático, talvez), poderá querer embutir o dicionário no projeto, colocando o arquivo de dicionário (geralmente, um XML ou arquivo binário, tal como o banco de dados SQLite) no diretório `assets`.

O Android trata os componentes como uma abordagem bruta para o gerenciamento de recursos. Você não está limitado àquilo que coloca no diretório `assets`. Note, porém, que o trabalho com componentes pode ser um pouco mais chato do que o trabalho com os recursos, pois você precisa trabalhar com fluxos de bytes e convertê-los em objetos que o interessam – áudio, vídeo ou texto, por exemplo.

Os componentes não recebem os IDs (identificadores) de recurso como os recursos no diretório `res`. Você tem que trabalhar com bits, bytes e fluxos manualmente para acessar o conteúdo.

Pasta de recursos (res)

A pasta `res` contém os vários recursos que seu aplicativo pode consumir. Sempre externalize qualquer recurso que seu aplicativo precise consumir. Os exemplos clássicos de tais recursos incluem strings e imagens. Como exemplo, você deve evitar colocar strings dentro de seu código. Ao contrário, crie um recurso de string e referencie esse recurso de dentro do código (você descobrirá como fazer isso posteriormente no livro). Esses tipos de recursos devem ser agrupados no subdiretório `res` mais adequado a eles.

Você também deve fornecer recursos alternativos para as configurações específicas do dispositivo, agrupando-os em diretórios de recursos nomeados especificamente. Em tempo de execução, o Android determina em qual configuração o aplicativo está sendo executado e escolhe o devido recurso (ou pasta de recursos) a partir do qual obterá o que precisa. Você pode querer fornecer um layout da interface de usuário (IU) diferente dependendo do tamanho da tela ou diferentes strings dependendo da definição do idioma, por exemplo.

Depois de externalizar seus recursos, você poderá acessá-los no código via IDs do recurso que são gerados pelo ADT na classe R (veja "A misteriosa pasta gen", posteriormente neste capítulo).

Você deve colocar cada recurso em um subdiretório específico do diretório `res` de seu projeto. A Tabela 3-1 lista os subdiretórios que são os tipos mais comuns de pastas de recursos sob o diretório-pai `res`.

Tabela 3-1 Subdiretórios suportados do diretório `res` Diretório

Diretório	*Tipo de recurso*
`anim/`	Arquivos XML que definem as animações.
`color/`	Arquivos XML que definem uma lista de cores.
`drawable/`	Arquivos bitmap (`.png, .9.png, .jpg, .gif`) ou arquivos XML que são compilados nos seguintes recursos de desenho.
`drawable-xhdpi/`	Desenhos para as telas com resolução extra-alta. O qualificador `xhdpi` significa telas com densidade extra-alta. É similar à pasta de recursos `drawable/`, exceto que todos os arquivos bitmap ou XML armazenados aqui são compilados nos recursos de desenho com resolução extra-alta.
`drawable-hdpi/`	Desenhos para as telas com uma resolução alta. O qualificador `hdpi` significa telas com densidade alta. É similar à pasta de recursos `drawable/`, exceto que todos os arquivos bitmap ou XML armazenados aqui são compilados nos recursos de desenho com resolução alta.
`drawable-ldpi/`	Desenhos para as telas com uma resolução baixa. O qualificador `ldpi` significa telas com densidade baixa. É similar à pasta de recursos `drawable/`, exceto que todos os arquivos bitmap ou XML armazenados aqui são compilados nos recursos de desenho com resolução baixa.
`drawable-mdpi/`	Desenhos para as telas com uma resolução média. O qualificador `mdpi` significa telas com densidade média. É similar à pasta de recursos `drawable/`, exceto que todos os arquivos bitmap ou XML armazenados aqui são compilados nos recursos de desenho com resolução média.
`layout/`	Arquivos XML que definem um layout de interface de usuário.
`menu/`	Arquivos XML que representam os menus do aplicativo.
`raw/`	Arquivos arbitrários para serem salvos em sua forma bruta. Os arquivos neste diretório não são compactados pelo sistema.
`values/`	Os arquivos XML que contêm valores simples, tais como strings, inteiros e cores. Enquanto os arquivos de recurso XML nas outras pastas `res/` definem um único recurso com base nos nomes de arquivo XML, os arquivos no diretório `values/` definem diversos recursos para vários usos. Você deve seguir algumas convenções de nomenclatura de arquivo descritas na próxima seção complementar "Nomeando recursos no diretório values", para os recursos que poderá criar neste diretório.

Nomeando recursos no diretório values

Você deve seguir algumas convenções de nomenclatura de arquivos para os recursos que pode criar no diretório `values`:

- `arrays.xml` para os arrays de recursos (armazenando como itens, tais como strings ou inteiros, juntos).
- `colors.xml` para os recursos que definem os valores das cores; acessados via classe `R.color`.
- `dimens.xml` para os recursos que definem os valores de dimensão (`20px` é igual a 20 pixels, por exemplo), acessados via classe `R.dimen`.
- `strings.xml` para os valores de texto, acessados via classe `R.string`.
- `styles.xml` para os recursos que representam os estilos; acessados via classe `R.style`. Um estilo é parecido com uma folha de estilo em cascata no HTML. Você pode definir muitos estilos e fazer com que herdem uns dos outros.

Nunca salve os arquivos de recurso diretamente no diretório `res`. Se o fizer, ocorrerá um erro do compilador.

Os recursos salvos nas pastas de recursos listadas na Tabela 3-1 são conhecidos como recursos *padrão* — eles definem o design e o layout padrão de seu aplicativo Android. Porém, diferentes tipos de dispositivos acionados pelo Android podem precisar de diferentes recursos. Se você tiver um dispositivo com uma tela maior que o normal, por exemplo, forneça recursos de layout alternativos para levar em conta essa diferença.

Uma análise do poderoso mecanismo de recursos do Android poderia requerer um capítulo próprio, mas este livro cobre o básico para você trabalhar. O mecanismo de recursos pode ajudar com a internacionalização (ativando seu aplicativo para diferentes idiomas e países), tamanho do dispositivo e densidade, e até recursos para o modo em que o telefone pode estar. Para mergulhar no oceano de recursos, reveja a seção "Providing Resources" (Fornecendo Recursos) do Guia do Dispositivo na documentação Android, em `http://d.android.com/guide/topics/resources/providing-resources.html` (conteúdo em inglês).

Pastas Bin, Libs e de Bibliotecas Referenciadas

As pastas de bibliotecas não são mostradas em seu aplicativo Hello Android, mas você deve conhecer algumas pastas extras, sendo uma delas o diretório `libs/`. Ele pode conter bibliotecas proprietárias de terceiros que executam uma função para você. Um exemplo é a jTwitter, uma biblioteca Java de terceiros para a API do Twitter. Se você fosse usar a jTwitter em seu aplicativo Android, colocaria a biblioteca `jtwitter.jar` no diretório `libs`.

Depois que a biblioteca for colocada no diretório `libs`, adicione-a ao seu caminho de construção Java — o caminho de classes usado para construir um projeto Java. Se seu projeto depender de outra biblioteca de terceiros ou privada, o Eclipse deverá saber onde encontrar tal biblioteca. Definir o caminho de construção via Eclipse faz exatamente isso. Supondo que você adicionou `jtwitter.jar` ao seu diretório `libs`, você poderá adicioná-la facilmente ao seu caminho de construção clicando com o botão direito no arquivo `jtwitter.jar` e escolhendo Build Path (Caminho de Construção)⇨Add to Build Path (Adicionar a Caminho de Construção) no menu contextual. Após isso, ela será listada sob a pasta libs no Package Explorer, como mostrado na Figura 3-24.

Você pode descobrir mais sobre a jTwitter em `www.winterwell.com/software/jtwitter.php` (conteúdo em inglês).

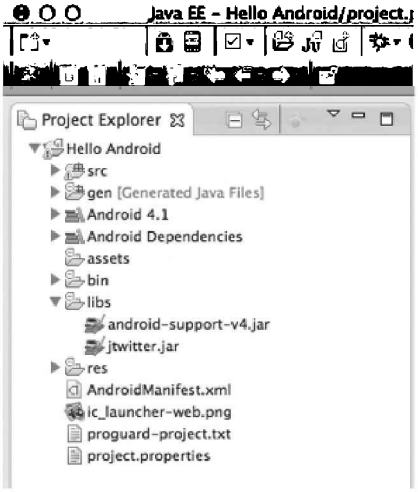


Figura 3-24: A pasta `libs` com `jtwitter.jar`.

Não usamos o diretório `libs` neste livro, embora os desenvolvedores usem comumente bibliotecas de terceiros nos aplicativos Android.

A misteriosa pasta gen

Ah, finalmente você irá testemunhar a mágica que é a pasta `gen`. Quando você cria um aplicativo Android, antes da primeira compilação, a pasta `gen` não existe. Na primeira compilação, o ADT gera a pasta `gen` e seu conteúdo.

A pasta `gen` contém os arquivos Java gerados pelo ADT. O ADT cria um arquivo `R.java` (falaremos mais sobre esse assunto em breve). A pasta `gen` contém os itens gerados a partir do diretório `res`. Sem a devida compreensão do que é a pasta `res` e o que ela contém, você não saberá para que serve a pasta `gen`. Mas, como você já é um especialista na pasta `res`, poderá mergulhar na pasta `gen`.

Quando você escreve código Java no Android, chega a um ponto em que é necessário referenciar os itens na pasta `res`. Você faz isso usando a classe R. O arquivo `R.java` é um índice de todos os recursos definidos em sua pasta `res`. Você usa essa classe como um atalho para referenciar os recursos incluídos em seu projeto. Isto é particularmente útil com os recursos de code-completion do Eclipse porque você pode identificar rapidamente o devido recurso via code-completion.

Expanda a pasta `gen` no projeto Hello Android e o nome do pacote contido na pasta `gen`. Agora, abra o arquivo `R.java` clicando-o duas vezes. Você verá uma classe Java que contém classes Java aninhadas. Essas classes Java aninhadas têm os mesmos nomes de algumas pastas `res` definidas na seção `res` anterior. Sob cada uma dessas subclasses, você poderá ver os membros que têm os mesmos nomes dos recursos em suas respectivas pastas `res` (excluindo suas extensões de arquivo). O arquivo `R.java` do projeto Hello Android deve parecer com o seguinte código:

```
/* ARQUIVO AUTOGERADO. NÃO MODIFIQUE.
 *
 * Esta classe foi gerada automaticamente pela
 * ferramenta aapt a partir dos dados de recurso encontrados. Ela
 * não deve ser modificada manualmente.
 */

package com.dummies.android.helloandroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Espere — o que é `0x`? A ferramenta ADT gera este código para você, para que não tenha que se preocupar com o que está acontecendo internamente. Quando você adiciona recursos e o projeto é reconstruído, o ADT gera de novo o arquivo `R.java`. Esse arquivo recém-gerado contém os membros que referenciam seus recursos recém-adicionados.

Você nunca deve editar o arquivo `R.java` manualmente. Se editar, seu aplicativo poderá não compilar, e então, estará perdido. Se você editar sem querer o arquivo `R.java` e não conseguir desfazer suas alterações, poderá apagar a pasta `gen` e construir seu projeto. Neste ponto, o ADT irá gerar de novo o arquivo `R.java` para você.

Exibindo o arquivo manifest do aplicativo

Você controla tudo que possui e precisa através de listas, não controla? Bem, é exatamente isso que o arquivo manifest do Android faz. Ele controla tudo que seu aplicativo precisa, solicita e tem que usar para ser executado.

O arquivo manifest do Android é armazenado na raiz de seu projeto e é nomeado como `AndroidManifest.xml`. Todo aplicativo deve ter um arquivo manifest do Android no diretório-raiz.

O arquivo manifest do aplicativo fornece todas as informações essenciais para o sistema Android — informações que ele deve ter antes de poder executar qualquer código de seu aplicativo. O arquivo manifest do aplicativo também fornece

- O nome de seu pacote Java para o aplicativo, que é o identificador único do seu aplicativo no sistema Android, assim como na Google Play Store.
- Os componentes do aplicativo, tais como as atividades e os serviços em segundo plano.
- A declaração das permissões que seu aplicativo requer para ser executado.
- O nível mínimo da API Android que o aplicativo requer.

O arquivo manifest do Android declara a versão de seu aplicativo. Você *deve* versionar o seu aplicativo. Como você versiona o seu aplicativo é parecido com o modo como o SO do Android é versionado. É importante determinar a estratégia de versionamento do seu aplicativo no início do processo de desenvolvimento, inclusive considerando as futuras releases de seu aplicativo. É requerido que cada aplicativo tenha um código de versão e um nome de versão.

Código de versão

O *código de versão* é um valor inteiro que representa a versão do código-fonte do aplicativo relativo às outras versões anteriores. Esse valor é usado para ajudar os outros aplicativos a determinarem sua compatibilidade com seu aplicativo. E mais, a Google Play Store usa-o como uma base para identificar o aplicativo internamente e para lidar com as atualizações.

Você pode definir o código de versão como qualquer valor inteiro desejado, mas deve assegurar que cada release sucessiva tenha um código de versão maior que a anterior.

Em geral, na primeira release, você define o código de versão para `1`. Então, aumenta uniformemente o valor a cada nova release, seja ela uma alteração significativa ou somente alguns ajustes. Isto significa que o código de versão não tem uma grande semelhança com a versão de release do aplicativo, visível para o usuário — que é o nome da versão (veja a próxima seção). O código de versão geralmente não é exibido para os usuários nos aplicativos.

Fazer o upgrade do código de seu aplicativo e publicá-lo sem aumentar o código de versão faz com que bases diferentes de código do seu aplicativo sejam publicadas sob a mesma versão. Considere um cenário no qual você disponibiliza seu aplicativo com o código de versão 1. Esta é a sua primeira release. Um usuário instala seu aplicativo via Google Play Store e nota um erro nele, comunicando-o a você. Você corrige o erro no código, recompila e lança a nova base de código sem atualizar o código de versão no arquivo manifest do Android. Nesse ponto, a Google Play Store não sabe que algo mudou porque está examinando seu código de versão através do manifest do aplicativo. Se o código de versão tivesse mudado para um valor superior a `1`, tal como `2`, a Google Play Store reconheceria que uma atualização foi feita e informaria aos usuários que instalaram o aplicativo com o código de versão 1, que uma atualização está disponível. Se você não atualizar o código de versão, os usuários nunca obterão a atualização para sua base de código e executarão um aplicativo com erros. Ninguém gosta disso!

Nome de versão

O *nome de versão* é um valor de tipo string que representa a versão de lançamento do código-fonte do aplicativo, como deveria ser mostrado para os usuários. O valor é uma string que pode ser qualquer coisa, mas geralmente segue uma nomenclatura padronizada de releases que descreve a versão do aplicativo:

```
<maior>.<menor>.<ponto>
```

Um exemplo desta nomenclatura de releases é 2.1.4 ou, sem o valor `<ponto>` (4, neste caso), 2.1.

O sistema Android não usa esse valor para nenhuma finalidade, exceto permitir que os aplicativos o exibam para os usuários.

O nome da versão pode ser qualquer outro tipo absoluto ou relativo de identificação. O aplicativo Foursquare, por exemplo, usa um esquema de nome de versão que corresponde à data. Um exemplo do nome da versão deste aplicativo é `2012.05.02`, que representa claramente uma data. O nome da versão é deixado para você. Você deve planejar de antemão e assegurar que a sua estratégia de versão faça sentido para você e seus usuários.

Permissões

Suponha que seu aplicativo precise acessar a Internet para recuperar alguns dados. O Android limita o acesso à Internet por padrão. Para seu aplicativo ter acesso à Internet, você precisa solicitar a permissão.

No arquivo manifest do aplicativo, você deve definir de quais permissões seu aplicativo precisa para operar. A Tabela 3-2 lista algumas permissões comumente solicitadas.

Tabela 3-2 Permissões de aplicativo comumente solicitadas

Permissão	*O que significa*
Internet	O aplicativo precisa acessar a Internet.
Gravar em Armazenamento Externo	O aplicativo precisa gravar dados no cartão Digital Seguro (cartão SD).
Câmera	O aplicativo precisa acessar a câmera.
Acessar Localização	O aplicativo precisa de acesso ao sistema de posicionamento global (GPS).
Ler Estado do Telefone	O aplicativo precisa acessar o estado do telefone (tal como o toque).

Exibindo o arquivo project.properties

O arquivo `project.properties` é usado em conjunto com o ADT e o Eclipse. Ele contém as definições do projeto, tais como o destino de construção. Esse arquivo é essencial para o projeto, portanto, não o perca.

O arquivo `project.properties` nunca deve ser editado manualmente. Para editar o conteúdo do arquivo, use o editor no Eclipse: clique com o botão direito no nome do projeto no Package Explorer e escolha Properties (Propriedades) no menu contextual. Esta ação abrirá o editor Properties, mostrado na Figura 3-25.

Esse editor permite que você altere várias propriedades do projeto selecionando qualquer opção à esquerda. Você pode selecionar a propriedade Android e mudar o SDK do Android, por exemplo.

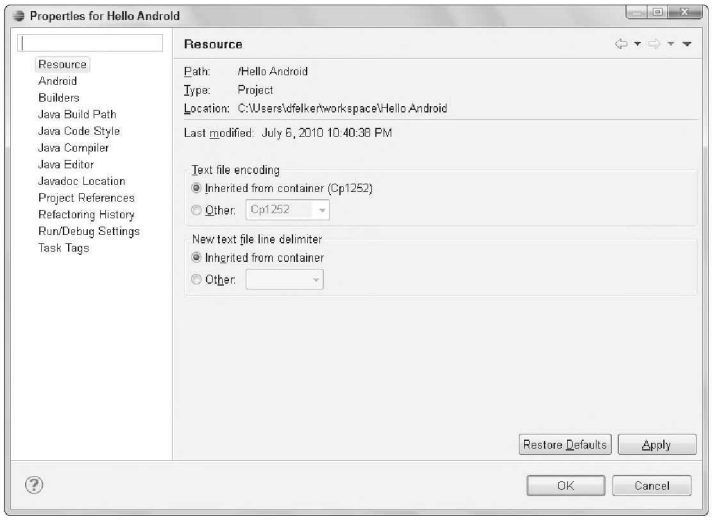


Figura 3-25: O editor Properties no Eclipse.

Fechando Seu Projeto

Após ter criado seu aplicativo, você precisará fechar qualquer arquivo aberto no Eclipse antes de iniciar seu próximo trabalho. Você pode fazer isso fechando cada arquivo individualmente ou clicando com o botão direito nos arquivos e escolhendo Close All (Fechar Todos) no menu de atalho.

Após ter fechado todos os arquivos, você precisará fechar o projeto em si. No Eclipse, no Package Explorer, clique com o botão direito no projeto e escolha Close Project (Fechar Projeto). Fechando o projeto, você está informando ao Eclipse que não precisa mais trabalhar nele. Isto libera os recursos que o Eclipse usa para controlar o estado do projeto, e portanto, aumenta a velocidade de seu aplicativo.

Capítulo 4

Projetando a Interface do Usuário

Neste Capítulo

- Configurando o aplicativo Silent Mode Toggle
- Projetando o layout
- Desenvolvendo a interface do usuário
- Adicionando uma imagem e um componente botão
- Criando um ícone de inicialização
- Visualizando seu trabalho

No Capítulo 3, você descobriu o que é o Android e como construir seu primeiro aplicativo. Este capítulo ajuda-o a entrar na parte divertida: construir um aplicativo real e publicá-lo na Google Play Store.

O aplicativo construído neste capítulo permite ao usuário ativar e desativar o modo campainha no telefone simplesmente pressionando um botão. Esse aplicativo parece simples, mas resolve um problema real.

Imagine que você está no trabalho, e prestes a entrar em uma reunião. Você pode diminuir o volume de seu telefone para silenciá-lo, e então, ir à reunião (você não desejaria ser a pessoa cujo telefone *sempre* toca durante uma reunião, não é?). O problema é que você gosta da campainha alta, mas não alta demais. Então, você ajusta a campainha para a segunda mais alta. Quando você sai da reunião, lembra de restaurar o volume da campainha, mas sempre tem que mover até o final para o volume máximo, e depois diminuir um ponto para assegurar que terá o volume correto. Embora não seja um grande sacrifício, é chato de realizar sempre que você precisa silenciar a campainha de seu telefone.

Se um aplicativo permitisse o desligamento da campainha com um clique de botão, então, quando você saísse da reunião, clicaria neste mesmo botão para restaurar a campainha ao estado anterior, e nunca teria que reajustar de novo. Este é o aplicativo que você irá construir.

Criando o Aplicativo Silent Mode Toggle

Crie o novo aplicativo escolhendo File (Arquivo)⇨New Project (Novo Projeto). Escolha Android Project (Projeto Android) na lista, e então, clique no botão Next (Próximo). Use a Tabela 4-1 para suas definições do projeto.

Tabela 4-1 Definições do projeto para Silent Mode Toggle

Definição	*Valor*
Nome do Aplicativo	Silent Mode Toggle
Nome do projeto	Silent Mode Toggle
Conteúdo	Deixar o padrão selecionado (criar um novo projeto no espaço de trabalho)
Destino da construção	Android 4.1
Nome do pacote	`com.dummies.android.silentmodetoggle`
Criar atividade	`MainActivity`
Versão SDK Mínima	8 (2.2)

Clique no botão Finish (Terminar). Agora, você deve ter o aplicativo Silent Mode Toggle em seu Package Explorer, como mostrado na Figura 4-1.

Se você vir um erro parecido com este — "The project cannot be built until build path errors are resolved" (o projeto não pode ser construído até que os erros do caminho de construção sejam resolvidos) — poderá resolvê-lo clicando com o botão direito no projeto e escolhendo Android Tools (Ferramentas Android)⇨Fix Project Properties (Corrigir Propriedades do Projeto). Isto irá realinhar seu projeto com o espaço de trabalho IDE.

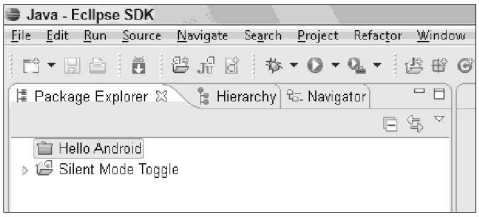


Figura 4-1: O aplicativo Silent Mode Toggle no Eclipse.

Note como você definiu o destino da construção para 4.1 e o SDK mínimo para 2.2 (API nível 8). O que você fez foi informar ao Android que seu código pode ser executado em qualquer dispositivo que execute, pelo menos, o código da versão 8 (Android 2.2). Se você mudar isto para o código da versão 16, estaria dizendo que seu aplicativo somente poderá ser executado em qualquer dispositivo que execute a versão 16 (Android 4.1) ou superior. Ao criar um novo aplicativo, você deve decidir se deseja que ele seja executado nas versões mais antigas.

Se você precisar de um lembrete sobre como criar um novo aplicativo Android no Eclipse, veja o Capítulo 3.

Layout do Aplicativo

Quando você tiver o aplicativo Silent Mode Toggle criado dentro do Eclipse, será hora de projetar a *interface de usuário* do aplicativo, a parte de um aplicativo com a qual os usuários interagem. Essa área de seu aplicativo deve ser tão elegante quanto possível.

Seu aplicativo terá um único botão centralizado na tela para acionar o modo silencioso. Logo acima dele, uma imagem fornecerá o retorno visual para permitir que o usuário saiba se o telefone está no modo silencioso ou no modo de campainha normal. As Figuras 4-2 e 4-3 mostram como ficará o aplicativo terminado.

Figura 4-2: O aplicativo Silent Mode Toggle no modo de campainha normal.

Figura 4-3: O aplicativo Silent Mode Toggle no modo silencioso, sem campainha.

Usando o arquivo de layout XML

Todos os arquivos de layout para um aplicativo são armazenados no diretório `res/layouts` do projeto Android no Eclipse. Quando você cria o aplicativo Silent Mode Toggle, as Ferramentas de Desenvolvimento Android (ADT) criam um arquivo chamado `activity_main.xml` no diretório `res/layouts`. Esse arquivo de layout padrão é aquele que o ADT fornece quando você cria um novo aplicativo.

Clique duas vezes no arquivo, clique na guia `activity_main.xml` na parte inferior da tela e veja um XML na janela do editor Eclipse, como mostrado na Figura 4-4.

A Figura 4-4 mostra um layout simples, no qual um valor de texto está no meio da tela. Seu código deverá ficar assim:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        android:text="@string/hello_world"
        tools:context=".MainActivity" />

</RelativeLayout>
```

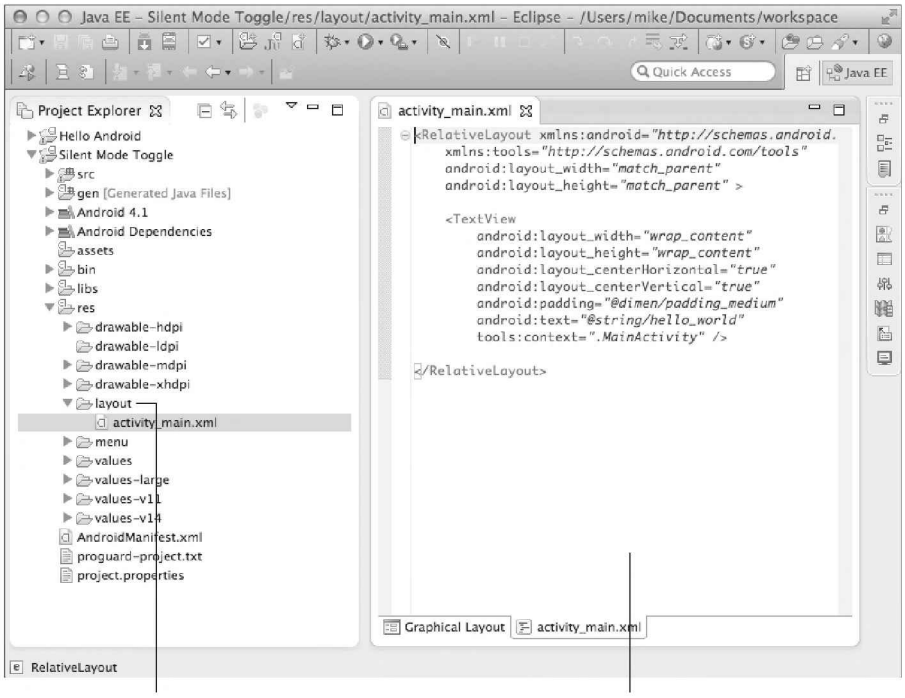


Figura 4-4: O arquivo de layout `activity_main.xml` aberto no Eclipse.

Os layouts são armazenados aqui

O arquivo activity_main.xml

Este arquivo XML define exatamente como deve ser a exibição. As seguintes seções detalharão esse arquivo, elemento por elemento.

Declaração XML padrão

O primeiro elemento no arquivo XML fornece a declaração XML padrão, permitindo que os editores de texto, tais como o Eclipse, e as plataformas, tais como o Android, saibam qual é o tipo de arquivo:

```
<?xml version="1.0" encoding="utf-8"?>
```

Tipo de layout

O próximo elemento no arquivo XML define o tipo de layout. Neste caso, você está trabalhando com `RelativeLayout`, onde os filhos podem ser organizados em relação uns aos outros. `RelativeLayout` é um contêiner para os outros itens que aparecem na tela:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

A tag `</RelativeLayout>` de fechamento não está aparecendo porque ela é um contêiner para outros itens. A tag de fechamento é inserida depois de todos os itens de exibição terem sido adicionados ao contêiner.

Veja a Tabela 4-2, posteriormente neste capítulo, para obter mais informações sobre os tipos de layout — `RelativeLayout` ou outro.

Exibições

`RelativeLayout` pode conter *exibições (view)*, que são os blocos de construção básicos dos componentes de interface do usuário. O seguinte código mostra `TextView`, que é responsável por exibir o texto na tela:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:padding="@dimen/padding_medium"
    android:text="@string/hello_world"
    tools:context=".MainActivity" />
```

Uma exibição ocupa um espaço retangular na tela e é responsável pelos gráficos e tratamento de eventos. Todos os itens que podem aparecer na tela do dispositivo são exibições. A classe `View` é a superclasse da qual todos os itens herdam no Android.

No final do arquivo XML, você tem a tag de fechamento para `RelativeLayout`. Esta linha fecha o contêiner:

```
</RelativeLayout>
```

A seguinte seção descreve a floresta preenchida com diferentes tipos de layouts.

Usando as ferramentas de layout do SDK do Android

Quando você cria uma interface de usuário, algumas vezes, tem que montar o layout dos componentes, posicionados uns em relação aos outros, em uma tabela ou, sob certas circunstâncias, até usando posicionamento absoluto. Felizmente, os gênios da Engenharia no Google, que criaram o Android, pensaram em tudo isso e forneceram as ferramentas necessárias para criar esses tipos de layout. A Tabela 4-2 apresenta rapidamente os tipos comuns de layouts disponíveis no Kit de Desenvolvimento de Software (SDK) do Android.

Tabela 4-2	Layouts do SDK do Android
Layout	***O que faz***
`LinearLayout`	Organiza seus filhos em uma única linha ou coluna.
`RelativeLayout`	Permite que as posições dos filhos sejam descritas em relação entre si ou com o pai.
`FrameLayout`	Projetado para bloquear uma área da tela para exibir um único item. Pode-se adicionar múltiplos filhos a um FrameLayout, mas todos eles estão pregados na área superior esquerda da tela. Os filhos são criados em uma fila, com o filho mais recente adicionado ao topo da fila. Este layout é comumente usado como um modo de fazer a disposição das exibições utilizando posição absoluta.
`GridLayout`	Organiza seus filhos em uma grade.

Existem outros tipos diferentes de ferramentas de layout, tais como `TabHost` para criar guias e `Sliding Drawer` para os movimentos do dedo que ocultam e mostram as exibições. Os programadores tendem a usar essas ferramentas de layout em situações especiais. Os itens na Tabela 4-2 descrevem os layouts mais usados.

Usando o designer visual

Uma boa notícia: O Eclipse tem um designer visual. A má notícia: O designer tem possibilidades limitadas (como todos os designers visuais).

Abrindo o designer visual

Para exibir o designer visual, com o arquivo `activity_main.xml` aberto no editor Eclipse, clique no botão Graphical Layout (Layout Gráfico), como mostrado na Figura 4-5.

Figura 4-5: O botão Graphical Layout, que mostra o designer visual.

O designer visual agora está na tela, como mostrado na Figura 4-6. Nela, você pode arrastar e soltar os itens a partir das caixas de ferramenta Layouts ou Views (Exibições).

Examinando as propriedades de uma exibição

Usando o designer visual, você pode exibir as propriedades de uma determinada exibição simplesmente clicando nela. Muito provavelmente sua janela Properties (Propriedades) estará oculta. Para ver as propriedades, siga estas etapas:

1. **Escolha Window (Janela)⇨Show View (Mostrar Exibição)⇨Other (Outra).**

2. **Expanda General (Geral) e escolha Properties (Propriedades).**

   A janela Properties será aberta no Eclipse, como mostrado na Figura 4-7.

3. **Selecione a exibição "Hello world" no designer visual.**

   A exibição tem uma borda azul e as propriedades aparecem na janela Properties abaixo dela.

4. **Percorra a lista de propriedades para determinar quais elementos podem ser alterados na exibição.**

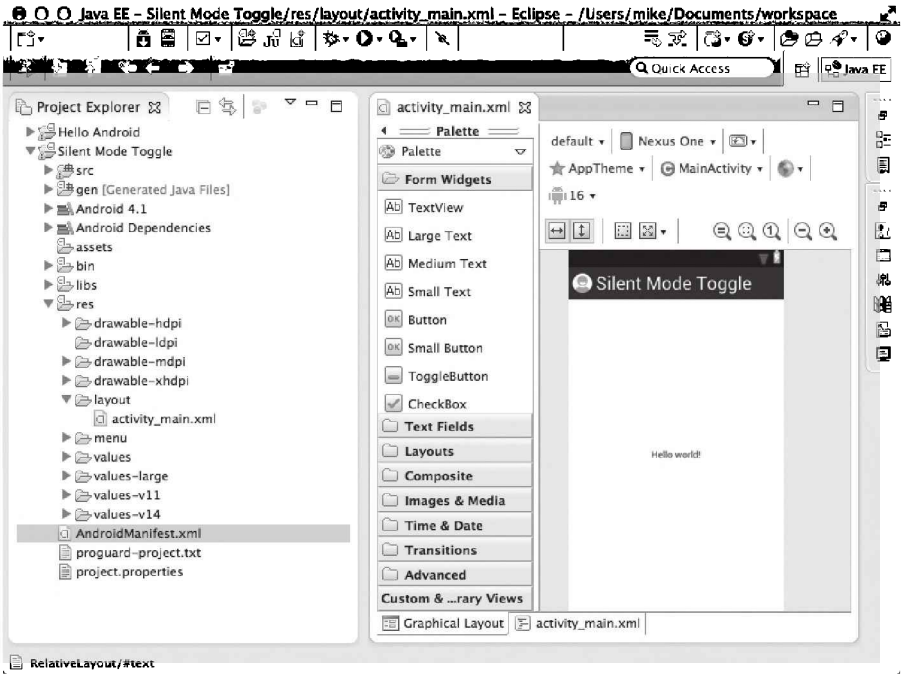


Figura 4-6: O designer visual.

Se você não estiver certo sobre quais propriedades uma exibição tem, abra o designer visual, clique na guia Properties e examine rapidamente a exibição Properties para ver o que ela tem a oferecer. Se a guia Properties não estiver visível, ative-a escolhendo Window (Janela) ->Show View (Mostrar Exibição)⇨Other (Outra)⇨General (Geral)⇨Properties (Propriedades).

As propriedades disponíveis de uma exibição podem mudar dependendo de seu layout-pai. Por exemplo, um `TextView` dentro de um `LinearLayout` tem um conjunto diferente de propriedades (para o layout) em relação a quando está dentro de um `RelativeLayout`.

O designer visual funciona bem para as situações simples nas quais o conteúdo é estático por natureza. Mas, o que acontece quando você precisa desenhar itens na tela dinamicamente com base na entrada do usuário? O designer não pode ajudá-lo nesta situação — ele é mais adequado para uma *situação de conteúdo estático*, onde você cria seu layout uma vez e não o atualiza dinamicamente. O texto de `TextView` ou as imagens podem mudar, mas o layout real das exibições dentro do layout não muda.

Figura 4-7: Um item selecionado no designer visual, com algumas propriedades listadas na janela Properties.

Desenvolvendo a Interface do Usuário

Tudo bem, é hora de começar a desenvolver a interface do usuário. Primeiro, certifique-se de que você esteja na exibição XML de seu layout, clicando na guia `activity_mail.xml`. Quando estiver na exibição XML, apague o XML e substitua-o pelo seguinte. Agora, seu layout deverá ficar assim:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

</LinearLayout>
```

Exibindo os atributos do layout XML

Antes de continuar, você precisa entender os atributos do layout XML do Android com o qual está trabalhando. Veja a Tabela 4-3.

Tabela 4-3	Layouts do SDK do Android
Layout	***O que faz***
`xmlns:android="..."`	Define o namespace XML que você usa para referenciar parte do SDK do Android.
`android:orientation="vertical"`	Informa ao Android que esta exibição terá um layout vertical, tal como o formato de retrato na impressão.
`android:layout_width="match_parent"`	Informa à exibição que ela deve preencher tanto espaço horizontal quanto possível, para deixar sua própria largura igual à de seu pai.
`android:layout_height="match_parent"`	Informa à exibição que ela deve preencher tanto para deixar sua própria altura igual à de seu pai.

Neste ponto, você definiu o layout para preencher a tela inteira definindo a largura e a altura para "`match_parent`".

Trabalhando com as exibições

Como afirmado anteriormente neste capítulo, as exibições no Android são os blocos de construção básicos dos componentes da interface de usuário. Sempre que você implementa um componente da interface, tal como `Layout` ou `TextView` no sistema Android, está usando uma exibição. Quando você trabalha com as exibições no Java, tem que fazer a conversão para o seu devido tipo para conseguir trabalhar com elas.

Definindo os valores layout_width e layout_height

Antes de uma exibição poder ser apresentada na tela, algumas definições devem ser configuradas na exibição para que o Android saiba fazer o layout da exibição na tela. Os atributos requeridos, `layout_width` e `layout_height`, são conhecidos como `LayoutParams` no SDK do Android.

O atributo `layout_width` especifica a largura de uma exibição e o atributo `layout_height` especifica a sua altura.

Definindo os valores match_parent e wrap_content

Os atributos `layout_width` e `layout_height` podem receber qualquer valor de pixel ou valor de pixel independente da densidade para especificar suas dimensões respectivas. Contudo, dois dos valores mais comuns para `layout_width` e `layout_height` são as constantes `match_parent` e `wrap_content`.

O valor `match_parent` informa ao sistema Android para preencher quanto espaço for possível na tela com base no espaço disponível do layout-pai. O valor `wrap_content` informa ao sistema Android para ocupar apenas o espaço necessário para mostrar a exibição. Quando o conteúdo da exibição cresce, como aconteceria a um `TextView`, o espaço ocupado pela exibição cresce, de modo semelhante à propriedade `Autosize` no desenvolvimento de formulários do Windows.

Se você estiver usando um layout estático, esses dois atributos deverão ser definidos no layout XML. Se você estiver criando as exibições dinamicamente via código, os parâmetros do layout deverão ser definidos via código Java. De qualquer modo, você não pode ficar sem eles. Para descobrir mais sobre a criação dinâmica das exibições, vejas as amostras da API que vêm com o SDK do Android.

Se você esquecer de fornecer valores para `layout_width` ou `layout_height`, seu aplicativo Android irá paralisar ao apresentar a exibição. Felizmente, você descobrirá rapidamente quando testar seu aplicativo.

Antes do Android 2.2, `match_parent` era chamado de `fill_parent` (embora seu significado permaneça o mesmo). Se você pretende suportar os dispositivos Android nas versões anteriores à 2.2, precisará usar `fill_parent`, ao invés de `match_parent`.

Adicionando uma Imagem ao Seu Aplicativo

Embora ver texto seja divertido, os componentes realmente interessantes são adicionados via mecanismos de entrada e imagens. As seguintes seções demonstrarão como incluir imagens em seu aplicativo — é hora de colocar algo na tela!

Colocando uma imagem na tela

O primeiro elemento a adicionar à tela é a imagem do telefone (consulte as Figuras 4-2 e 4-3), portanto, primeiro você precisa da imagem do telefone, claro. Você pode fazer download de uma imagem no código-fonte deste livro, disponível no site web do livro ou pode usar uma imagem própria.

Por que você deve preocupar-se com as pastas de densidade

O Android suporta vários tamanhos de tela e densidades. Em vários pontos neste capítulo, mencionamos colocar imagens na pasta `mdpi`, que é para os dispositivos de densidade média. E os dispositivos de densidade pequena e grande? Se o Android não conseguir encontrar o recurso solicitado na densidade desejada, irá optar por uma densidade que ele puder encontrar. Se seu dispositivo tiver uma tela com alta densidade, a imagem será estendida e, muito provavelmente, terá muitos pixels.

Se seu dispositivo tiver um dispositivo de baixa densidade, a imagem será comprimida para caber nas dimensões da tela. Para evitar este problema, crie diversas versões de sua imagem para atender às diversas densidades de tela. Para obter mais informações, veja a página Supporting Multiple Screens (Suportando Diversas Telas) na documentação do Android em `http://developer.android.com/guide/practices/screens_support.html` (conteúdo em inglês).

Adicionar imagens a um projeto é simples: Arraste-as da pasta onde estão armazenadas para a pasta `res/drawable-mdpi`, como mostrado na Figura 4-8.

Para o aplicativo Silent Mode Toggle, você precisa de duas imagens de telefone: normal e silenciosa. Coloque ambas as imagens na pasta `res/drawable.mdpi`.

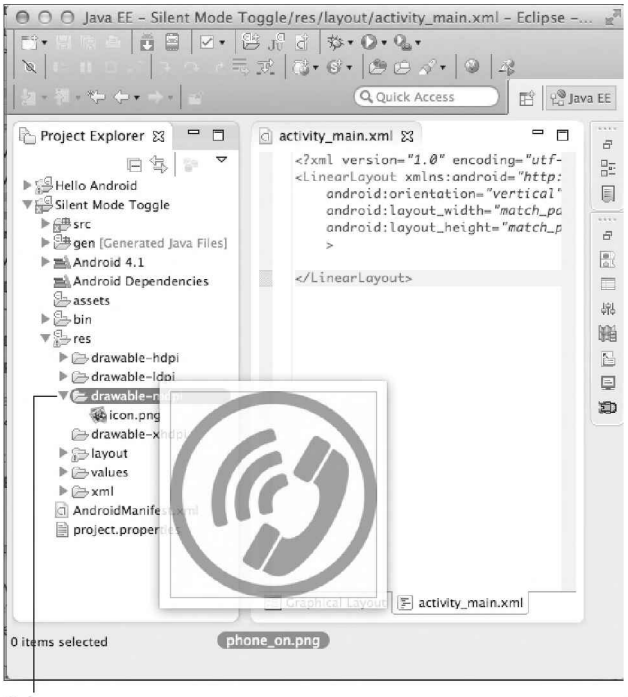


Figura 4-8: Arrastando o arquivo de imagem para a pasta `res/drawable.mdpi`.

Coloque as imagens nesta pasta.

Para acompanhar o resto do capítulo, certifique-se de que as imagens estejam nomeadas assim:

- **Imagem no modo normal:** `phone_on.png`
- **Imagem no modo silencioso:** `phone_silent.png`

Se suas imagens não estiverem nomeadas corretamente, você poderá renomeá-las agora. Seu projeto Eclipse deverá ficar como o mostrado na Figura 4-9.

Figura 4-9: O projeto Silent Mode Toggle, com as imagens de telefone.

Quando você arrasta imagens para o Eclipse, o ADT reconhece que a estrutura de arquivos do projeto mudou. Então, ele reconstrói o projeto porque a seleção Build Automatically (Construir Automaticamente) está ativada no menu Project (Projeto). A pasta `gen`, onde reside o arquivo `R.java`, é novamente gerada e o arquivo `R.java` recebe uma referência para as duas imagens novas adicionadas.

Você pode usar referências para esses recursos para adicionar imagens ao seu layout no código ou na definição XML. Você irá declará-las no layout XML na seção seguinte.

Adicionando a imagem ao layout

Para adicionar uma imagem ao layout, digite o seguinte no arquivo `activity_main.xml`, sobrescrevendo o conteúdo atual do arquivo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <ImageView
              android:id="@+id/phone_icon"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="center_horizontal"
              android:src="@drawable/phone_on" />

</LinearLayout>
```

Este código adiciona `ImageView` dentro de `LinearLayout`. Um `ImageView` permite que você projete uma imagem na tela do dispositivo.

Definindo as propriedades da imagem

O `ImageView` contém alguns parâmetros extras:

- **A propriedade** `android.id="@+id/phone_icon"`**:** o atributo id define o identificador exclusivo para a exibição no sistema Android. Você pode encontrar uma explicação detalhada sobre a nomenclatura do valor `android:id` em `http://developer.android.com/guide/topics/ui/declaringlayout.html` (conteúdo em inglês).
- **A propriedade** `layout_gravity`**:** esta propriedade define como colocar a exibição (em seus eixos x e y) em seu pai. Neste exemplo, o valor é definido como a constante `center_horizontal`. Esse valor instrui ao sistema Android para colocar o objeto no centro horizontal de seu contêiner, sem mudar seu tamanho. Você pode usar muitas outras constantes, tais como `center_vertical`, `top`, `bottom`, `left`, `right` e muito mais. Veja a documentação do Android `LinearLayout.LayoutParams` para obter uma lista completa.
- **A propriedade** `android:src="@drawable/phone_on"`**:** esta propriedade é um filho direto da classe `ImageView`. Você a usa para definir a imagem que deseja mostrar na tela.

Note o valor da propriedade `src` — `"@drawable/phone_on"`. O que você está vendo é o uso do arquivo `R.java`. Você pode referenciar os recursos de desenho via XML digitando o símbolo `(/)` no recurso desejado.

Certos atributos Android começam com o prefixo `layout_` — `android:layout_width`, `android:layout_height`, `android:layout_gravity` são exemplos. A convenção `layout_` informa que o atributo relaciona-se ao *pai* da exibição. Os atributos que não começam com `layout_` pertencem à exibição em si. Portanto, o atributo `android:src`

de `ImageView` informa à `ImageView` qual imagem usar, mas seu `android:layout_gravity` informa ao pai de `ImageView` (`LinearLayout`, neste caso) para fazer o layout de `ImageView` no centro do pai.

Definindo os recursos de desenho

Você não digita **@drawable-mdpi** para o identificador do recurso de desenho, **@drawable**, pois é trabalho do Android (não seu) suportar diversos tamanhos de tela. O sistema de layout Android sabe apenas sobre os desenhos — ele não sabe nada sobre os desenhos de densidade baixa, média, alta ou extra-alta durante o design. Na execução, o Android determina se e quando poderá usar os desenhos de densidade baixa, média ou alta.

Por exemplo, se o aplicativo estiver sendo executado em um dispositivo de alta densidade e o recurso de desenho solicitado estiver disponível na pasta `drawable-hdpi`, o Android usará esse recurso. Do contrário, usará a correspondência mais próxima que puder encontrar. O suporte para os vários tamanhos de tela e densidades é um tópico amplo (e complexo, em alguns aspectos). Para ter uma visão detalhada sobre este assunto, leia o artigo "Managing Multiple Screen Sizes" (Gerenciando Diversos Tamanhos de Tela) na documentação do Android em `http://developer.android.com/guide/practices/screens_support.html` (conteúdo em inglês).

A parte `phone_on` identifica o desenho que você deseja usar. O nome de arquivo da imagem é `phone_on.png`. Para ficar dentro das diretrizes de nomenclatura dos membros do Java, porém, a extensão de arquivo é removida, deixando `phone_on`. Se você abrir o arquivo `R.java` na pasta `gen`, verá uma variável-membro com o nome `phone_on`, não `phone_on.png`.

Graças ao ADT, você pode ver as opções disponíveis para esta propriedade via code completion (assistente de código). Coloque o cursor logo depois de `@drawable/` na propriedade `src` de `ImageView` no editor Eclipse, pressione Ctrl+barra de espaço. A janela code completion será aberta, como mostrado na Figura 4-10. Os outros nomes de recurso na janela são outras opções que você pode escolher para a parte `src` da definição do desenho.

Figura 4-10: Code completion, com os recursos.

Criando um Ícone de Inicialização para o Aplicativo

Quando o aplicativo é instalado, seu ícone ajuda os usuários a identificarem sua presença na inicialização do aplicativo. Quando você cria o aplicativo Silent Mode Toggle, o ADT inclui automaticamente um ícone de inicialização padrão, como mostrado à esquerda na Figura 4-11.

Você deve mudar esse ícone um tanto insosso para um próprio. Um ícone de telefone redondo funciona para o aplicativo Silent Mode Toggle, como mostrado à direita na Figura 4-11. Você pode criar um próprio (como mostrado na seção seguinte) ou usar aquele do código-fonte baixado no site web deste livro.

Figura 4-11: O ícone padrão (esquerda) e um ícone exclusivo (direita).

Projetando um ícone de inicialização personalizado

Criar seus próprios ícones de inicialização é bem fácil, graças ao projeto Android. O artigo "Icon Design Guidelines" (Diretrizes de Design de Ícones) na documentação do Android cobre todos os aspectos do design de ícones — um manual prático para criar ícones para a plataforma Android, um guia de estilo, uma lista de melhores práticas, informações sobre materiais e cores, diretrizes sobre tamanhos e posicionamento e (o melhor de tudo) modelos de ícones que você pode usar. Você pode encontrar recursos úteis para projetar os ícones em `http://d.android.com.guide/practices/ui_guidelines/icon_design.html` e `http://d.android.com/design/style/iconography.html` (conteúdo em inglês).

Trabalhando com modelos

Depois de fazer o download do SDK do Android, alguns modelos de ícone e materiais ficam disponíveis no disco rígido de seu computador para você usar imediatamente. Navegue para seu diretório de instalação SDK do Android (veja o Capítulo 2) e nele, navegue para o diretório `docs/shareables`. Você encontrará vários arquivos `.zip` que contêm modelos e amostras. Abra os modelos no programa de edição de imagem escolhido

e siga as diretrizes de design na documentação para criar seu próximo conjunto de ícones de arrasar.

Coincidindo os tamanhos dos ícones e as densidades das telas

Como toda densidade da tela requer um ícone com um tamanho diferente, você, como designer, precisa saber o tamanho que o ícone deve ter. Cada densidade deve ter seu próprio tamanho de ícone para uma exibição adequada na tela (sem ficar borrado, esticado ou comprimido).

A Tabela 4-4 resume os tamanhos de ícone para cada uma das três densidades de tela.

Tabela 4-4	Layouts do SDK do Android
Densidade da tela	Tamanho do ícone em pixels
Tela com baixa densidade (ldpi)	36 x 36
Tela com densidade média (mdpi)	48 x 48
Tela com alta densidade (hdpi)	72 x 72
Tela com extra-alta densidade (xhdpi)	96 x 96

Adicionando um ícone de inicialização personalizado

Para colocar seu ícone de inicialização personalizado no projeto, siga estas etapas:

1. **Renomeie o ícone da imagem como** `ic_launcher.png`**.**
2. **Arraste seu ícone para a pasta** `drawable-mdpi`**.**

   O Eclipse pergunta se você deseja sobrescrever o arquivo `ic_launcher.png` existente.

3. **Clique em Yes (Sim).**

   O arquivo `ic_launcher.png` agora está na pasta `drawable-mdpi`.

Você não terminou ainda! Para as pastas `ldpi, hdpi` e `xhdpi`, você precisa de versões do ícone com densidades baixa, alta e extra-alta. Copie os respectivos ícones para as pastas `ldpi, hdpi` e `xhdpi`.

Se você não copiar os ícones de outras densidades para suas respectivas pastas, os usuários que têm um dispositivo com baixa ou alta densidade receberão

o ícone de inicialização padrão (consulte a Figura 4-11), ao passo que os dispositivos com densidade média receberão o novo ícone incluído no projeto.

Você arrastou o arquivo para a pasta `drawable_mdpi` — e daí? Cada uma das outras pastas contém sua própria versão do ícone. Abra as pastas `drawable-hdpi`, `drawable-xhdpi` e `drawable-ldpi` em seu projeto Eclipse e poderá ver que cada densidade tem seu próprio arquivo `ic_launcher.png`. Certifique-se de colocar o ícone correto em cada pasta específica.

Adicionando uma Exibição do Botão de Alternância

Os dispositivos Android vêm totalmente equipados com várias exibições que incluem botões, caixas de seleção e campos de entrada de texto para que você possa construir rapidamente sua interface de usuário. Algumas exibições são mais complexas, tais como seletores de data, relógios e controles de zoom.

As exibições também fornecem eventos de interface que informam quando um usuário interagiu com determinada exibição, como, por exemplo, clicar em um botão.

Você precisa adicionar uma exibição do botão ao seu aplicativo para que possa alternar o modo silencioso no telefone.

Para adicionar um botão ao seu layout, digite o seguinte código depois de `ImageView`:

```
<Button
        android:id="@+id/toggleButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Toggle Silent Mode"
 />
```

Agora, você adicionou um botão à sua exibição com o identificador de recurso `toggleButton`. É assim que você referencia o botão no código Java. (O Capítulo 5 cuida do código.)

A altura e a largura foram definidas para `wrap_content`, o que informa ao sistema de layout do Android para colocar a exibição na tela e ocupar apenas o espaço necessário. A propriedade `layout_gravity` é igual à `ImageView` acima, centralizada na horizontal.

A última propriedade introduzida nesta exibição é a propriedade `text`, que define o texto do botão para `Toggle Silent Mode`.

Sua base de código completa agora deverá ficar assim:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <ImageView
              android:id="@+id/phone_icon"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="center_horizontal"
              android:src="@drawable/phone_on" />
    <Button
              android:id="@+id/toggleButton"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="center_horizontal"
              android:text="Toggle Silent Mode" />

</LinearLayout>
```

Visualizando o Aplicativo no Designer Visual

Para dar uma olhada em como está o layout no designer visual, clique na guia Graphical Layout (Layout Gráfico) para exibi-la, como mostrado na Figura 4-12.

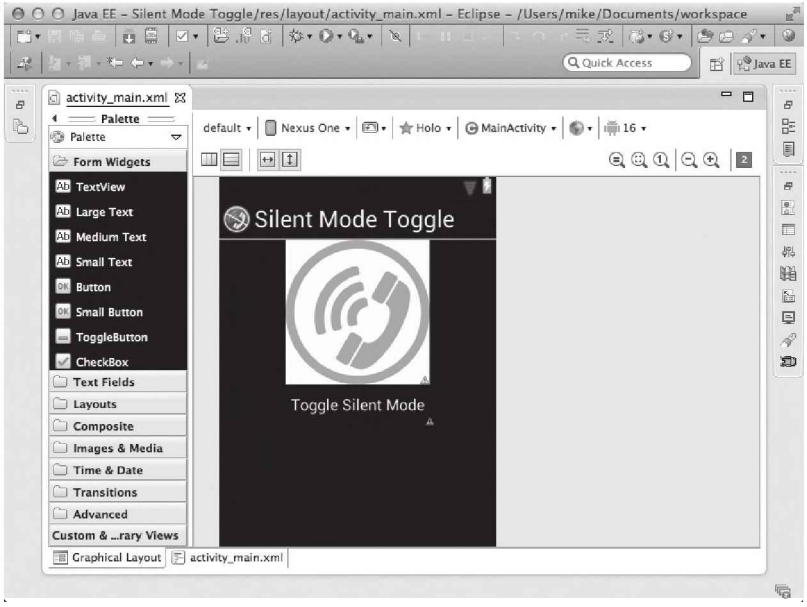


Figura 4-12: A exibição do layout segundo o designer visual.

Que horror! O fundo está preto e sua imagem é branca. Não parece certo. Você deve tornar o fundo de seu layout branco para fazer a imagem se misturar com ele. Eis como fazer:

1. **Selecione a guia** `activity_main.xml`**.**

2. **Adicione a propriedade** `background` **a seu** `LinearLayout`**:**

```
android:background="#ffffff"
```

O valor hexadecimal #ffffff é a cor branca opaca. Você pode digitar qualquer cor, tal como #ff0000, que é vermelho.

Você também pode definir uma imagem como fundo, usando um recurso.

3. **Verifique se a definição de** `LinearLayout` **ficou assim:**

```
<LinearLayout xmlns:android="http://schemas.android.
        com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffffff">
```

4. **Salve o arquivo.**

5. **Selecione a guia Graphical Layout (Layout Gráfico) para exibir o designer visual.**

A Figura 4-13 mostra o layout final.

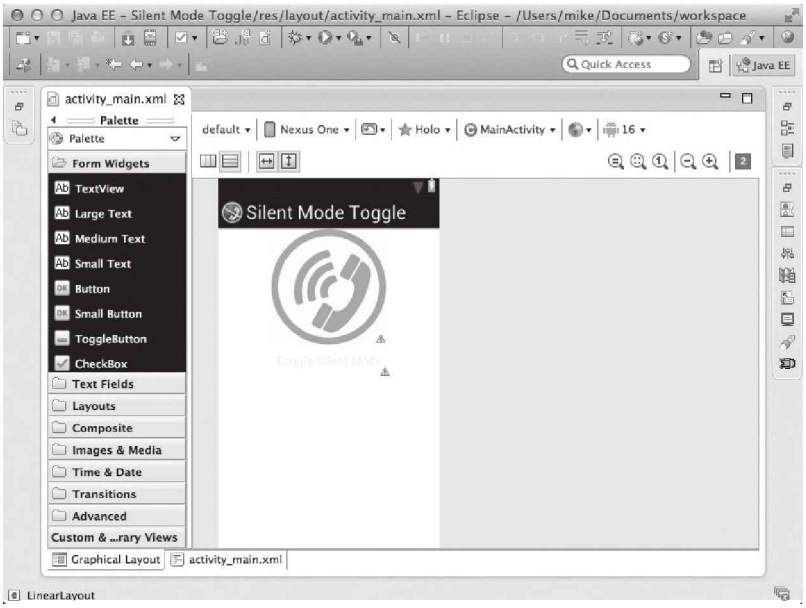


Figura 4-13: O layout Silent Mode Toggle final.

Designer visual ADT

O designer visual tem muitas configurações diferentes. Por padrão, o designer é definido para Nexus One, um dos primeiros e principais smartphones da Google. Selecionar a lista suspensa Devices (Dispositivos) no designer visual mostra quais dispositivos você pode simular com seu layout. As configurações representam as várias configurações possíveis do dispositivo. O Nexus One tinha dois estados que eram válidos em execução:

- **Paisagem:** O telefone está no modo paisagem horizontal.
- **Retrato:** O telefone é segurado em um modo retrato vertical.

Cada dispositivo na lista suspensa Devices tem seu próprio conjunto de configurações. Você pode criar suas próprias configurações personalizadas escolhendo Devices (Dispositivos)⇨Custom (Personalizar)⇨Custom⇨New (Novo).

Capítulo 5

Codificando Seu Aplicativo

Neste Capítulo

- Vendo como as atividades funcionam no Android
- Codificando sua própria atividade
- Usando as classes da estrutura Android
- Instalando um aplicativo
- Usando ferramentas de depuração
- Testando seu aplicativo no mundo real

Provavelmente, você está ansioso para começar a codificar seu aplicativo. Neste capítulo, você codifica-o, do início ao fim. Mas antes de poder começar a lidar com os bits e os bytes, você precisa compreender bem o que são as atividades.

Compreendendo as Atividades

Uma *atividade* é uma ação única e fechada em si mesma, que um usuário pode tomar. Por exemplo, uma atividade pode apresentar uma lista de itens de menu na qual um usuário pode escolher ou pode exibir fotografias junto com títulos. Um aplicativo pode consistir em apenas uma atividade ou (como a maioria dos aplicativos no sistema Android) várias. Embora as atividades possam trabalhar juntas, dando a ideia de serem apenas um aplicativo coeso, elas funcionam independentemente uma das outras.

A atividade no Android é parte importante do ciclo de vida geral de um aplicativo e o modo como as elas são inicializadas e agrupadas é um aspecto fundamental do modelo do aplicativo Android. Cada atividade é implementada como uma especialização da classe de base `Activity`.

Quase todas as atividades interagem com o usuário, portanto, a classe `Activity` cria para você a janela na qual você pode colocar sua interface de usuário (IU). As atividades são apresentadas, com muita frequência, no modo de tela cheia, mas em algumas situações, você pode encontrar uma atividade flutuando em uma janela ou incorporada dentro de outra atividade – algo conhecido como *grupo de atividades*.

Trabalhando com métodos, pilhas e estados

Dois métodos importantes que quase todas as atividades implementam são

- `onCreate`: Onde a atividade é inicializada. E mais importante, é onde você informa à atividade qual layout usar, utilizando um identificador de recurso de layout — considerado o ponto de entrada de sua atividade.
- `onPause`: Onde você lida com o usuário deixando sua atividade. Qualquer alteração feita pelo usuário deve ser aplicada neste ponto (se você precisar salvá-la).

As atividades no sistema são gerenciadas como uma *pilha de atividades*. Quando uma nova atividade é criada, é colocada no topo da pilha e torna-se a atividade em execução. A atividade que estava em execução anteriormente sempre permanece abaixo na pilha e retorna ao primeiro plano apenas quando a nova atividade sai.

Para ser um programador bem-sucedido, você deve entender a importância de como e por que a atividade funciona internamente. Você não só entenderá melhor a plataforma Android, como também solucionará com precisão os problemas de comportamento estranho de seu aplicativo em tempo de execução.

Uma atividade tem basicamente quatro estados, como descrito na Tabela 5-1.

Tabela 5-1 Layouts do SDK do Android

Estado da atividade	*Descrição*
Ativa/em execução	A atividade está no primeiro plano da tela (no topo da pilha).
Pausada	A atividade perdeu o foco, mas ainda é visível (uma atividade nova, de menor tamanho ou transparente tem o foco no topo de sua atividade). Como uma atividade pausada está completamente ativa, ela pode manter o estado e as informações do membro, e permanecer anexada ao gerenciador de janelas no Android. Contudo, até o Gingerbread (3.0), a atividade pode ser encerrada pelo sistema Android em condições de memória extremamente baixa.
Parada	Se uma atividade ficar encoberta por outra atividade, ela será parada. Ela mantém todo o estado e as informações do membro, mas não é visível para o usuário. Portanto, a janela é ocultada e geralmente será encerrada pelo sistema Android quando a memória for necessária em outro lugar.
Criada e retomada	O sistema pausou ou parou a atividade. O sistema pode requerer a memória solicitando que termine ou pode encerrar o processo. Quando exibe a atividade para o usuário, deve retomar reiniciando e restaurando seu estado anterior.

Controlando o ciclo de vida de uma atividade

A Figura 5-1 mostra os caminhos importantes de uma atividade — o *ciclo de vida da atividade*.

Os retângulos representam os métodos de callback que você pode implementar para responder aos eventos na atividade. Os ovais sombreados representam os principais estados da atividade.

O ciclo de vida da atividade é um tópico grande e complexo, e as seguintes seções cobrirão apenas o básico. Se você quiser ler mais sobre os ciclos de vida da atividade, verifique o artigo "Activity Life Cycle and Process Life Cycle" (Ciclo de Vida da Atividade e Ciclo de Vida dos Processos) na documentação Android em `http://d.android.com/reference/android/app/Activity.html#ProcessLifecycle` (conteúdo em inglês).

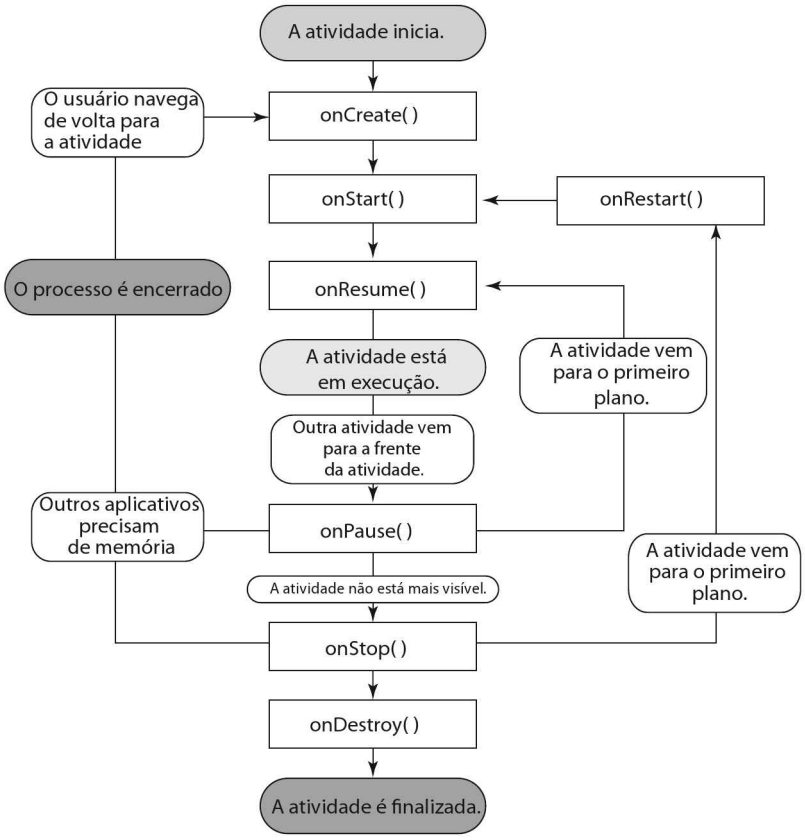


Figura 5-1: O ciclo de vida da atividade.

Monitorando os principais loops

Você pode estar interessado em monitorar estes três loops em sua atividade:

- A **duração inteira** ocorre entre a primeira chamada para `onCreate()` e a chamada final para `onDestroy()`. A atividade realiza toda a configuração global em `onCreate()` e libera todos os recursos restantes em `onDestroy()`. Por exemplo, se você criar um thread para fazer o download de um arquivo na Internet em segundo plano, ele poderá ser inicializado no método `onCreate()`. Esse thread pode ser parado no método `onDestroy()`.
- A **duração visível** da atividade ocorre entre os métodos `onStart()` e `onStop()`. Durante este tempo, o usuário pode ver a atividade na tela (embora possa não estar no primeiro plano interagindo com o usuário, o que pode acontecer quando o usuário está interagindo com uma caixa de diálogo). Entre esses dois métodos, você pode manter os recursos necessários para mostrar e executar sua atividade. Por exemplo, você pode criar uma sub-rotina de tratamento de evento para monitorar o estado do telefone. O estado do telefone pode mudar e essa sub-rotina pode informar à atividade que o telefone entrou no modo Avião e reagir de acordo. Você configuraria a sub-rotina em `onStart()` e destruiria qualquer recurso que estivesse acessando em `onStop()`. Os métodos `onStart()` e `onStop()` podem ser chamados diversas vezes quando a atividade torna-se visível ou oculta para o usuário.
- A **duração em primeiro plano** da atividade começa na chamada para `onResume()` e termina na chamada para `onPause()`. Durante esse tempo, a atividade está na frente de todas as outras atividades e está interagindo com o usuário. Normalmente, uma atividade alterna entre `onResume()` e `onPause()` diversas vezes. Por exemplo, quando o dispositivo dorme ou quando uma nova atividade lida com determinado evento. Portanto, o código nesses métodos deve ser bem leve.

Exibindo os métodos da atividade

O ciclo de vida inteiro da atividade resume-se a estes métodos:

```
public class Activity extends ApplicationContext {
     protected void onCreate(Bundle savedInstanceState);
     protected void onStart();
     protected void onRestart();
     protected void onResume();
     protected void onPause();
     protected void onStop();
     protected void onDestroy();
  }
```

Todos o métodos podem ser sobrescritos e o código personalizado pode ser colocado em todos eles. Todas as atividades implementam `onCreate()` para a inicialização e também podem implementar `onPause()` para a limpeza. Você sempre deve chamar a superclasse (classe de base) ao implementar esses métodos.

Seguindo o caminho de uma atividade

O movimento de uma atividade em seu ciclo de vida é assim:

- `onCreate()`: Chamado quando a atividade é criada pela primeira vez. Você inicializa a maioria das variáveis da classe de sua atividade aqui. `onStart()` é sempre chamado em seguida. Destrutível: Não. Próximo: `onStart()`.
- `onRestart()`: Chamado após a sua atividade ser interrompida, antes de ser iniciada novamente. `onStart()` é sempre chamado em seguida. Destrutível: Não. Próximo: `onStart()`.
- `onStart()`: Chamado quando sua atividade está ficando visível para o usuário. Seguida por `onResume()` se a atividade for trazida para o primeiro plano ou `onStop()` se ficar oculta para o usuário. Destrutível: Não. Próximo: `onResume()` ou `onStop()`.
- `onResume()`: Chamado quando a atividade ficar disponível para interagir com o usuário. A atividade fica no topo da pilha de atividades neste ponto. Destruível: Não. Próximo: `onPause()`.
- `onPause()`: Chamado quando o sistema está prestes a retomar uma atividade anterior ou se o usuário navegou para outra parte do sistema, tal como pressionando a tecla Home. Este estágio é geralmente usado para confirmar as alterações não gravadas que precisam ser persistidas. Se a atividade for trazida de volta para o primeiro plano, `onResume()` será chamado; se a atividade ficar invisível para o usuário, `onStop()` será chamado. Destrutível: Sim, mas apenas no Gingerbread (2.3) ou anterior. Próximo: `onResume()` ou `onStop()`.
- `onStop()`: Chamado quando a atividade não fica mais visível para o usuário porque outra atividade foi retomada. Isso pode acontecer porque outra atividade foi iniciada ou uma atividade anterior foi retomada e agora está no topo da pilha de atividades. É seguido por `onRestart()` se esta atividade está retornando para interagir com o usuário ou por `onDestroy()` se esta atividade está indo embora. Destruível: Sim. Próximo: `onRestart()` ou `onDestroy()`.
- `onDestroy()`: A chamada final que você recebe antes que sua atividade seja destruída. Este método é chamado porque a atividade está terminando (tal como alguém chamando `finish()` nela) ou porque o sistema está destruindo temporariamente a atividade para recuperar espaço. Você pode distinguir entre esses dois com o método `isFinishing()`, que ajuda a identificar se o método está terminando ou o sistema está encerrando-o. O método `isFinishing()` é geralmente usado dentro de `onPause()` para determinar se a atividade está pausando ou sendo destruída. Destrutível: Sim. Próximo: Nada.

O indicador *destrutível* no final de cada descrição do método da atividade aponta as atividades que o sistema Android pode encerrar a qualquer momento e sem aviso. Você deve, portanto, usar o método `onPause()` para completar qualquer limpeza para gravar os dados que devem ser persistidos (tais como os dados alterados pelo usuário) em seu mecanismo de armazenamento.

Reconhecendo as alterações da configuração

Uma *alteração da configuração* é uma mudança feita na orientação da tela (por exemplo, se o usuário move a tela para o lado e para trás, se move do modo retrato para paisagem ou vice-versa), no idioma ou em um dispositivo de entrada. Uma alteração da configuração faz com que sua atividade seja destruída ao completar o ciclo de vida normal: `onPause()` seguido de `onStop()`, e então, de `onDestroy()`. Depois do método `onDestroy()` ser chamado, o sistema cria uma nova instância da atividade a ser criada, o que ocorre porque os recursos, os arquivos de layout e outros elementos podem precisar mudar dependendo da configuração atual do sistema. Por exemplo, um aplicativo pode ficar completamente diferente se o usuário estiver interagindo com ele no modo Retrato, em comparação com o modo Paisagem (de sua parte).

Criando Sua Primeira Atividade

Você já pode ter criado sua primeira atividade se criou um projeto usando o New Android Project Wizard (Assistente para Novo Projeto Android) no Capítulo 3: a atividade `MainActivity`. Abra o arquivo `MainActivity.java` em seu projeto para aperfeiçoá-lo nas seguintes seções.

Iniciando com onCreate

O ponto de entrada em seu aplicativo é o método `onCreate()`. O código do arquivo `MainActivity.java` já contém uma implementação do método `onCreate()`. É onde você começa a escrever o código! No momento, seu código deve ser assim:

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

Você escreve o código de inicialização diretamente abaixo do método `setContentView()`.

Sempre inclua a seguinte chamada de método em seu método `onCreate( )`:

```
super.onCreate(savedInstanceState);
```

É necessário para que o aplicativo seja executado. Esta linha direciona a classe de base `Activity` para realizar o trabalho de configuração para a classe `MainActivity`. Se você omitir essa linha de código, verá uma exceção de execução.

Informando ao Android para exibir a interface do usuário

Por padrão, uma atividade não tem ideia do que é sua interface de usuário. Pode ser um formulário simples que permite ao usuário digitar informações a serem gravadas; um aplicativo visual, baseado em câmera, aumentado e com realidade virtual (tal como o Layar na Google Play Store); ou uma interface de usuário desenhada dinamicamente, como, por exemplo, um jogo em 2D ou 3D. Como desenvolvedor, é seu trabalho informar à atividade qual layout ela deve carregar.

Para mostrar a interface do usuário na tela, você tem que definir a exibição do conteúdo para a atividade, adicionando esta linha de código:

```
super.onCreate(savedInstanceState);
```

`R.layout.activity_main` é o arquivo `activity_main.xml` localizado no diretório `res/layouts`. É o layout definido no Capítulo 4.

Lidando com a entrada do usuário

O aplicativo Silent Mode Toggle tem pouca interação com o usuário. A única interação que seu aplicativo terá é um botão que o usuário clica para alternar o modo silencioso.

Para responder a esse evento de toque, você precisa registrar um *atendente de evento (event listener)*, que responde a um evento no sistema Android. Embora você encontre vários tipos de eventos no sistema Android, dois dos mais usados são os eventos do teclado e os eventos de toque (também conhecidos como cliques).

Eventos do teclado

Um *evento do teclado* ocorre sempre que determinada tecla do teclado é pressionada. Por exemplo, se o usuário pressionar a tecla de ativação Alt+E em seu aplicativo, você poderá querer que a exibição alterne para o modo Edit (Editar). Responder aos eventos do teclado permite que você faça isto. Se você precisar sobrescrever o método `onKeyDown` para usar seu próprio evento do teclado, faça assim:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
          // TODO Auto-generated method stub
          return super.onKeyDown(keyCode, event);
}
```

Eventos de toque

Um *evento de toque* ocorre sempre que o usuário toca em um componente na tela. A plataforma Android reconhece cada evento de toque como um evento de clique. Os exemplos de exibições que podem responder aos eventos de toque incluem (mas não estão limitados a):

- `Button`
- `ImageButton`
- `EditText`
- `Spinner`
- `ListView Rows`
- `MenuItem`

Todas as exibições no sistema Android podem reagir a um toque; porém, alguns componentes têm sua propriedade *clickable* definida para false por padrão. Você pode anular essa definição em seu arquivo de layout ou no código para permitir que uma exibição seja clicada definindo o atributo `clickable` na exibição ou o método `setClickable()` no código.

Escrevendo sua primeira sub-rotina de eventos

Para seu aplicativo responder ao evento de clique do usuário, alternando o modo silencioso, você responderá ao evento de clique exposto pelo botão.

Inserindo o código

Digite em seu editor o código mostrado na Listagem 5-1. Ele demonstra como implementar uma sub-rotina (um handle) de clique para `toggleButton`. O código consiste no método `onCreate()` inteiro com o novo código. Você pode inserir o código do botão (em negrito) ou sobrescrever seu código `onCreate` inteiro.

Listagem 5-1: O arquivo inicial da classe com um método onClickListener padrão para botões

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

    Button toggleButton = (Button)findViewById(R.id.toggleButton);
    toggleButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
        }
    });
}
```

Esta listagem usa o método `findViewById()`, que está disponível para todas as atividades no Android. Esse método, que permite encontrar qualquer exibição dentro do layout da atividade e fazer algum trabalho com ela, sempre retorna uma classe `View`, que você deve fazer a conversão para o tipo apropriado antes de poder iniciar. No seguinte código (que é uma linha da Listagem 5-1), você está fazendo a conversão do `View` retornado de `findViewById()` para um `Button` (que é uma subclasse de `View`).

```
Button toggleButton = (Button)findViewById(R.id.toggleButton);
```

Imediatamente após esta linha de código, você começa a configurar a sub-rotina de eventos.

O código do tratamento de eventos é colocado em seguida depois de você recuperar `Button` no layout. Configurar a sub-rotina de eventos é tão simples quanto definir um novo `View.OnClickListener`. Esse atendente (listener) de clique contém um método `onClick( )` que é chamado depois do usuário tocar no botão. É onde você coloca o código para lidar com a alternância do modo silencioso.

Faça a conversão para o devido tipo. Se o tipo em seu arquivo de layout for diferente do que você está tentando converter (se está tentando transformar um `ImageView` no arquivo de layout em um `ImageButton`, por exemplo), você irá paralisar seu aplicativo.

Quando você digitar este código em seu editor, poderá ver linhas vermelhas e irregulares, como mostrado na Figura 5-2. Essas linhas informam que o Eclipse não sabe o que é um "botão".

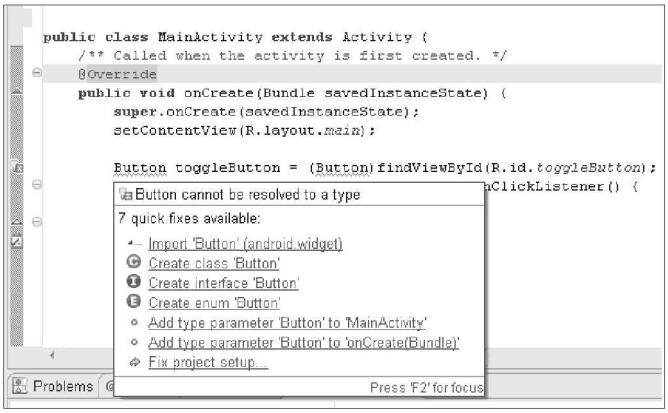


Figura 5-2: O Eclipse informa que não consegue encontrar a classe.

Siga estas etapas para corrigir o problema:

1. **Coloque o cursor sobre a linha irregular e deixe-o lá por um momento.**

   Uma pequena janela contextual é aberta fornecendo várias opções. (Consulte a Figura 5-2).

2. **Selecione a primeira opção — Import 'Button' (Importar 'Botão').**

   A seguinte instrução de importação é adicionada ao topo do arquivo:

   ```
   import android.widget.Button;
   ```

   Esta instrução de importação informa ao Eclipse onde `Button` está localizado nos pacotes Android.

Extraindo o código para um método

O código está começando a ficar pesado e difícil de ler. Neste ponto, a melhor coisa a fazer é extrair o novo código do botão para um método que você pode chamar de dentro de `onCreate()`. Para tanto, você cria um método `void` privado chamado `setButtonClickListener()` que contém o código do botão digitado. Esse novo método é colocado no método `onCreate()`. O novo código é mostrado na Listagem 5-2.

Listagem 5-2: O atendente do botão extraído para um método

```
public class MainActivity extends Activity {
     /** Called when the activity is first created. */
     @Override
     public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        setButtonClickListener();                                  →8
    }

    private void setButtonClickListener() {                        →11
            Button toggleButton = (Button)findViewById(R.id.toggleButton);
            toggleButton.setOnClickListener(new View.OnClickListener() {
               public void onClick(View v) {
                  // TODO Auto-generated method stub
               }
            });

    }
}
```

A Listagem 5-2 funciona assim:

→ **8** Nesta linha, um método é chamado para configurar o atendente de clique botão.

→ **11** O novo método está sendo chamado.

Agora, você pode responder ao evento de clique fornecendo o código para o método `onClick()` de seu botão.

Trabalhando com as Classes da Estrutura Android

Esta seção entra na parte boa — os detalhes do desenvolvimento Android e suas classes da estrutura Android! Sim, as atividades e as exibições são partes integrantes do sistema, mas simplesmente são o "encanamento" requerido em qualquer sistema operacional moderno (em uma capacidade ou outra). A diversão real está para começar.

As seguintes seções descreverão como verificar o estado da campainha do telefone para determinar se está no modo normal (campainha alta) ou no modo silencioso. A partir daí, você pode começar a alternar o modo de campainha do telefone.

Obtendo um bom serviço

Para acessar a campainha Android, você precisará de muitos acessos ao `AudioManager` no Android, que é responsável por gerenciar o estado da campainha. Portanto, você deve inicializá-lo em `onCreate()`.

Toda inicialização importante precisa acontecer em `onCreate()`.

Primeiro, você precisa criar uma variável AudioManager privada no nível da classe com o nome mAudioManager. Digite esse nome no topo de seu arquivo de classe, diretamente após a linha de declaração da classe, como mostrado na Listagem 5-3.

Listagem 5-3: Adicionando a variável AudioManager no nível da classe

```
package com.dummies.android.silentmodetoggle;

import android.app.Activity;
import android.media.AudioManager;                                  → 4
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

    private AudioManager mAudioManager;                             →11

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

        setButtonClickListener();

        mAudioManager = (AudioManager)getSystemService(AUDIO_SERVICE);   →20
    }

    private void setButtonClickListener() {
            Button toggleButton = (Button)findViewById(R.id.toggleButton);
            toggleButton.setOnClickListener(new View.OnClickListener() {
                    public void onClick(View v) {
                        // TODO Auto-generated method stub
                    }
            });
    }
}
```

Esta lista explica rapidamente as linhas numeradas:

→ 4 A instrução `import` que introduz o pacote necessário para que você possa usar `AudioManager`.

→ **11** A variável `private AudioManager` no nível da classe. Como é de toda a classe, você pode ter acesso a ela em outras partes da atividade.

→ **20** Inicializa a variável `mAudioManager` obtendo o serviço na chamada do método de base `Activity getSystemService()`.

Pare! O que é `getSystemService()`? Herdando da classe de base `Activity`, `AudioManager` recebe todas as vantagens de ser uma atividade, inclusive o acesso à chamada do método `getSystemService()`. Esse método retorna a classe de base `Object` do Java, portanto, você tem que fazer a conversão dela para o tipo de serviço que está solicitando.

Esta chamada retorna todos os serviços do sistema disponíveis com os quais você pode precisar trabalhar. Todos os serviços que são retornados podem ser encontrados na classe `Context` na documentação Android em `http://d.android.com/reference/android/content/Context.html` (conteúdo em inglês). Os tipos mais populares de serviços do sistema são

- `AUDIO_SERVICE`
- `LOCATION_SERVICE`
- `ALARM_SERVICE`

Alternando o modo silencioso com AudioManager

Depois de ter uma instância global da classe `AudioManager`, você pode começar a verificar o estado da campainha e alterná-la. O código que você precisa adicionar ou modificar está em negrito na Listagem 5-4.

Listagem 5-4: Adicionando a alternância ao aplicativo

```
package com.dummies.android.silentmodetoggle;

import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.media.AudioManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends Activity {
```

Listagem 5–4: (continuação)

```
private AudioManager mAudioManager;
private boolean mPhoneIsSilent;                                    →14

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

    mAudioManager = (AudioManager)getSystemService(AUDIO_SERVICE);

    checkIfPhoneIsSilent();                                        →23

    setButtonClickListener()                                       →25
}

private void setButtonClickListener() {
        Button toggleButton = (Button)findViewById(R.id.toggleButton);
        toggleButton.setOnClickListener(new View.OnClickListener() {
           public void onClick(View v) {

            if (mPhoneIsSilent) {                                  →32
                // // Mude para o modo normal
                mAudioManager
                        .setRingerMode(AudioManager.RINGER_MODE_NORMAL);
                mPhoneIsSilent = false;
            } else {
                // Mude para o modo silencioso
                mAudioManager
                    .setRingerMode(AudioManager.RINGER_MODE_SILENT);
                    mPhoneIsSilent = true;
            }

            // Agora, alterne a IU novamente
            toggleUi();                                            → 44
           }
    });
}

/**
 * Alterna as imagens da IU de silencioso para normal e vice-versa.
 */
private void checkIfPhoneIsSilent() {                              → 53
    int ringerMode = mAudioManager.getRingerMode();
    if (ringerMode == AudioManager.RINGER_MODE_SILENT) {
        mPhoneIsSilent = true;
    } else {
        mPhoneIsSilent = false;
    }
```

```
    }

    /**
     * Toggles the UI images from silent to normal and vice versa.
     */
     private void toggleUi() {                                          →66


        ImageView imageView = (ImageView) findViewById(R.id.phone_icon);
        Drawable newPhoneImage/

        if (mPhoneIsSilent) {
            newPhoneImage =
                getResources().getDrawable(R.drawable.phone_silent);

        } else {
            newPhoneImage =
                getResources().getDrawable(R.drawable.phone_on);
        }

        imageView.setImageDrawable(newPhoneImage);
    }

    @Override                                                           →84
    protected void onResume() {
        super.onResume();
        checkIfPhoneIsSilent();
        toggleUi();
    }
}
```

Esta lista explica rapidamente o que faz cada nova seção do código:

→ 14 Configura uma nova variável `booleanmPhoneIsSilent` no nível da classe para controlar o estado da campainha.

→ 23 Chama o método `checkIfPhoneIsSilent()` para inicializar `mPhoneIsSilent`. O valor padrão de uma variável `boolean` é falso — que poderá estar errado se o telefone estiver no modo Silencioso. Esta linha descobre o que acontece quando o modo da campainha é alternado.

→ 25 O código de tratamento do evento do botão foi movido para a parte inferior do método `onCreate()` porque ele depende da configuração da variável `mPhoneIsSilent`. Mesmo que nada provavelmente aconteça, você deve manter o código organizado.

Código limpo é código gerenciável.

→ 32 O código entre as Linhas 32 e 44 lida com um toque do usuário no botão de alternância, verificando se a campainha está ativada através da variável `mPhoneIsSilent` do nível da classe.

Se a campainha estiver silenciosa, o código entrará no primeiro bloco `if` e mudará o modo da campainha para `RINGER_MODE_NORMAL`, que ativa novamente a campainha. A variável `mPhoneIsSilent` também será alterada para `false` na próxima vez em que o código for executado.

Se a campainha não estiver silenciosa, o código entrará no bloco de código `else`. Esse bloco de código muda o modo da campainha de seu estado atual para `RINGER_MODE_SILENT`, que desativa a campainha. O bloco `else` também definirá a variável `mPhoneIsSilent` para `true` na próxima vez.

→ **44** O método `toggleUi()` muda a interface de usuário para dar a ele um identificador visual de que o modo no telefone mudou. Sempre que o modo da campainha muda, o método `toggleUi()` precisa ser chamado.

→ **53** O método `checkIfPhoneIsSilent()` inicializa a variável de nível da classe `mPhoneIsSilent` no método `onCreate()`. Se você não fizer isso, seu aplicativo não saberá em qual estado está a campainha de `AudioManager`. Se o telefone estiver silencioso, `mPhoneIsSilent` será definida para `true`; do contrário, será `false`.

→ **66** O método `toggleUi()` muda `ImageView` a partir do layout criado no Capítulo 4, dependendo do estado da campainha.

Se a campainha estiver silenciosa, a interface de usuário exibirá uma imagem mostrando que a campainha do telefone está desligada. Se a campainha estiver no modo Normal, a imagem indicará que a campainha do telefone está ligada.

Ambas as imagens estão nos diretórios de recursos (veja Capítulo 4). `ImageView` encontra-se dentro do layout e depois de detectado o modo, `View` é atualizado obtendo a imagem correta em `getResources().getDrawable(...)` e definido com a chamada `setImageDrawable(...)` em `ImageView`. Esse método atualiza a imagem que é exibida em `ImageView` na tela.

→ **84** O método `onResume()` é sobrescrito para que seu aplicativo identifique corretamente seu estado. A variável `mPhoneIsSilent` controla o estado da campainha do telefone, mas apenas para a classe. A pessoa que usar seu aplicativo também precisará saber em qual estado está o telefone, portanto, `onResume()` chama `toggleUi()` para alternar a interface de usuário.

A chamada `toggleUi()` é colocada estrategicamente no método `onResume()` por uma simples razão: supor que o

usuário pode abrir o aplicativo Silent Toggle Mode, e então, retornar para a tela inicial e desligar o telefone usando os controles do hardware. Quando o usuário retorna para a atividade, ela é retomada e trazida para o primeiro plano. Nesse momento, `onResume()` é chamado para verificar o estado do modo da campainha e atualizar a interface do usuário de acordo. Se o usuário alterou o modo, o aplicativo reagirá corretamente.

Instalando Seu Aplicativo

Você terminou — escreveu seu primeiro aplicativo. Nas próximas seções, você instalará seu aplicativo no emulador e o fará entrar em ação!

Executando seu aplicativo em um emulador

Quando você executa seu aplicativo em um emulador, o ADT é esperto o bastante para lembrar a última configuração de inicialização e usá-la por padrão (se você precisar mudar a configuração de inicialização — digamos para iniciar uma atividade diferente ou aplicativo — volte ao Capítulo 3).

O emulador e o Eclipse conversam entre si através da Ponte de Depuração Android (Android Debuc Bridge — adb). Você instalou o adb com as Ferramentas de Desenvolvimento Android (Android Development Tools ADT) no Capítulo 2.

É hora de instalar o aplicativo no emulador. Siga estas etapas:

1. **No Eclipse, escolha Run (Executar)⇨Run ou pressione Ctrl+F11 para executar o aplicativo.**

   Você verá a janela Run As (Executar Como), mostrada na Figura 5-3.

2. **Escolha Android Application (Aplicativo Android) e clique em OK para iniciar o emulador.**

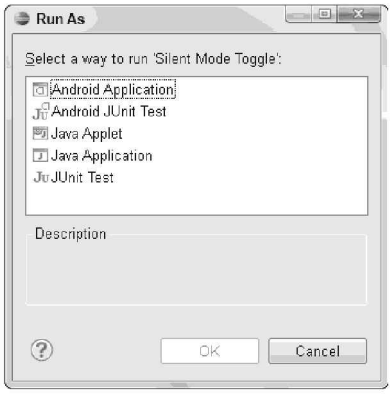


Figura 5-3: A caixa de diálogo de configuração Run As.

3. **Aguarde o emulador ser carregado, e então, desbloqueie-o.**

   Se você não souber como desbloquear o emulador, consulte o Capítulo 3. Quando o emulador for desbloqueado, seu aplicativo iniciará e o emulador executará seu programa, como mostrado na Figura 5-4.

   Se seu aplicativo não iniciar, execute-o de novo escolhendo Run⇨Run ou pressionando Ctrl+F11

Figura 5-4: O emulador executando o aplicativo.

4. **Clique no botão Toggle Silent Mode (Alternar Modo Silencioso) para ver a imagem mudar para o telefone dentro do círculo com uma barra vermelha, como mostrado na Figura 5-5.**

   Observe o novo ícone na barra de notificação — o ícone Silent Notification (Notificação de Modo Silencioso).

Figura 5-5: O aplicativo no modo silencioso, com o ícone Silent Notification.

5. **Retorne para a tela Inicial clicando no botão Home (Início) no emulador.**

6. **Abra o aplicativo (com o botão central na parte inferior da tela).**

   Você verá o ícone de inicialização do aplicativo na lista de aplicativos.

Depois do emulador estar em execução, será executado por si só. O emulador não tem nenhuma dependência com o Eclipse. Na verdade, você pode fechar o Eclipse e ainda interagir com o emulador.

Instalando em um dispositivo Android físico

Instalar um aplicativo em um dispositivo não é diferente de instalar no emulador, exceto pela necessidade de fazer alguns pequenos ajustes para que funcione. Você provavelmente instalou o driver no Capítulo 2, portanto, as etapas restantes são simples:

1. **Na tela Inicial de seu telefone, acesse o painel Settings (Definições).**

2. **Em Security (Segurança) para os telefones mais recentes e Applications (Aplicativos) para os mais antigos, selecione a caixa de seleção Unknown Sources (Fontes Desconhecidas), como mostrado na Figura 5-6.**

   Selecione esta definição para instalar os aplicativos que não estão na Google Play Store.

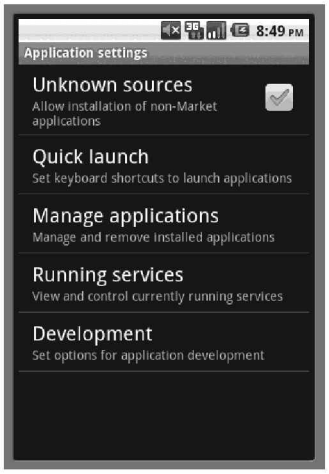


Figura 5-6: Esta configuração permite a instalação de aplicativos que não têm origem na Google Play Store.

3. **Escolha Development (Desenvolvimento) e selecione a opção USB Debugging (Depuração USB), como mostrado na Figura 5-7.**

   Esta etapa permite depurar seu aplicativo em um dispositivo (você poderá descobrir mais sobre a depuração posteriormente neste capítulo, na seção "Usando o depurador do Eclipse").

Figura 5-7: Ativando seu dispositivo para executar a depuração USB.

4. **Conecte seu telefone ao computador usando um cabo USB.**

5. **Quando o telefone for detectado em seu sistema, execute o aplicativo escolhendo Run (Executar)⇨Run ou pressionando Ctrl+F11.**

   O ADT reconhece outra opção de configuração de inicialização e pergunta em qual dispositivo você deseja executar o aplicativo, através da caixa de diálogo Android Device Chooser (Seletor de Dispositivos Android, mostrada na figura 5-8).

   O emulador não aparece na lista de opções disponíveis, a menos que esteja em execução.

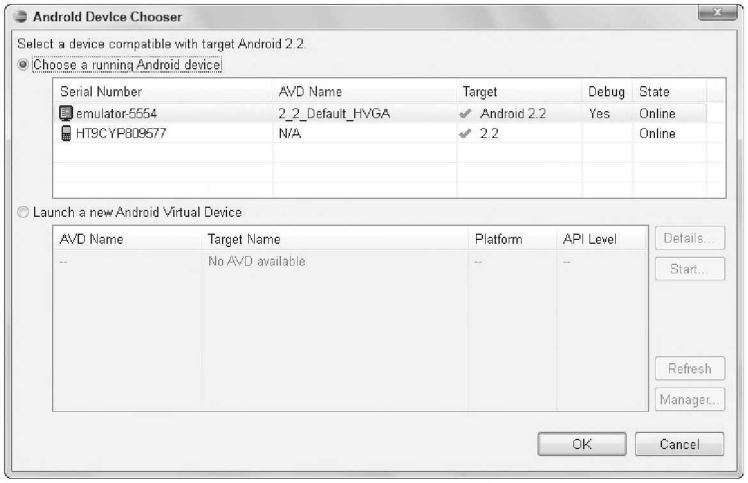


Figura 5-8: O seletor de dispositivos Android.

6. **Escolha seu telefone na lista e clique em OK.**

   Esta etapa envia o aplicativo para seu telefone e inicializa-o como inicializaria no emulador. Em alguns segundos, o aplicativo deverá aparecer em seu telefone.

   Agora, você implantou o aplicativo em seu telefone.

Se você mudar o aplicativo e precisar testá-lo novamente, terá que reinstalá-lo em seu telefone. Basta conectar seu telefone e selecionar Run (Executar)⇨Run ou pressionar Ctrl+F11.

Uh-Oh! (Respondendo aos Erros)

Você escreveu um código perfeito, certo? Mesmo que ele seja perfeito neste momento, chegará o dia em que não será. Quando a codificação não ocorre como o planejado, você tem que descobrir o problema. Para ajudar os desenvolvedores na terrível situação de uma paralisação aleatória do aplicativo, o ADT fornece ferramentas valiosas para ajudar a depurar os aplicativos.

Usando o servidor do monitor de depuração Dalvik

O Dalvik Debug Monitor Server (DDMS ou Servidor do Monitor de Depuração Dalvik) é uma ferramenta de depuração que fornece estes recursos, entre outros:

- Redirecionamento de portas
- Captura de telas
- Informações sobre thread e heap no dispositivo
- Mensagens de log do sistema via LogCat
- Informações do processo e do estado do rádio
- Spoofing de SMS e ligações
- Spoofing dos dados de localização

O DDMS, localizado no diretório `tools` da plataforma SDK do Android, pode trabalhar com um emulador e um dispositivo conectado. No Capítulo 2, você adicionou o diretório `tools` da plataforma ao seu caminho de diretórios, portanto, deverá ser capaz de acessar o DDMS a partir da linha de comando.

Por que você deve conhecer o DDMS

Depurar raramente é divertido. Felizmente, o DDMS fornece as ferramentas necessárias para ajudá-lo a consertar um programa cheio de erros. Um dos recursos mais usados no DDMS é o visor LogCat, que permite exibir a saída das mensagens de log do sistema no ambiente de desenvolvimento, como mostrado na Figura 5-9.

O registro de log informa tudo desde as mensagens de informações básicas (que incluem o estado do aplicativo e do dispositivo) até as informações de aviso e erro. Ver apenas uma mensagem de erro "Application Not Responding" (Aplicativo Não Responde) ou de fechamento forçado no dispositivo não esclarece o que está acontecendo. Abrir o DDMS e revisar as entradas no LogCat pode ajudar a identificar o número da linha onde a exceção está ocorrendo.

O DDMS não resolve o problema para você (droga!), mas pode tornar muito mais fácil rastrear a causa raiz do problema.

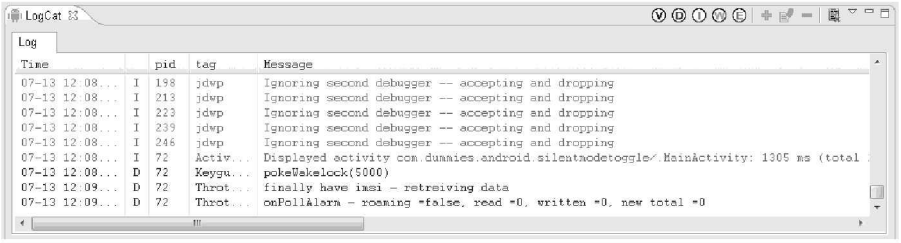


Figura 5-9: Uma exibição do LogCat.

O DDMS também é útil nos cenários onde você não tem nenhum dispositivo físico para testar. Por exemplo, se seu aplicativo for baseado no rastreamento de um usuário que está movendo-se em um mapa e o dispositivo do usuário não tem um GPS (ou o usuário não tem nenhum dispositivo), a tarefa será complicada. Felizmente, o DDMS está aqui para ajudar. Ele fornece ferramentas para controle de localização. Como desenvolvedor, você pode fornecer manualmente as coordenadas GPS ou um arquivo GPS eXchange Format (GPX ou Formato de Troca GPS) ou um arquivo Keyhole Markup Language (KML ou Linguagem de Marcação Obtida) que representem os pontos em um mapa que podem ser ajustados na hora. Por exemplo, você pode especificar se o usuário fica neste ponto por cinco segundos, vai para outro ponto, segue para o próximo, etc.

Exibindo registros de log no DDMS

Exibir registros de log no DDMS é tão simples quanto adicionar uma linha de código ao seu aplicativo. Abra o arquivo `MainActivity.java` e na parte inferior do método, adicione um registro de log, como mostrado em negrito na Listagem 5-5.

Listagem 5-5: O método onCreate( )

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

    mAudioManager = (AudioManager)getSystemService(AUDIO_SERVICE);

    checkIfPhoneIsSilent();

    setButtonClickListener();

    Log.d("SilentModeApp", "This is a test");                    →12
}
```

A linha 12 demonstra como enviar uma mensagem para o registro de log do sistema. `SilentModeApp` é conhecido como a `TAG` que você fornece a essa entrada de registro; o segundo parâmetro para a chamada de registro é a mensagem que você deseja enviar. A tag ajuda a filtrar as mensagens ao observá-las no DDMS.

Declare uma constante `TAG` em seu código e use-a, ao invés de digitar repetidamente a `TAG`, como neste exemplo:

```
private static final String TAG = "SilentModeApp";
```

Note o `d` em `Log.d` na Listagem 5-5, indicando que isto é uma mensagem de depuração. As outras opções são

- `e`: erro
- `i`: informações
- `wtf`: que falha terrível (sim, é uma opção.)
- `v`: explicação

Os vários tipos de registro existem para você decidir como as diversas mensagens devem ser registradas.

Para o registro de log funcionar, você tem que importar o pacote `android.util.Log`.

Exibindo as mensagens DDMS

Você pode exibir as mensagens DDMS abrindo o DDMS manualmente ou abrindo a perspectiva DDMS no Eclipse:

- **Manualmente:** Navegue para o lugar onde você instalou o SDK do Android. Dentro do diretório `tools`, clique duas vezes no arquivo ddms.bat. O aplicativo DDMS será aberto fora do IDE do Eclipse, como mostrado na Figura 5-10.
- **No Eclipse:** O ADT instalou a perspectiva DDMS. Para abri-la, clique no botão Open Perspective (Abrir Perspectiva) (veja Figura 5-11) e escolha DDMS.

  Se o DDMS não estiver visível nesta exibição, selecione a opção Other (Outro), então, selecione DDMS para adicionar uma perspectiva DDMS à lista de perspectivas que você poderá alternar facilmente.

  Se você preferir mover a janela LogCat (geralmente, próxima à parte inferior da tela) para a área principal da tela, como mostrado na Figura 5-12, simplesmente arraste o título da guia LogCat e solte-o no local desejado.

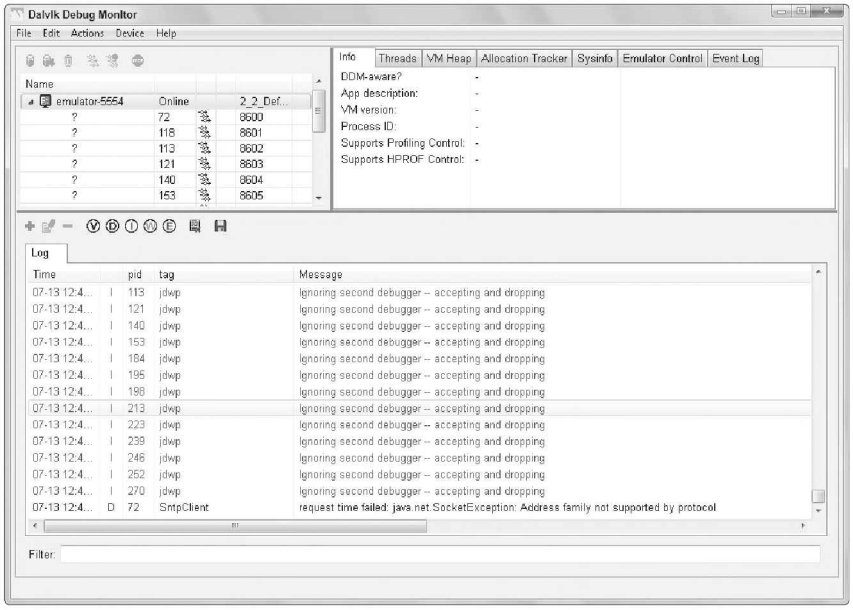


Figura 5-10: Uma instância do DDMS sendo executada separadamente do Eclipse.

Inicie o aplicativo selecionando Run (Executar)⇨Run ou pressionando Ctrl+F11. Quando seu aplicativo estiver sendo executado no emulador, abra a perspectiva DDMS e procure sua mensagem de log. Ela deverá parecer com a mostrada na Figura 5-13.

Agora, você pode voltar para a perspectiva Java clicando no botão Java Perspective (Perspectiva Java). (Consulte a Figura 5-11.)

Figura 5-11: O botão Open Perspective.

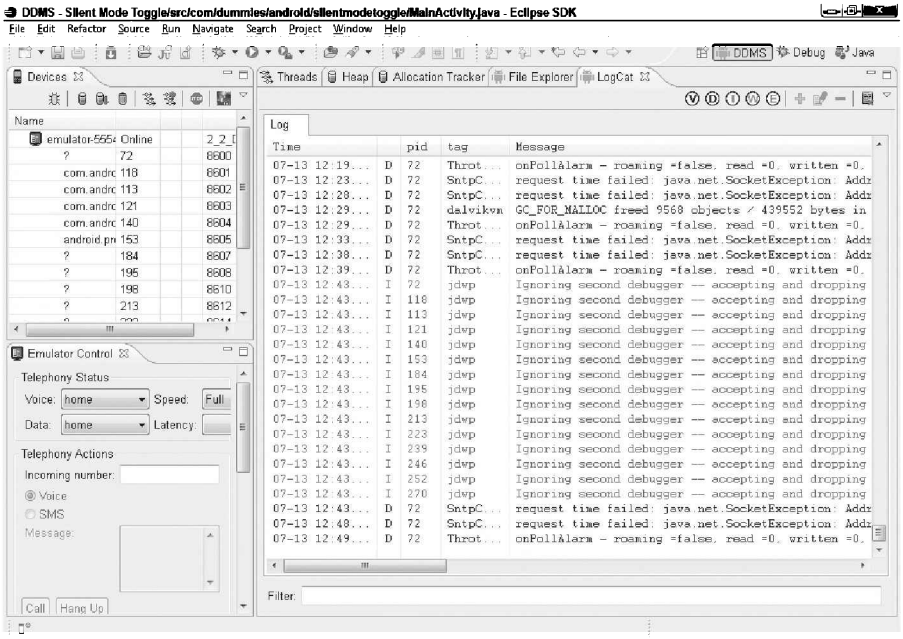
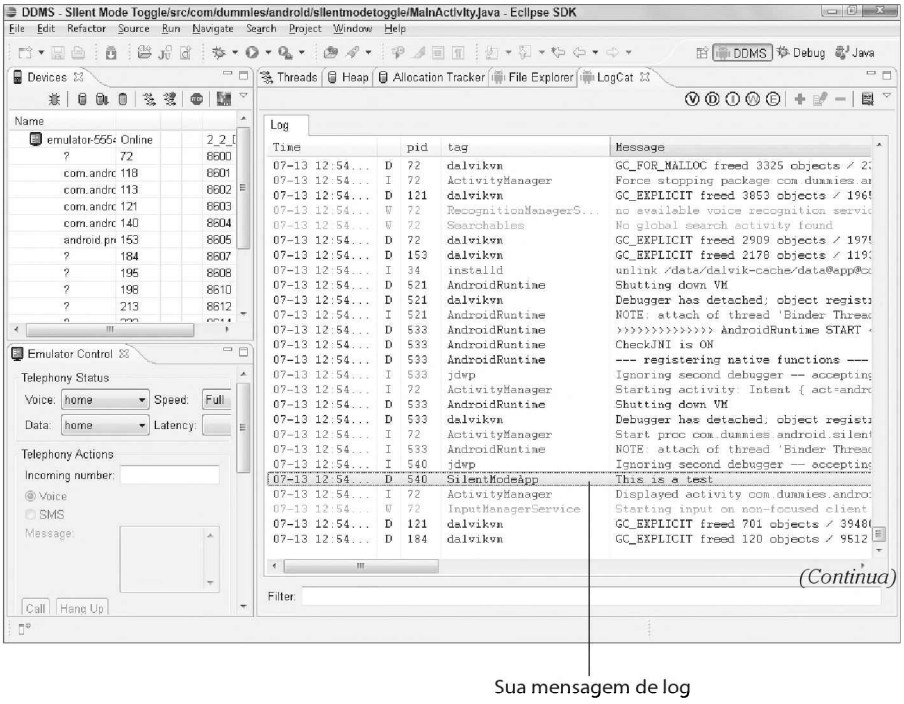


Figura 5-12: A janela LogCat na área de exibição principal do Eclipse.



(Continua)

Figura 5-13: Exibindo sua mensagem LogCat no Eclipse através da perspectiva DDMS.

Usando o depurador do Eclipse

Embora o DDMS possa ser um de seus melhores aliados, sua melhor arma para a batalha contra o exército de erros é o *depurador Eclipse*, que o permite definir vários pontos de interrupção, examinar as variáveis usando a janela de observação, exibir o LogCat e muito mais. Você usa o depurador para os erros de execução ou os erros lógicos.

O Eclipse descobre os erros na sintaxe. Quando o aplicativo não compila, o Eclipse alerta-o colocando uma linha colorida e pontilhada sob a área problemática.

Verificando os erros de execução

O *erro de execução* é a Bruxa Má do Leste — surge do nada e deixa tudo uma bagunça. No Android, os erros de execução ocorrem enquanto um aplicativo está em execução. Seu aplicativo pode estar cantarolando e, de repente, paralisa quando você clica em uma opção de menu ou em um botão, por exemplo. As possíveis razões para esse comportamento são inúmeras — talvez, você não tenha inicializado o `AudioManager` no método `onCreate()`, e então tentou acessar a variável posteriormente no aplicativo, o que causaria uma exceção de execução.

O depurador pode ajudar nesta situação porque você pode definir um ponto de interrupção no início de `onCreate()`, que o permite examinar os valores das variáveis através da perspectiva da depuração. Então, provavelmente você perceberia que esqueceu de inicializar `AlarmManager`.

A Listagem 5-6 demonstra o que criaria esta situação — comentar a inicialização `AlarmManager` faz com que uma exceção seja gerada na execução.

Listagem 5-6: Comentando a inicialização de AlarmManager

```
private AudioManager mAudioManager;                                  →1
private boolean mPhoneIsSilent;

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

    //mAudioManager =                                                →9
    //     (AudioManager)getSystemService(AUDIO_SERVICE);
```

(Continuação)

```
    checkIfPhoneIsSilent();

        setButtonClickListener();

        Log.d("SilentModeApp", "This is a test");
}
/**
 *  Verifica se o telefone está atualmente no modo silencioso.
 */
private void checkIfPhoneIsSilent() {
        int ringerMode = mAudioManager.getRingerMode();                    →22
        if (ringerMode == AudioManager.RINGER_MODE_SILENT) {
            mPhoneIsSilent = true;
        } else {
            mPhoneIsSilent = false;
        }

}
```

A Listagem 5-6 funciona assim:

→ **1** `AudioManager`, no nível da classe, é introduzido.

→ **9** Este código, que está comentado, deixa a variável `mAudioManager` em um estado nulo.

→ **22** Quando `onCreate()` chamou `checkIfPhoneIsSilent()`, o aplicativo gerou uma exceção de execução porque `mAudioManager` era nula e o aplicativo tentou referenciar um membro em um objeto que não existe.

Anexar um depurador ao método `onCreate()` permite controlar a principal causa do erro.

Criando pontos de interrupção

Existem algumas maneiras de criar um ponto de interrupção para pausar seu aplicativo no meio da execução e permitir que você examine seu estado de execução:

- Escolha a linha onde você deseja colocar o ponto de interrupção, clicando-a com o mouse. Escolha Run (Executar)⇨Toggle Breakpoint (Alternar Ponto de Interrupção), como mostrado na Figura 5-14, ou pressione Ctrl+Shift+B.
- Dê um duplo clique na margem esquerda no editor Eclipse, onde você deseja criar um ponto de interrupção.

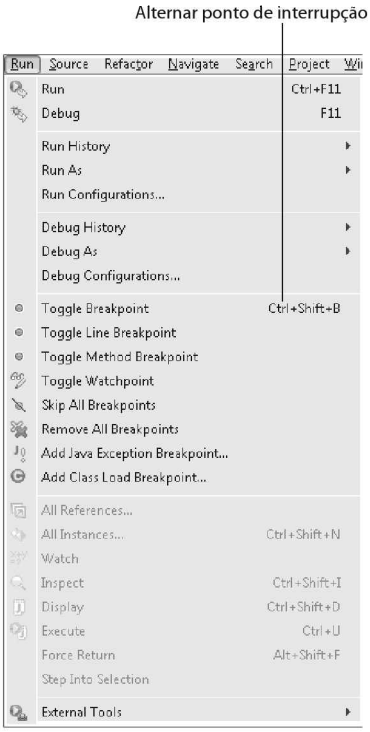


Figura 5-14: Definindo um ponto de interrupção usando um menu ou teclas de ativação.

Ambos os métodos criam um pequeno ícone redondo na margem esquerda do editor Eclipse, como mostrado na Figura 5-15.

Para tentar depurar no Eclipse, comente a linha 3 do método `onCreate()`, como mostrado na Listagem 5-7.

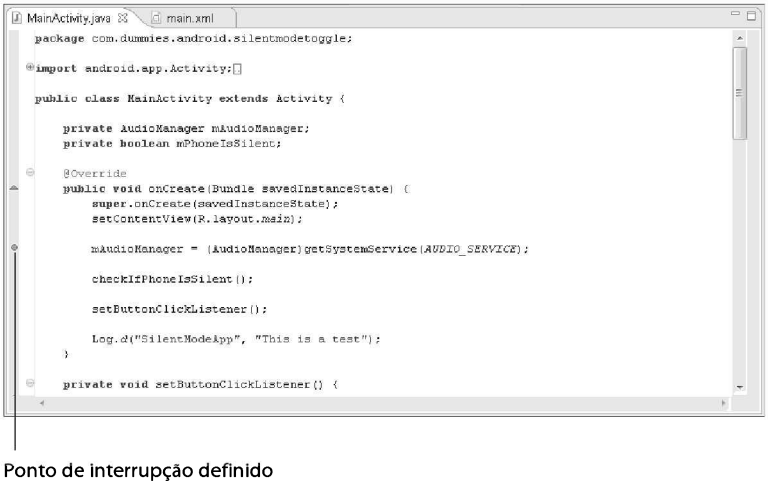
Listagem 5-7: Comentando o código para gerar um erro

```
setContentView(R.layout.activity_main);

//mAudioManager = (AudioManager)getSystemService(AUDIO_SERVICE);        →3

checkIfPhoneIsSilent();                                                  →5
```

→ **3** `AudioManager` é comentado.

→ **5** O método é chamado, fazendo com que o aplicativo falhe.

Defina um ponto de interrupção na linha 5.

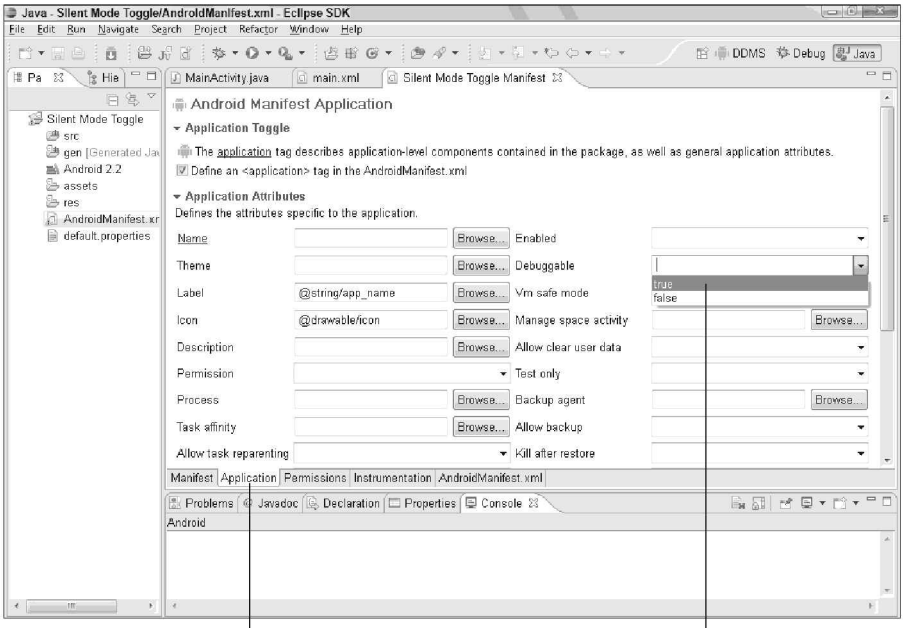


Ponto de interrupção definido

Figura 5-15: Um ponto de interrupção definido na margem esquerda da janela do editor do Eclipse.

Iniciando o depurador e a perspectiva Debug

Você tem uma tarefa a fazer antes de iniciar a depuração. Informe ao aplicativo Android que ele é depurável. Para tanto, abra o arquivo `AndroidManifest.xml`, selecione a guia Application (Aplicativo) na parte inferior (veja Figura 5-16), então, escolha a propriedade depurável (Debuggable) e defina-a para `true`, como mostrado na figura. Depois, salve o arquivo.



Guia Application

Escolha True

Figura 5-16: Configurando o aplicativo como depurável.

Não definir a propriedade depurável como `true` assegura que você nunca chegará a depurar seu aplicativo. Seu aplicativo nem mesmo tentará conectar-se ao depurador. Se você tiver problemas com a depuração, verifique se essa propriedade está definida como `true`.

Siga estas etapas para depurar seu código:

1. **Escolha Run (Executar)⇨Debug (Depurar) ou pressione F11.**

   O ADT e o Eclipse irão instalar o aplicativo no emulador (ou dispositivo), então, irão anexar o depurador a ele.

2. **Abra seu emulador.**

   O aplicativo é instalado e você verá a tela mostrada na Figura 5-17. Ela notifica que o ADT e o emulador estão tentando fazer uma conexão internamente.

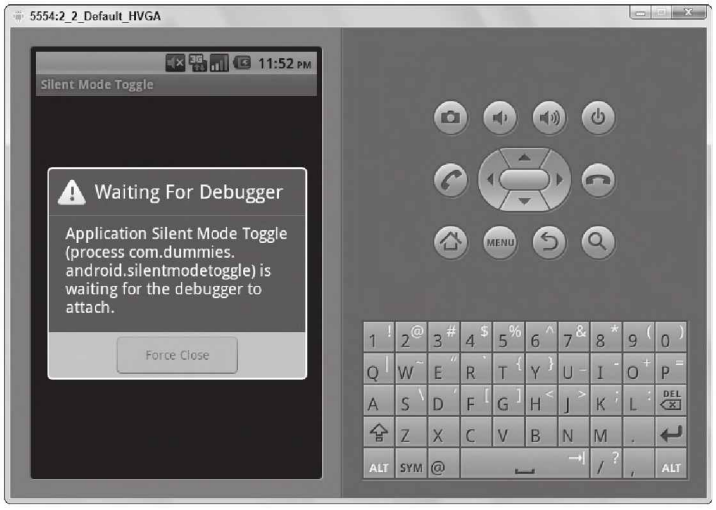


Figura 5-17: O emulador aguarda que o depurador seja anexado.

O emulador pode aguardar um momento enquanto e depurador é anexado. Então, o emulador executa o código de seu aplicativo e para quando encontra seu primeiro ponto de interrupção.

Depois, você vê uma caixa de diálogo perguntando se a perspectiva Debug (Depurar) pode ser aberta.

3. **Clique em Yes (Sim) para abrir a perspectiva Debug.**

   Agora, você está em um ponto de interrupção, como mostrado na Figura 5-18. Você pode passar o cursor sobre as variáveis para ver seus valores.

4. **Passe o cursor sobre a variável `mAudioManager`.**

   A variável é nula porque você comentou o código, como mostrado na Figura 5-18.

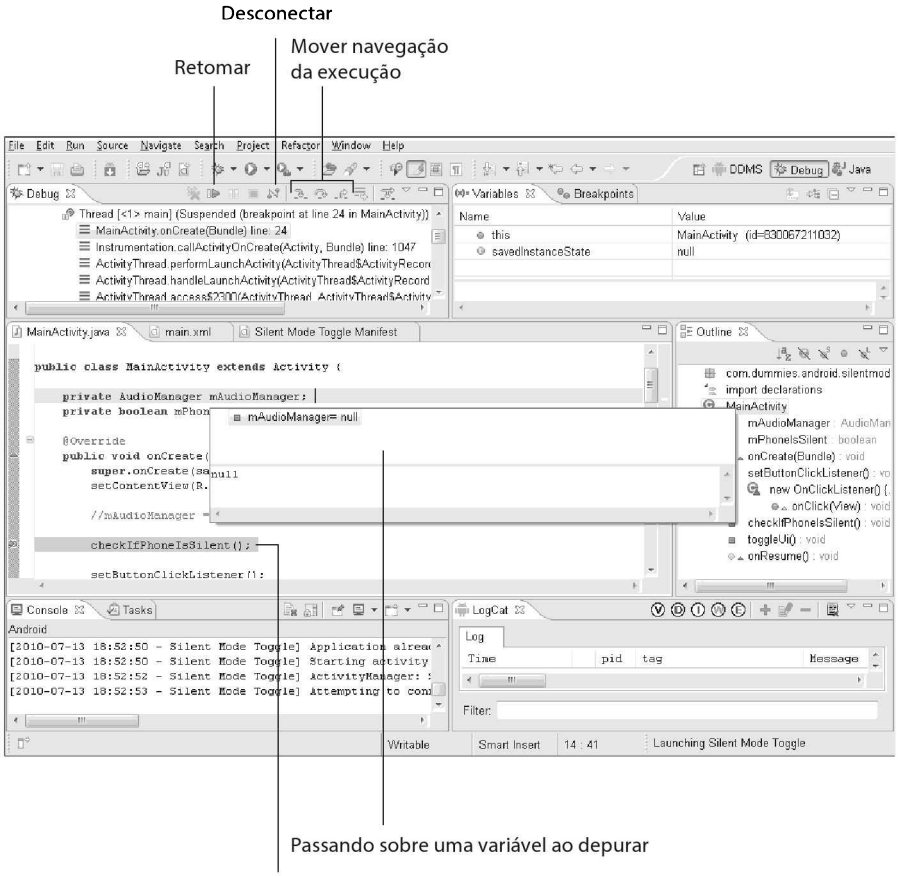


Figura 5-18: A perspectiva Debug é explicada.

Você também pode percorrer a execução do código operando a navegação da depuração, como mostrado na Figura 5-18. Se você clicar no botão Continue (Continuar) (ou pressionar F8), poderá ver a perspectiva Debug (Depurar) mudar e, finalmente, informar `source not found` (origem não encontrada). Abra o emulador e poderá ver que seu aplicativo paralisou, como mostrado na Figura 5-19. Na Google Play Store, os usuários conhecem essa tela como a tela para fechar à força (force-close ou FC). Um fechamento à força ocorre quando uma exceção de execução não é tratada dentro de seu código.

Figura 5-19: A caixa de diálogo para fechar à força é exibida após uma exceção na execução.

5. **Para desconectar o depurador, clique no botão Disconnect (Desconectar).**

Retorne para a perspectiva Java e retire o comentário da linha 3 na Listagem 5-7 no arquivo `MainActivity.java` para assegurar que o aplicativo seja construído com sucesso.

Verificando os erros lógicos

Os computadores fazem exatamente o que são informados para fazer e este pequeno smartphone não é inteligente o suficiente para entender o que é certo ou errado na lógica literal. Um exemplo de erro de lógica é demonstrado na Listagem 5-8.

Listagem 5-8: O código que não verifica o telefone quanto ao modo silencioso

```
/**
 * Alterna as imagens IU de silenciosa
 * para normal e vice-versa.
 */
private void toggleUi() {

    ImageView imageView =
         (ImageView) findViewById(R.id.phone_icon);
            Drawable newPhoneImage;
```

(continua)

Listagem 5-8: *(continuação)*

```
        if (mPhoneIsSilent) {                                    →11
                newPhoneImage =
                    getResources().getDrawable(R.drawable.phone_silent);

        } else {
            newPhoneImage =
                getResources().getDrawable(R.drawable.phone_on);
        }

        imageView.setImageDrawable(newPhoneImage);
}

@Override
protected void onResume() {
    super.onResume();
    //checkIfPhoneIsSilent();                                    →26
    toggleUi();
};
```

A Listagem 5-8 funciona assim:

→ **11** Esta linha verifica se o telefone está no modo Silent (Silencioso).

→ **26** Para o método `toggleUi()` exibir devidamente a interface correta para o usuário, o aplicativo tem que conhecer em qual estado está a campainha. Esta linha comenta o método `checkIfPhoneIsSilent()`, que atualiza a variável de nível de classe `mPhoneIsSilentVariable`.

Como isto ocorre no método `onResume()`, o usuário pode deixar o aplicativo, alterar o estado de sua campainha via aplicativo Settings (Definições) do telefone, e então voltar para o aplicativo — e o aplicativo estaria em um estado incorreto simplesmente por causa de um erro lógico.

Usando um depurador, você pode anexar um ponto de interrupção na primeira linha do método `toggleUi()` para examinar as diversas variáveis que ajudam a fazer as chamadas lógicas. Então, você notaria que `mPhoneIsSilent` não está sendo definida.

Pensando Além dos Limites do Aplicativo

Às vezes, o dispositivo pode fazer um trabalho extra que afeta seu aplicativo, como por exemplo, fazer download de um grande arquivo em segundo plano enquanto reproduz música a partir de um aplicativo de rádio on-line. Estas atividades pesadas e consumidoras de rede afetam o aplicativo? Depende. Se seu aplicativo precisar de uma conexão com a Internet e por algum motivo não conseguir conectar-se, irá paralisar? O que acontecerá? Saber as respostas para essas perguntas significa que você está pensando além dos limites de seu aplicativo.

Nem todos os aplicativos são criados igualmente — alguns bons existem por aí, junto com outros *ruins*. Antes de construir ou lançar seu primeiro aplicativo Android, saiba os prós e os contras de seu aplicativo, e qualquer coisa que possa afetá-lo. Certifique-se de que seu aplicativo não paralise quando os usuários executarem eventos rotineiros de toque e navegação da tela.

Construir aplicativos em dispositivos embutidos é muito diferente de construí-los em um PC ou Mac, e a razão é simples: os recursos (memória e processador, por exemplo) são limitados. Se o dispositivo Android for um telefone, sua principal finalidade será executar tarefas como um telefone, tais como reconhecer uma chamada de entrada, manter o sinal de conexão, enviar e receber mensagens de texto.

Se uma chamada de telefone estiver em andamento, o sistema Android tratará esse processo como sendo vital, ao passo que um arquivo sendo baixado em segundo plano será considerado não vital. Se o telefone começar a ficar sem recursos, o Android irá encerrar os processos não vitais para manter os vitais ativos. Um arquivo pode ser baixado novamente, mas quando uma chamada é perdida, está perdida para sempre — você tem que fazer a chamada novamente, o que frustraria o usuário se a principal finalidade para comprar o dispositivo foi ter um telefone. Seu aplicativo pode fazer o download de um arquivo em segundo plano e o processo ser encerrado — esta é uma situação que você precisa testar. Também poderá acontecer se seu telefone encontrar uma área com um sinal sem fio ruim ou inexistente. Se a conexão cair, seu arquivo não será baixado.

Teste todas as possíveis soluções e dê um tratamento adequado a elas. Do contrário, seu aplicativo terá exceções de execução, que podem levar a avaliações ruins dos usuários na Google Play Store.

E o teste automático?

Com a ascensão das metodologias ágeis na última década, é apenas uma questão de tempo antes de você começar a imaginar como realizar um teste automático no Android. O SDK instala as ferramentas de teste da unidade Android que você poderá usar para testar não apenas as classes Java, mas também as classes baseadas no Android e as interações de interface do usuário. Você pode ler mais sobre o teste da unidade na documentação Android em `http://d.android.com/guide/topics/testing/testing_android.html` (conteúdo em inglês).

Eis as ferramentas à sua disposição:

- **jUnit:** O SDK instala a integração jUnit com o ADT. Você pode usar o jUnit, um framework popular para testes unitários que é usada no Java para realizar os testes unitários e os testes de integração. Você pode encontrar mais informações sobre o jUnit em `www.junit.org`. Para facilitar seu desenvolvimento, o Eclipse tem ferramentas predefinidas para facilitar o teste no jUnit através do Eclipse.
- **Monkey:** O testador de interface de usuário e de aplicativo, conhecido como Monkey, executado em seu emulador ou dispositivo, gera fluxos pseudoaleatórios de eventos do usuário, incluindo toques, gestos, cliques e vários eventos do sistema. O Monkey, que é instalado com o SDK do Android, é um modo útil de testar intensamente um aplicativo.

Interagindo com seu aplicativo

Para assegurar que seu aplicativo funciona, inicialize e utilize seus recursos. Enquanto seu aplicativo estiver em execução, inicie outro aplicativo, como por exemplo, um navegador. Visite alguns sites, e então volte para seu aplicativo. Clique em qualquer botão relacionado ao seu aplicativo para ver o que acontece. Tente todos os tipos de ações para saber se ocorrem resultados que não considerou. O que acontecerá se um usuário estiver interagindo com seu aplicativo e receber uma chamada telefônica? Você está salvando o estado necessário em `onPause()` e restaurando-o em `onResume()`?

O Android lida com o difícil gerenciamento de tarefas para você, mas é sua responsabilidade gerenciar o estado de seu aplicativo.

Testando se seu aplicativo funciona

No emulador, abra o aplicativo Silent Mode Toggle na inicialização. Você já realizou a primeira etapa no processo de teste — certificou-se de que o aplicativo possa ser iniciado!

Após abrir o aplicativo, verifique se o telefone está no modo Silent (Silencioso), procurando o pequeno ícone do telefone na barra de notificação (consulte a Figura 5-5).

Clique no botão Toggle Silent Mode (Alternar Modo Silencioso) para alternar o modo da campainha. A imagem do aplicativo mudou de telefone verde para telefone silencioso (ou vice-versa)? Tente várias ações para assegurar que seu aplicativo funciona como o esperado. Se encontrar uma falha, use as ferramentas de depuração encontradas neste capítulo para identificar o problema.

Capítulo 6

Compreendendo os Recursos Android

Neste Capítulo

Entendendo por que os recursos são importantes no Android

Extraindo recursos

Trabalhando com recursos de imagem

Os recursos são mencionados em detalhes neste livro, portanto, você pode imaginar por que um capítulo inteiro é dedicado a eles. Analisar os recursos e seu uso nos Capítulos 3 e 4 foi necessário para ajudar a entender a estrutura básica do diretório de recursos e o uso dos recursos para construir um aplicativo simples. Um motivo convincente para usar recursos em seu aplicativo, a globalização, é tratado neste capítulo.

Compreendendo os Recursos

Recursos são o conteúdo estático adicional e os arquivos que são parte intrínseca de seu aplicativo, mas não fazem parte de seu código Java. Os recursos podem ter estas formas:

- Layout
- String
- Imagem
- Dimensão
- Estilo
- Tema
- Valor
- Menu
- Cor

Os capítulos anteriores neste livro apresentaram os layouts, strings e imagens porque são os tipos mais comuns de recursos que você usa em todo desenvolvimento de aplicativos Android. Os recursos restantes podem ser confusos, portanto, as seguintes seções irão clareá-los.

Dimensões

Em um recurso Android, uma *dimensão* é um número seguido de uma unidade de medida, tal como 10 px, 2 in (pol.) ou 5 sp. Você usa uma dimensão para especificar qualquer propriedade no Android que necessite uma unidade numérica de medida. Por exemplo, você pode querer que o preenchimento de um layout tenha 10 px. As seguintes unidades de medida são suportadas pelo Android:

- **pixel independente da densidade (dp):** Esta unidade abstrata é baseada na densidade física da tela. Essas unidades são relativas a uma medida de tela de 160 pontos por polegada (dpi); portanto, 1 dp é equivalente a 1 pixel em uma tela com 160 dpi. A proporção de dp para pixels muda com a densidade da tela, mas não necessariamente em proporção. É a unidade de medida que a maioria dos desenvolvedores usa ao desenvolver os layouts.

  O conceito dp é complexo. Se você pretende suportar diversas densidades de tela, o artigo Supporting Multiple Screen Sizes (Suportando Diversos Tamanhos de Tela) em `http://developer.android.com/guide/practices/screens_support.html` (conteúdo em inglês) é uma leitura obrigatória.

- **pixel independente da escala (sp):** Esta unidade lembra a unidade dp, mas é dimensionada de acordo com a preferência do tamanho de fonte do usuário. Use as dimensões sp ao especificar os tamanhos da fonte em seu aplicativo.
- **pixel (px):** Um pixel corresponde a um pixel na tela. Esta unidade de medida não é recomendada. Seu aplicativo pode parecer ótimo em um dispositivo com densidade média, mas ficará distorcido e inadequado em uma tela com alta densidade (e vice-versa) porque o dpi difere.
- **ponto (pt):** Um ponto é `1/72` de polegada, com base no tamanho físico da tela. Assim como px, não é recomendado.
- **milímetro (mm):** Esta unidade é baseada no tamanho da tela. Assim como px, não é recomendado.
- **polegada (in):** Esta unidade é baseada no tamanho físico da tela. Assim como px, não é recomendado.

Estilos

Os estilos no Android são parecidos com as Folhas de Estilo em Cascata (CSS) no domínio do desenvolvimento web: Você usa os estilos para (adivinhe) aplicar estilo em um aplicativo. Um *estilo* é uma coleção de propriedades que pode ser aplicada em uma exibição individual (no arquivo de layout), em uma atividade ou em seu aplicativo inteiro (de dentro do arquivo manifest). Os estilos suportam herança, portanto, você pode fornecer um estilo básico, e então modificá-lo para cada caso de uso particular em seu aplicativo. Os exemplos de propriedade do estilo incluem o tamanho da fonte, cor da fonte e do fundo da tela.

Temas

Um *tema* é um estilo aplicado a uma atividade ou aplicativo inteiro, ao invés de uma exibição individual. Quando um estilo é aplicado como um tema, todas as exibições na atividade e/ou aplicativo herdam as definições do estilo. Por exemplo, você pode definir todas as exibições `TextView` para uma determinada fonte e todas as exibições na atividade ou no aplicativo usando este tema mostrarão o texto nessa fonte.

Valores

O recurso de valor pode conter muitos subtipos diferentes de recursos para seu aplicativo, inclusive

- **Booleano:** Um valor booleano definido no XML cujo valor é armazenado em um nome de arquivo arbitrário no arquivo `res/values/<nomearquivo>.xml`, onde *<nomearquivo>* é o nome do arquivo. Um exemplo é `bools.xml`.
- **Inteiro:** Um valor inteiro definido no XML cujo valor é armazenado com um nome de arquivo arbitrário no arquivo `res/values/<nomearquivo>.xml`. Um exemplo é `integers.xml`.
- **Array de inteiros:** Um array de inteiros definido no XML cujo conjunto de valores é armazenado com um nome arbitrário no arquivo `res/values/<nomearquivo>.xml`, onde *<nomearquivo>* é o nome do arquivo. Um exemplo é `integers.xml`. Você pode referenciar e usar esses inteiros em seu código para ajudar a definir os loops, comprimentos e outros elementos.
- **Array tipado:** Um *array tipado* é usado para criar um array de recursos, tal como `drawables`. Você pode criar arrays com tipos misturados. Portanto, os arrays não precisam ser homogêneos — contudo, você deve conhecer o tipo de dado para que possa fazer a conversão dele devidamente. Como em outros recursos, o nome de arquivo é arbitrário no arquivo `res/values/<nomearquivo>.xml`. Um exemplo é `types.xml`.

Menus

Esteja seu aplicativo usando a barra de ação ou um menu, o Android irá tratá-los igualmente e você irá defini-los do mesmo modo. Um menu pode ser definido via código ou XML. O modo mais adequado para definir um é via XML, portanto, os seus menus devem ser colocados no diretório `menu/`. Cada menu tem seu próprio arquivo `.xml`.

Cores

O arquivo `colors`, localizado no arquivo `values/colors.xml`, permite nomear as cores, tais como `login_screen_font_color`. Isto pode representar a cor da fonte que você está usando na página de logon, por exemplo. Cada cor é definida como um valor hexadecimal.

Trabalhando com Recursos

Você deve ter trabalhado com recursos algumas vezes neste livro e provavelmente está familiarizado com o uso da classe `R` para acessar os recursos de dentro de seu aplicativo. Se você estiver inseguro em relação aos recursos e ao arquivo `R`, veja o Capítulo 3.

Movendo strings para recursos

Quando você for um programador experiente, poderá começar a pegar atalhos para ter seu projeto construído e funcionando. Digamos que você tenha esquecido inicialmente de mover as strings para os recursos e tenha que voltar depois para fazer isso. Você poderá extrair uma string para um recurso usando ferramentas predefinidas.

Modo longo

Eis um modo de extrair uma string para um recurso:

1. **Crie um novo recurso de string.**
2. **Copie seu nome.**
3. **Substitua o valor da string em seu layout pelo identificador do recurso.**

Esta tarefa pode não ser um grande problema, mas leva tempo, possivelmente de 30 a 45 segundos para o desenvolvedor normal.

Modo rápido

Você pode reduzir o tempo para criar um recurso de string para menos de 15 segundos. Se você fizer isso 30 vezes por dia (o que é possível em um dia de 8 horas), poderá economizar 15 minutos apenas copiando e colando. São cinco horas por mês fazendo a cópia e a cola!

Siga estas etapas:

1. **No Eclipse, abra o arquivo** `main.xml` **no diretório** `layouts`**.**

2. **Encontre o seguinte bloco de código no arquivo:**

```
<Button
          android:id="@+id/toggleButton"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_gravity="center_horizontal"
          android:text="Toggle Silent Mode"
/>
```

3. **Selecione a linha em negrito** `"Toggle Silent Mode"`**.**

4. **Pressione Ctrl+1 no Windows ou ⌘+1 em um Mac.**

   Um menu será aberto com três opções.

5. **Escolha a opção Extract Android String (Extrair String Android).**

   A caixa de diálogo Extract Android String será aberta, como mostrado na Figura 6-1 e você poderá definir várias opções para o recurso.

6. **Deixe o padrão como está e clique em OK.**

   Agora, você pode ver que o arquivo de layout foi modificado. O texto `"Toggle Silent Mode"` foi substituído por `"@string/toggle_silent_mode"`.

   Se você abrir o arquivo `string.xml` na pasta `res/value`, poderá ver um novo recurso de string com esse nome e o valor `"Toggle Silent Mode"`.

Isso é muito interessante! Você pode ver que fazer isto 20 ou 30 vezes por dia pode fazer sentido e economizar muito tempo.

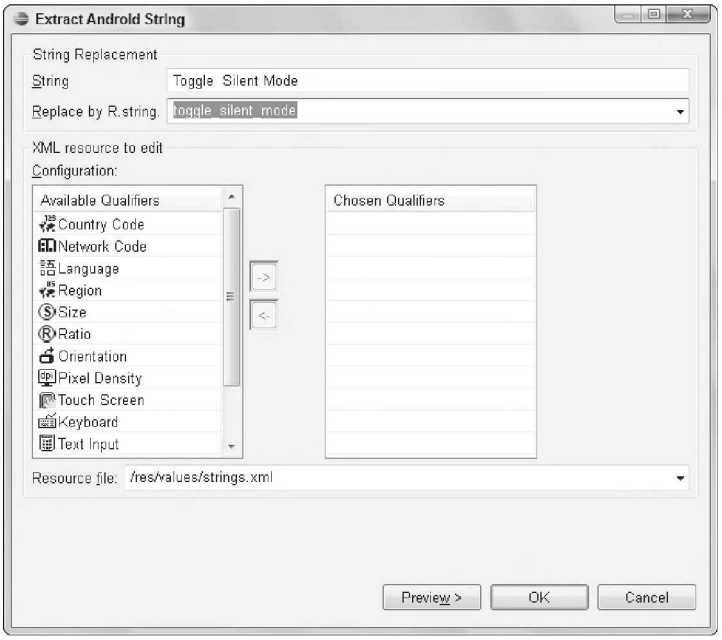


Figura 6-1: A caixa de diálogo Extract Android String.

Lutando com a imagem

Uma das partes mais difíceis sobre os recursos pode ser as imagens. Elas podem ficar ótimas em dispositivos com densidade média, mas ficar um lixo nos dispositivos com alta densidade. É onde entra em cena as pastas com diversas densidades. Essas pastas de desenho com densidades específicas foram explicadas no Capítulo 3.

Lutando com o pixel e a compressão

O problema que muito provavelmente você encontrará é o borrado e a compressão/expansão (ir dos dispositivos com densidade mais alta para a mais baixa, e vice-versa). Para resolver o problema, projete seu gráfico com uma resolução alta, tal como 300 dpi no formato grande. Por exemplo, se você estiver construindo o ícone de inicialização, construa-o com 250 px de altura e 250 px de largura. Embora a pasta `xhdpi` possa precisar de uma imagem com apenas 96 px de altura por 96 px de largura (a maior em uso), não significa que em dois ou três meses um dispositivo com resolução mais alta não será lançado.

Esta situação pode ser um problema porque trabalhar com arquivos de imagem grandes em programas de edição de imagens poderá ser difícil se você não tiver um computador com bom desempenho. Mas, confie em mim: ter um arquivo grande de imagem bruta com densidade alta é muito mais fácil do que moldar e modelar nas densidades corretas.

Reduzir uma imagem com densidade alta não distorce sua qualidade (exceto pela perda das bordas finas e os detalhes), mas aumentar distorce-a porque cria borrados e distorção. Iniciar com um arquivo grande reduz a chance de você ter que aumentar a escala, significando que seu aplicativo gráfico sempre parecerá nítido.

Usando camadas

Se você estiver criando gráficos em uma ferramenta de edição de imagens que suporta camadas, coloque cada item em seu gráfico em uma camada diferente. Os motivos são muitos, mas eis alguns fatores principais:

- **Alterações:** Em algum momento, você precisará mudar algo em seu gráfico — seu fundo, fonte ou logotipo, por exemplo. Se você tiver todos esses itens em camadas diferentes, poderá fazer a alteração sem afetar o resto do gráfico.
- **Localização:** Um exemplo de uma seção anterior neste capítulo fala sobre várias strings em diferentes idiomas e os gráficos não são diferentes. Muitas vezes quando você desenvolver aplicativos, encontrará gráficos com texto estilizado. Se seu aplicativo estiver sendo traduzido para o japonês e seu gráfico contiver texto em português estilizado, você poderá criar uma versão japonesa desses gráficos e colocá-los em uma pasta de desenho da região japonesa, tal como `res/drawable-ja`. A plataforma Android reconhecerá em qual região está (neste caso, Japão). Se as pastas de recursos da região (`res/drawable-ja, res/values-ja` etc.) estiverem disponíveis, o Android irá usá-las no aplicativo. Dito isso, sempre é mais fácil manter seu texto nos recursos de texto e suas imagens nos recursos de imagem. Traduzir os recursos de texto é mais fácil do que fazer novas cópias de suas imagens para cada novo idioma.

Tornando seus aplicativos globais com recursos

A plataforma Android ultrapassou o Apple iPhone na participação do mercado americano no primeiro trimestre de 2010. Agora, as portadoras em todo o mundo estão desenvolvendo smartphones baseados no Android, e isso significa mais usuários em potencial para seus aplicativos.

O que esta afirmação significa para você, como desenvolvedor, é que o Android é um mercado enorme, com milhares de oportunidades esperando para serem aproveitadas. Embora essa oportunidade seja animadora, obter esta vantagem requer que você entenda os recursos e como eles afetam a utilização de seus aplicativos. Por exemplo, se um usuário nos Estados Unidos usar seu aplicativo que foi escrito para um público falante do inglês (usando

recursos ou não), o usuário será capaz de utilizá-lo. Contudo, se você codificar rigidamente todos os valores de tipo string de suas exibições e atividades e precisar lançar uma versão em chinês, terá que reescrever seu aplicativo para usar os recursos. Quando você usa recursos, pode ter uma tradução linguística de suas strings e desenhos para a região alvo — como, por exemplo, a China.

Os recursos permitem extrair strings legíveis pelos humanos, imagens e layouts para recursos que você pode referenciar. Várias pastas de recursos podem ser criadas para lidar com telas de vários tamanhos, diferentes idiomas (strings e desenhos) e opções de layout, tais como paisagem e retrato. As exibições de paisagem e retrato entram em cena quando um usuário gira o dispositivo em 90 graus, em qualquer direção.

Se você quiser que seus aplicativos sejam visíveis para o maior número possível de dispositivos Android em todo o mundo, deverá usar recursos sempre. Sempre coloque todas as strings no arquivo `strings.xml` porque, algum dia, alguém de outro país desejará seu aplicativo em outro idioma. Para transportar seu aplicativo para outro idioma, você simplesmente precisará ter uma tradução linguística de seu arquivo `strings.xml` para o outro idioma, e então, poderá criar várias pastas `values` para manter os valores da devida região. O Android cuida do trabalho pesado. Por exemplo, se o usuário estiver na China e seu telefone estiver definido para o conjunto de caracteres chinês, o Android irá procurar a pasta de valores denominada `values-cn`, que é onde estão armazenados os valores chineses — inclusive a versão chinesa do arquivo `strings.xml`. Se o Android não puder encontrar tal pasta, a plataforma terá como padrão a pasta `values` padrão, que contém a versão em inglês do arquivo `strings.xml` (para saber mais sobre as strings, veja a seção "Movendo strings para recursos", anteriormente neste capítulo).

Em relação a isso, providenciar uma atualização linguística de suas strings e criar uma nova pasta com o novo arquivo `strings.xml` são tarefas simples. Expanda este conceito para outros idiomas, tablets, televisões e poderá ver o potencial. Você não mais verá somente os usuários móveis como seu público-alvo. Estará vendo usuários *Android*, e com as opções sendo lançadas, poderá estar vendo *bilhões* de usuários. Usar os recursos corretamente pode facilitar muito a sua expansão pelos mercados estrangeiros.

Embora nada supere ter uma pessoa para traduzir as strings do aplicativo para você, encontrar um falante nativo para todos os idiomas que você deseja suportar pode ser difícil. Há um atalho que você pode tomar: deixar que o Google faça a tradução para você. Visite `http://translate.google.com/toolkit` e faça o upload de seu arquivo `strings.xml` para que ele seja traduzido automaticamente por um computador. Os resultados podem parecer um pouco desajeitados para um falante nativo, mas pelo menos lhe dará uma vantagem inicial.

Projetar seu aplicativo para várias regiões é um tópico extenso. Você pode encontrar informações mais detalhadas no artigo "Localization" (Localização) da documentação SSK em `http://developer.android.com/guide/topics/resources/localization.html` (conteúdo em inglês).

Embora projetar um aplicativo para várias regiões pareça necessário, também ajuda saber que a Google Play Store permite especificar a região alvo de seu dispositivo. Você não é obrigado a lançar seu aplicativo em todas as regiões. Portanto, se você escreveu um aplicativo para o sistema de ônibus de Berlim, provavelmente não fará sentido ter uma versão chinesa, a menos que você queira cuidar dos turistas chineses, assim como dos residentes alemães.

Capítulo 7

Transformando seu Aplicativo em um Componente da Tela Inicial

Neste Capítulo

Vendo como os componentes (widget) de aplicativo trabalham no Android

Entendendo as intenções pendentes

Construindo um App Widget Provider

Colocando seu componente na tela Inicial

Usabilidade é o ponto principal em relação a todas as disciplinas do desenvolvimento de aplicativos: Se seu aplicativo não for utilizável, os usuário simplesmente não irão usá-lo.

Se você acompanhou os seis primeiros capítulo deste livro para construir o aplicativo Silent Mode Toggle, viu que certamente ele funciona bem e tem boa usabilidade. Infelizmente, se o aplicativo fosse publicado na Google Play Store, provavelmente não seria popular porque um usuário teria que abrir o aplicativo e tocar em um botão para silenciar o telefone. Se o usuário não criou um atalho de tela Inicial para o aplicativo e ele estiver oculto no inicializador junto com os outros aplicativos, algumas etapas extras serão necessárias: Desbloquear o telefone, abrir a inicialização, localizar o aplicativo, abrir o aplicativo, e então tocar no botão Silent (Silenciar). Neste ponto, o usuário pode também pressionar as teclas Volume Up (Aumentar Volume) e Volume Down (Diminuir Volume) para silenciar o telefone. Para aumentar a usabilidade e praticidade desse aplicativo, simplesmente transforme-o em um componente de tela Inicial (home screen widget).

Neste capítulo, você construirá um componente de tela Inicial para seu aplicativo. Normalmente, um componente de aplicativo é um pequeno ícone ou exibição minúscula na tela Inicial. Os usuários podem interagir com seu aplicativo simplesmente tocando em seu ícone — o componente de tela Inicial. Então, a funcionalidade básica será iniciada e alternará o modo silencioso. Este capítulo apresenta estas classes:

- `Intent`
- `BroadcastReceiver`
- `AppWidgetProvider`
- `IntentService`

Cada uma dessas classes desempenha um papel vital no Android, assim como na estrutura do componente de aplicativo.

Trabalhando com os Componentes de Aplicativo no Android

Um *componente de tela Inicial* (ou *componente de aplicativo*) no Android é um tipo especial de exibição que pode ser incorporada na tela Inicial de seu dispositivo. Um componente de aplicativo pode aceitar a entrada do usuário via eventos de clique e pode atualizar-se regularmente. Um usuário pode adicionar um componente de aplicativo à tela Inicial tocando no botão Applications (Aplicativos), então, selecionando Widgets (Componentes). O resultado é mostrado na Figura 7-1.

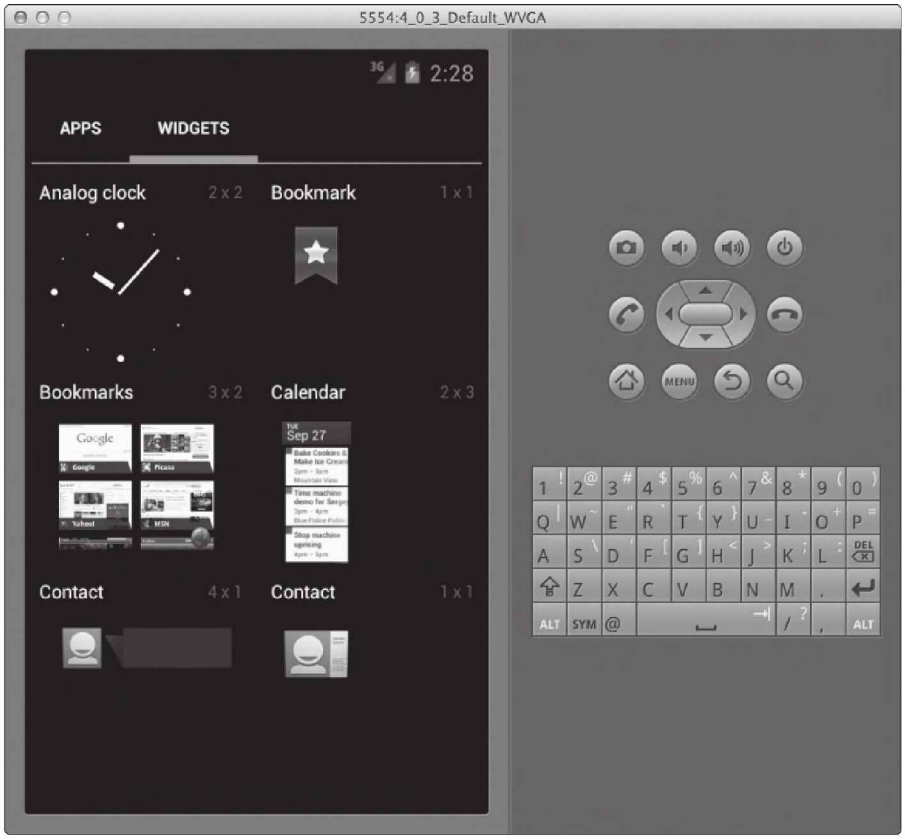


Figura 7-1: Adicionando um componente à tela Inicial.

Para tornar mais útil o aplicativo Silent Mode Toggle, construa um componente de tela Inicial para que os usuários possam usá-lo na tela Inicial. Tocar no componente mudará automaticamente o modo da campainha do telefone sem ter que abrir o aplicativo. O componente também atualiza seu layout para indicar em qual estado está o telefone, como mostrado na Figura 7-2.

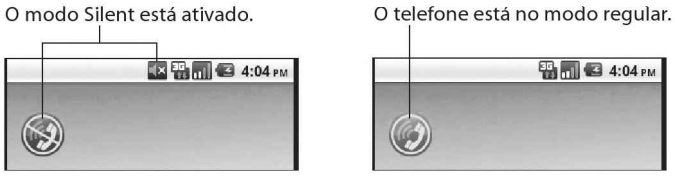


Figura 7-2: Os dois estados do componente de aplicativo.

Trabalhando com exibições remotas

Quando você desenvolver aplicativos no Android, lembre-se que ele é baseado no kernel Linux 2.6. O Linux vem com seus próprios idiomas (ou "dialetos") sobre segurança e a plataforma Android herda-os. Por exemplo, o modelo de segurança Android é muito baseado no modelo de segurança de usuários, arquivos e processos do Linux.

Como todo aplicativo Android é (geralmente) associado a seu próprio usuário exclusivo, o Android impede que os aplicativos modifiquem os arquivos de outros aplicativos. Isto impede que os desenvolvedores introduzam um código malicioso em outros aplicativos.

Como a tela Inicial é também um aplicativo e por isso tem seu próprio usuário exclusivo, os desenvolvedores não têm permissão de executar diretamente o código de seus aplicativos na tela Inicial por questões de segurança. Para fornecer um modo de acessar a tela Inicial e modificar o conteúdo de uma determinada área nela a partir de um aplicativo, os desenvolvedores Android implementaram a arquitetura `RemoteViews`: Ela permite executar o código de dentro de seu aplicativo, em um processo separado do aplicativo da tela Inicial, mas que permite que a exibição de um componente seja atualizada na tela Inicial. O resultado é que você ainda pode ter seu componente, mas nenhum código arbitrário precisa ser executado dentro do aplicativo de tela Inicial — todo o código do componente de seu aplicativo é executado dentro de seu aplicativo.

Suponha que um usuário toque no widget de aplicativo de tela Inicial (neste caso, um ícone adicionado à tela Inicial). Esta ação envia uma solicitação — endereçada ao *seu* aplicativo — para mudar o modo da campainha. O Android roteia a solicitação para seu aplicativo e o aplicativo processa a solicitação, instruindo a plataforma Android para mudar o modo da campainha e atualizar o componente de aplicativo de tela Inicial com uma nova imagem. Nenhum código é executado no aplicativo de tela Inicial — tudo é executado remotamente em seu aplicativo, com o sistema de mensagens Android roteando a mensagem para o devido aplicativo.

Uma *exibição remota* combina um pouco de mágica com uma engenharia inovadora. Conhecida como classe `RemoteViews` na plataforma Android, ela permite que seu aplicativo forneça programaticamente uma interface de usuário remota para a tela Inicial em outro processo. O código do componente de aplicativo não é uma atividade real (como nos capítulos anteriores), mas uma implementação de `AppWidgetProvider`. Quando o Android roteia uma mensagem (como descrito no parágrafo anterior) para seu aplicativo a partir da tela Inicial, a mensagem é tratada em sua implementação da classe `AppWidgetProvider`.

Usando AppWidgetProviders

A classe `AppWidgetProvider` permite que o desenvolvedor interaja programaticamente com o componente de aplicativo na tela Inicial. Quando essa interação ocorre, as mensagens são enviadas do componente de aplicativo de tela Inicial para seu aplicativo via eventos de transmissão. Usando esses eventos de transmissão, você pode responder quando o componente de aplicativo é atualizado, ativado, desativado ou apagado. Você também pode atualizar a aparência do componente de aplicativo na tela Inicial fornecendo uma nova exibição. Como essa exibição está

localizada na tela Inicial e não dentro de seu aplicativo em execução, você usa `RemoteViews` para atualizar o layout da tela Inicial. Toda a lógica que determina o que deve acontecer é iniciada via implementação de `AppWidgetProvider`.

Imagine a estrutura do componente de aplicativo como o tradutor de uma conversa entre duas entidades. Se você precisar falar com alguém que fala italiano, mas você não conhece o idioma, deverá encontrar um tradutor que aceite sua entrada, traduza-a para o italiano e depois transmita sua mensagem para o falante italiano nativo. O mesmo processo aplica-se à estrutura do componente de aplicativo. Essa estrutura é seu tradutor.

Quando o nativo italiano (a tela Inicial, neste caso) precisa comunicar que algo aconteceu (tal como um usuário tocando em um botão), o tradutor (a estrutura do componente de aplicativo no sistema Android) traduz a ação em uma mensagem que você possa entender (tocar em um botão em particular). Nesse momento, você pode responder com a ação que deseja tomar (como por exemplo, mudar a cor de fundo do componente do aplicativo para um verde-limão) e o tradutor (a estrutura do componente de aplicativo) repete a mensagem para o falante italiano nativo (para tela Inicial via sistema Android). A tela Inicial atualizará a cor de fundo da exibição.

Os componentes de aplicativo podem aceitar apenas a entrada dos eventos do tipo toque. Quando você estiver trabalhando com um componente de aplicativo, não terá acesso às outras exibições básicas de entrada, tais como uma caixa de texto editável ou listas suspensas.

Trabalhando com Intenções Pendentes

Quando o usuário precisa interagir com seu aplicativo, ele se comunica tocando no componente de aplicativo usando a arquitetura de transmissão de mensagem Android (como descrito anteriormente) e você não é notificado imediatamente. Contudo, isto não significa que você *não pode* ser notificado sobre um evento de clique em seu componente de aplicativo — isso apenas é feito de um modo um pouco diferente das exibições normais.

Os eventos de clique do componente de aplicativo contêm instruções para o que fazer quando um evento de clique ocorre via classe `PendingIntent` na estrutura Android. Uma *intenção pendente* é uma implementação da classe `Intent` no Android, como explicado na seguinte seção.

Compreendendo o sistema de intenções do Android

Um objeto `Intent` no Android é uma mensagem informando ao Android para fazer algo acontecer. Quando você acende uma lâmpada usando um

interruptor na parede, a ação de sua intenção é acender a lâmpada, portanto, você muda a chave para a posição ligado. No Android, essa ação corresponde a criar uma instância da classe `Intent` com uma ação especificando que a luz será acesa:

```
Intent turnLightOn = new Intent("TURN_LIGHT_ON");
```

Essa intenção é iniciada no sistema de mensagens Android (como descrito no Capítulo 1) e a devida atividade lida com `Intent` (se diversas atividades responderem, o Android permitirá que o usuário escolha uma para fazer o trabalho). Contudo, no mundo físico, uma conexão elétrica é feita posicionando a chave na posição ligado, resultando na iluminação da lâmpada. No Android, você tem que fornecer um código, na forma de uma atividade, para fazer isto acontecer. Essa atividade (que pode ser chamada de `TurnLightOnActivity`) responde à intenção `turnLightOn`. Se você estiver trabalhando com um componente de aplicativo, deverá lidar com a intenção em um `BroadcastReceiver`, ao invés de uma atividade. `AppWidgetProvider` é uma instância de `BroadcastReceiver`, com alguns mecanismos que configuram grande parte da estrutura do componente do aplicativo para você. O objeto `BroadcastReceiver` é responsável por receber as mensagens de transmissão (broadcast).

`AppWidgetProvider` lida com a intenção da tela Inicial e responde com o devido resultado determinado, usando seu código, dentro de sua `AppWidgetProvider` personalizada. Porém, ela não trabalha com qualquer intenção. Se você quiser receber a entrada do componente de seu aplicativo, use `PendingIntent`.

Para entender o que é uma classe `PendingIntent`, você precisa compreender totalmente o conceito da classe `Intent` básica. Uma `PendingIntent` contém um objeto-filho `Intent`. De modo geral, uma intenção pendente age como uma intenção normal. Uma *intenção* é uma mensagem que pode ter uma grande variedade de dados descrevendo uma operação que precisa ser executada. Uma intenção pode ser endereçada a uma atividade específica ou transmitida para uma categoria genérica de receptores, conhecidos como `BroadcastReceivers` (que inclui `AppWidgetProvider`). O sistema `Intent`, `Activity` e `BroadcastReceiver` é remanescente da arquitetura de barramento de mensagens, onde uma mensagem é colocada em um barramento e qualquer endpoint responde à mensagem se (e somente se) souber como. Se nenhum endpoint souber como responder à mensagem ou se a mensagem não foi endereçada ao endpoint, ela será ignorada.

Uma intenção pode ser lançada no sistema de barramento de mensagens de algumas maneiras:

- **Iniciar outra atividade:** Use a chamada `startActivity()`, que aceita um objeto `Intent` como parâmetro.

- **Notificar qualquer componente** `BroadcastReceiver` **interessado:** Use a chamada `sendBroadcast()`, que também aceita uma intenção como parâmetro.
- **Comunicar-se com um serviço em segundo plano:** Use a chamada `startService()` ou `bindService()`, que aceita as intenções como parâmetros.

Uma atividade é a cola que liga os vários componentes de aplicativo porque fornece um mecanismo de vinculação tardia que permite a comunicação entre os aplicativos e dentro dos aplicativos.

Compreendendo os dados da intenção

Os dados de uma intenção consistem nestes elementos:

- **Ação:** A ação geral a ser executada. Algumas ações comuns incluem `ACTION_VIEW`, `ACTION_EDIT` e `ACTION_MAIN`. Você também pode fornecer sua própria ação personalizada.
- **Dados:** Os dados nos quais operar, tais como um registro em um banco de dados ou um identificador de recurso uniforme que deve ser aberto, tal como uma URL.

A Tabela 7-1 demonstra alguns parâmetros de ação e dados para os objetos `Intent` e sua estrutura de dados simples.

Tabela 7-1 Exemplos de dados de intenção

Definição	Dados	Resultado
`ACTION_VIEW`	`tel:123`	Exibe o discador com o número fornecido (123) preenchido.
`ACTION_DIAL`	`content://contacts/people/1`	Exibe o discador mostrando o número de telefone de contato com o ID1.
`ACTION_EDIT`	`content://contacts/people/1`	Edita as informações sobre a pessoa cujo identificador fornecido é 1.
`ACTION_VIEW`	`http://www.example.org`	Exibe a página Web da intenção fornecida.
`ACTION_VIEW`	`content://contacts/people`	Exibe uma lista de todas as pessoas no sistema Contatos.

As intenções também podem ter um array de outros dados que incluem estes elementos:

- **categoria:** Fornece informações adicionais sobre a ação a executar. Como exemplo, se `CATEGORY_LAUNCHER` estiver presente, o aplicativo deverá aparecer na inicialização de aplicativos como um aplicativo de alto nível. Outra opção, `CATEGORY_ALTERNATIVE`, pode fornecer ações alternativas que o usuário pode executar em uma parte dos dados.

- **tipo:** Especifica um determinado tipo (tipo MIME) dos dados de intenção. Por exemplo, quando você está definindo o tipo para `audio/mpeg`, o sistema Android reconhece que você está trabalhando com um arquivo MP3. Normalmente, o tipo é deduzido pelos dados em si. Definindo o tipo, você anula a dedução de tipo, definindo explicitamente o tipo na intenção.

- **componente:** Especifica explicitamente um nome de componente da classe na qual executar a intenção. Normalmente, o componente é deduzido pelo exame de outras informações na intenção (ação, dados/tipo e categorias) e os componentes coincidentes podem lidar com elas. Se este atributo for definido, nenhuma avaliação ocorrerá e esse componente será usado exatamente como especificado (provavelmente, o caso de uso mais comum em seus aplicativos). Você pode fornecer outra atividade como componente — isto endereça o Android para interagir com a classe específica.

- **extras:** Um grupo de informações adicionais baseadas em chaves, usado para fornecer informações extras para o componente receptor. Por exemplo, se você precisar enviar um endereço de e-mail, usará o grupo de extras para fornecer o corpo, o assunto e outros componentes do e-mail.

Avaliando as intenções

As intenções são avaliadas no sistema Android em uma de duas maneiras:

- **Explicitamente:** A intenção especificou um componente explícito ou a classe exata que executará os dados na intenção (mais uma vez, este é o modo mais comum de endereçar as intenções). Esse tipo de intenção geralmente não contém dados porque é um meio de iniciar outras atividades dentro de um aplicativo. Você descobrirá posteriormente neste capítulo como usar uma intenção explícita em um aplicativo.

- **Implicitamente:** A intenção não especificou um componente ou classe. Ao contrário, a intenção deve fornecer informações suficientes sobre a ação que precisa ser executada com os dados fornecidos para o sistema Android determinar quais componentes disponíveis podem lidar com a intenção — algumas vezes, referidos como *endereço* e *payload* (carga útil).

Um exemplo é configurar uma intenção de e-mail que contém os campos Para, CC, Assunto e Corpo, e um tipo MIME. O Android interpreta-o como um e-mail e dá ao usuário do dispositivo a oportunidade de escolher qual aplicativo deve lidar com a intenção. As possibilidades incluem Gmail, Exchange ou uma conta de e-mail POP. O usuário determina qual programa de e-mail usar. A capacidade do Android para identificar as possíveis correspondências para a intenção dada é conhecida como *resolução da intenção*.

Usando intenções pendentes

`PendingIntent` é uma intenção em si, mas com uma pequena mudança de paradigma em relação à funcionalidade: É criada por seu aplicativo e dada a outro aplicativo completamente diferente. Ao dar a outro aplicativo uma `PendingIntent`, você está concedendo a ele o direito de executar a operação especificada como se o aplicativo fosse seu. Em termos leigos, você está dando informações sobre como chamar seu aplicativo para realizar um trabalho em nome de outro aplicativo. Quando o outro aplicativo considera que certo trabalho precisa ocorrer, ele executa a `PendingIntent`, que instrui o sistema de mensagens Android para informar ao seu aplicativo que o trabalho deve ser executado.

Evitando o temido erro Aplicativo Não Respondendo (ANR)

Como todo o trabalho que acontece em `AppWidgetProvider` ocorre no principal thread da interface de usuário, você deve completar todo seu trabalho o mais rapidamente possível. Se sua `AppWidgetProvider` levar tempo demais para responder, seu código irá segurar o thread ida interface e fazer com que seu aplicativo exiba uma caixa de diálogo Application Not Responding (ANR — Aplicativo Não Respondendo) porque o sistema Android acredita que o aplicativo está congelado e não responde. Um exemplo é a comunicação em rede para fazer o download das atualizações do status a partir de um serviço, tal como o Twitter. Se o download dos status levar tempo demais (que pode ser muito menor do que você pode esperar), o Android mostrará a caixa de diálogo ANR, permitindo que o usuário saiba que o componente de aplicativo não está respondendo; nesse ponto, o usuário poderá fechar à força o aplicativo.

Um modo de evitar o erro ANR é implementar um serviço dentro de `AppWidgetProvider`, que executa seu trabalho em um thread em segundo plano. O `IntentService` que você implementará nas seguintes seções ajudará a evitar os erros ANR e permitirá que o componente continue sendo muito rápido.

Para o aplicativo Silent Mode Toggle, você usará a chamada `PendingIntent.getBroadcast()` para obter uma instância de intenção pendente. Essa chamada retorna uma `PendingIntent`, que é usada para as transmissões no sistema. A chamada aceita estes quatro parâmetros:

- `Context`: O contexto no qual `PendingIntent` deve realizar a transmissão.
- `RequestCode`: O código de solicitação privada para o emissor. Não usado atualmente; portanto, é transmitido o valor zero.
- `Intent`: A intenção a ser transmitida.
- `Flags`: Um conjunto de controles usado para controlar a intenção quando ela é iniciada. Não é usado atualmente no aplicativo Silent Mode Toggle; portanto, é transmitido o valor zero.

Espere um pouco — este código usa `Intent`, assim como `PendingIntent`. Por quê? O objeto `Intent` é colocado dentro de `PendingIntent` porque `PendingIntent` é usada para a comunicação entre processos. Quando `PendingIntent` é inicializada, o trabalho real que precisa ser feito é colocado para o objeto-filho `Intent`.

É muita informação! Agora que você entende o básico do sistema de intenções do Android, é hora de implementar as partes internas do aplicativo dentro deste componente de aplicativo.

Criando o Componente da Tela Inicial

O processo de enviar mensagens entre o componente de aplicativo da tela Inicial e seu aplicativo é tratado pelo sistema de mensagens do Android, a classe `PendingIntent` e a `AppWidgetProvider`. Nesta seção, você construirá cada componente para fazer funcionar seu primeiro componente de aplicativo na tela Inicial.

Implementando a AppWidgetProvider

Implementar a `AppWidgetProvider` é bem simples. Abra o Eclipse e o aplicativo Silent Mode Toggle.

Para adicionar uma nova classe ao pacote `com.dummies.android.silentmodetoggle` e fornecer um nome, tal como `AppWidget.java`, siga estas etapas:

1. **Clique com o botão direito em** `com.dummies.android.silentmodetoggle` **na pasta src/ e escolha New (Nova)⇨Class (Classe).**

   A caixa de diálogo New Java Class (Nova Classe Java) será aberta, como mostrado na Figura 7-3.

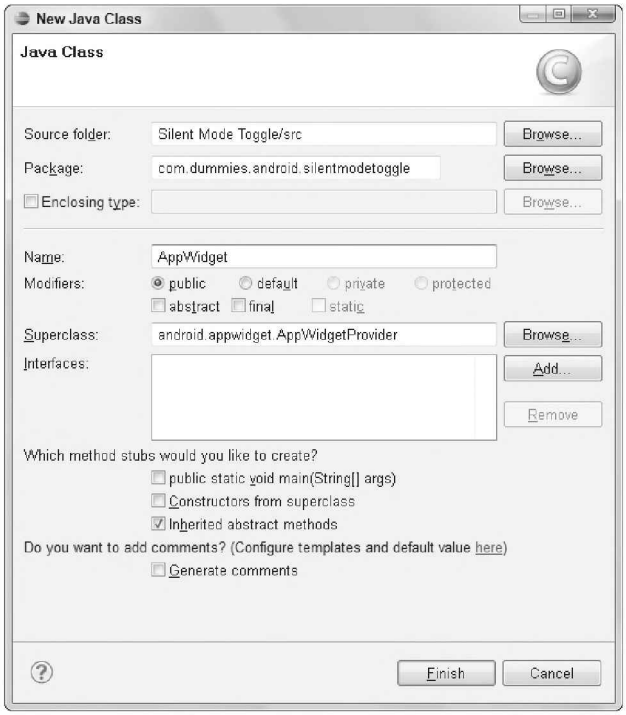


Figura 7-3: A caixa de diálogo New Java Class.

2. **Forneça o nome da classe e defina sua superclasse para** `android. appwidget.AppWidgetProvider`**.**

3. **Clique em Finish (Terminar) quando terminar.**

   A nova classe será adicionada ao pacote selecionado.

`AppWidgetProvider` faz todo o trabalho de responder aos eventos de `RemoteViews`, mas como? Se você vir a documentação Android `AppWidgetProvider`, poderá observar que é uma subclasse direta de `BroadcastReceiver`. Em uma visão de alto nível, `BroadcastReceiver` é um componente que pode receber mensagens de transmissão do sistema Android. Quando um usuário toca em uma exibição clicável em `RemoteViews` na tela Inicial (tal como um botão), o sistema Android transmite uma mensagem informando ao receptor que a exibição foi clicada. Depois de a mensagem ser transmitida, `AppWidgetProvider` pode tratá-la.

Note que como essas mensagens são de *transmitidas em broadcast*, elas são enviadas para todo o sistema. Se o payload da mensagem e as informações do endereço de destino forem vagos, vários objetos `BroadcastReceiver` poderão tratar a mensagem. A `AppWidgetProvider` construída nesta seção é endereçada a um único destino, isso é parecido com entrar em uma sala repleta de empreiteiros e perguntar se algum deles poderia fazer um trabalho para você — todos responderiam. Você tem um endereço de mensagem e payload vagos. Porém, se você solicitar ao mesmo grupo um

pequeno empreiteiro eletricista eletrônico com o nome João Silva, apenas um responderá. Você tem uma mensagem especificamente endereçada, com um endereço detalhado e informações de payload.

Comunicando-se com o componente do aplicativo

A classe `AppWidgetProvider` não tem código no início — é uma concha vazia. Para tornar sua `AppWidgetProvider` capaz de fazer algo, você adiciona um código para responder à intenção (a mensagem) que foi enviada para sua `AppWidgetProvider`. No arquivo de código recém-criado, digite no editor o código mostrado na Listagem 7-1 (nota: O nome da classe é `AppWidget`, portanto, se a sua for diferente, mude essa linha).

Listagem 7-1: A configuração inicial do componente do aplicativo

```
public class AppWidget extends AppWidgetProvider {                    →1
     @Override
     public void onReceive(Context ctxt, Intent intent) {             → 4
                 if (intent.getAction()==null) {                        5
                  // Faça algo
                 } else {
                   super.onReceive(ctxt, intent);                     →10
                 }
             }

             @Override
             public void onUpdate(Context context, AppWidgetManager
              appWidgetManager, int[] appWidgetIds) {                 → 15
              // Faça algo
            }
}
```

Esta lista descreve rapidamente as linhas numeradas:

→ 1 Esta linha de código informa ao Android que a classe é `AppWidgetProvider` porque a classe está herdando de `AppWidgetProvider`.

→ 4 Esta linha sobrescreve o método `onReceive()` para ser capaz de detectar quando uma nova intenção é recebida de `RemoteViews`. Essa intenção pode ter sido iniciada por um usuário tocando em uma exibição para executar uma ação, tal como, um clique do botão. O objeto `Intent` está contido em `PendingIntent`, que iniciou a solicitação.

→ 5 Os objetos `Intent` podem conter várias partes de dados. Uma dessas partes é a ação. Essa linha de código verifica se a

intenção tem uma ação. Se não tiver, a intenção será enviada. Este processo pode parecer um retrocesso, mas você poderá descobrir mais nas futuras seções.

→ **10** Esta linha delega o trabalho para a superclasse porque você não precisa fazer nada com a intenção (essa intenção não é o que você estava esperando; a intenção tinha uma ação e você está esperando uma intenção sem uma ação). Isso aconteceria se o componente de aplicativo atualizasse a si mesmo automaticamente com regularidade, quando você o define nos metadados do componente (os metadados serão explicados na seção "Trabalhando com os metadados do componente de aplicativo", posteriormente neste capítulo). Fazer isso chamaria um dos muitos métodos predefinidos para ativar, desativar, iniciar, parar ou atualizar o componente de aplicativo, como na linha 15.

→ **15** O método `onUpdate()` é chamado pela estrutura Android de tempos em tempos, que você pode definir nos metadados do componente. Esse método é chamado porque a estrutura Android primeiro percebe que o tempo passou, e então, deseja que você tenha a oportunidade de atualizar a exibição de modo proativo, sem a interação do usuário, tal como, um componente de aplicativo de notícias atualizando-se a cada 30 minutos com as últimas manchetes. A operação não requer interação com o usuário porque ocorreria de tempos em tempos. Esse método assegura que seu componente seja configurado corretamente.

Construindo o layout do componente do aplicativo

O componente de aplicativo precisa ter um certo layout para o Android determinar como exibir o componente na tela Inicial. O arquivo de layout do componente define como o componente será exibido na tela Inicial. Anteriormente neste capítulo, a Figura 7-2 mostrou o componente de aplicativo sendo executado no emulador. O ícone na tela Inicial é definido pelo arquivo de layout do componente. Se você alterar a cor de segundo plano do arquivo de layout para verde-limão, a cor de fundo do componente na tela Inicial mudaria também para verde-limão, ao invés de transparente.

A caixa verde-limão também identifica o espaço da tela disponível para o componente do aplicativo. Seu componente de aplicativo pode ocupar uma célula da tela Inicial ou muitas células. Esse componente de aplicativo está ocupando apenas uma.

Para criar o layout do componente, crie um arquivo de layout XML no diretório `res/layouts`. Nomeie-o como `widget.xml`.

O conteúdo de `widget.xml` é mostrado na Listagem 7-2.

Listagem 7-2: O conteúdo de widget.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ImageView android:id="@+id/phoneState"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/ic_launcher"
        android:clickable="true" />                                    → 9
</RelativeLayout>
```

Este layout não deve ser novo. É um `RelativeLayout` que tem uma exibição-filha: um `ImageView` clicável. Você pode clicá-lo definindo a propriedade clicável para `true` na linha 9 da Listagem 7-2.

Note a propriedade `src` de `ImageView` definida para o ícone do aplicativo. Pode parecer estranho, mas quando você constrói o layout, não criou ainda os botões de estado do telefone que representavam os estados silencioso e normal. Você precisa visualizar a exibição no designer de layout enquanto projeta o layout. Portanto, este código usa `@drawable/ic_launcher` como o valor de `ImageView` para ter uma visão de como ficará a exibição. Não se preocupe em usar o ícone do aplicativo — quando o componente de aplicativo carregar, `ToggleService` mudará o valor do ícone para o ícone do modo Silent (Silencioso) ou Normal, como mostrado posteriormente neste capítulo.

Estes ícones ajudam o usuário a identificar o estado atual do aplicativo. O ícone `phone_state_normal` indica quando o telefone está no modo de campainha Normal. O ícone `phone_state_silent` indica quando o telefone está no modo de campainha Silent.

Fazendo o trabalho dentro de uma AppWidgetProvider

Depois da intenção pendente ter iniciado sua `AppWidgetProvider`, você executará um trabalho em nome do aplicativo chamador (neste caso, o aplicativo da tela Inicial). Nas seguintes seções, você fará um trabalho relativo ao tempo em nome de quem faz a chamada.

Antes de entrar no código, você deve entender como a `AppWidgetProvider` funciona. Devido à natureza e ao uso intenso de recursos dos processos remotos, todo o trabalho não trivial deve ser feito dentro de um serviço em segundo plano, tal como, mudar o modo da campainha via serviço em segundo plano.

Entendendo IntentService

Esta seção explica por que você usa um serviço em segundo plano para uma tarefa comum, tal como, mudar o modo da campainha do telefone.

Qualquer código executado por tempo demais sem responder ao sistema Android está sujeito ao erro Aplicativo Não Respondendo (ANR). Os componentes de aplicativo são especialmente vulneráveis aos erros ANR porque estão executando código em um processo remoto e porque os componentes de aplicativo executam tarefas que atravessam as fronteiras de vários processos, que levam tempo para configurar, executar e abrir — o processo inteiro requer muita CPU, memória e bateria. O sistema Android observa os componentes de aplicativo para assegurar que eles não levarão tempo demais para serem executados. Quando levam, o aplicativo chamador (a tela Inicial) é bloqueada e o dispositivo fica inutilizado. Portanto, a plataforma Android deseja assegurar que você nunca seja capaz de tornar o dispositivo não responsivo por mais de alguns segundos.

Como os componentes de aplicativo utilizam muita CPU e memória, julgar se um componente de aplicativo causará um erro ANR é difícil. Se o dispositivo não estiver fazendo nenhuma outra tarefa pesada, o componente do aplicativo provavelmente funcionará bem. Porém, se o dispositivo estiver no meio de operações com uso intenso de CPU ou de E/S, o componente de aplicativo poderá levar tempo demais para responder – causando um erro ANR. Para resolver o problema, mova qualquer trabalho intensivo de CPU ou de E/S do componente de aplicativo para um `IntentService`, que pode levar quanto tempo for necessário para completar — o que, por sua vez, não afeta o aplicativo da tela Inicial.

Diferentemente da maioria dos serviços em segundo plano, que têm uma longa execução, um `IntentService` usa o padrão de processador fila de trabalho, que lida com uma intenção por vez usando um thread de trabalho e para, quando fica sem trabalho. Em termos leigos, `IntentService` simplesmente executa o trabalho dado a ele como um serviço em segundo plano, e então, para o serviço quando nenhum outro trabalho precisa ser feito.

Implementando AppWidgetProvider e IntentService

Na classe `AppWidgetProvider`, digite o código na Listagem 7-3 em seu editor de código.

Listagem 7-3: A implementação completa de AppWidget

```
public class AppWidget extends AppWidgetProvider {
              @Override
              public void onReceive(Context context, Intent intent) {
                 if (intent.getAction()==null) {
                     context.startService(new Intent(context,
                                          ToggleService.class));          → 6
                 } else {
                     super.onReceive(context, intent);
                  }
```

```
    }

    @Override
    public void onUpdate(Context context, AppWidgetManager
         appWidgetManager, int[] appWidgetIds) {
    context.startService(new Intent(context,
                                ToggleService.class));          → 16
  }

  public static class ToggleService extends IntentService {     → 19

  public ToggleService() {
     super(ToggleService.class.getName());                      → 22
  }
  @Override
  protected void onHandleIntent(Intent intent) {                → 26
      ComponentName me=new ComponentName(this, AppWidget.class); → 27
      AppWidgetManager mgr=AppWidgetManager.getInstance(this);  → 28
      mgr.updateAppWidget(me, buildUpdate(this));               → 29
  }
  private RemoteViews buildUpdate(Context context) {            → 30
         RemoteViews updateViews=new
         RemoteViews(context.getPackageName(),R.layout.widget);  → 32
         AudioManager audioManager =
(AudioManager)context.getSystemService(Activity.AUDIO_SERVICE);  → 34

         if(audioManager.getRingerMode() ==
               AudioManager.RINGER_MODE_SILENT) {

                updateViews.setImageViewResource(R.id.phoneState,
                    R.drawable.phone_state_normal);             → 40

         audioManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
         } else {
           updateViews.setImageViewResource(R.id.phoneState,
               R.drawable.phone_state_silent);                  → 45

          audioManager.setRingerMode(AudioManager.RINGER_MODE_SILENT);
         }
       Intent i=new Intent(this, AppWidget.class);              → 49

       PendingIntent pi
              = PendingIntent.getBroadcast(context, 0, i,0);

       updateViews.setOnClickPendingIntent(R.id.phoneState,pi);  → 54

       return updateViews;                                      → 56
    }
}
}
```

A seguinte lista explica rapidamente a finalidade das principais seções do código:

→ **6** Esta linha de código inicia uma nova instância de `ToggleService`. O objeto `context` refere-se ao objeto `Context` do Android, que é uma interface para as informações globais sobre o aplicativo. O contexto é transmitido para as chamadas do método `onReceive()` e `onUpdate()`. Uma nova intenção é criada para permitir que o sistema Android saiba o que deve acontecer. Esse método é iniciado pelo usuário quando ele toca no componente de aplicativo na tela Inicial.

→ **16** Esta linha executa as mesmas ações da linha 6.

→ **19** Esta implementação de `IntentService` lida com a mesma lógica de `MainActivity` para lidar com a troca de modo do telefone, mas em relação à infraestrutura do componente de aplicativo. É uma implementação de um serviço em segundo plano na plataforma Android. Essa classe é uma classe estática aninhada dentro do componente de aplicativo.

→ **22** Este método chama a superclasse com o nome "`com.dummies.android.silentmodetoggle.AppWidget$ToggleService`". Essa chamada do método está ocorrendo para ajudar a depurar a thread. Se você omitir esta linha de código, um erro do compilador informará que você deve chamar explicitamente o construtor da superclasse.

→ **26** O método `onHandleIntent()` é responsável por lidar com a intenção que foi transmitida para o serviço. Neste caso, seria a intenção criada nas linhas 6 e 16. Como a intenção criada é uma intenção explícita (você especificou um nome de classe para executar), nenhum dado extra foi fornecido e quando você chegar à linha 26, não precisará mais usar a intenção (você pode fornecer informações extras para o objeto `Intent` ser extraído deste parâmetro de método. Neste caso, o objeto `Intent` seria meramente um mensageiro para instruir `ToggleService` para iniciar seu processamento).

→ **27** Cria um objeto `ComponentName`. Esse objeto é usado com `AppWidgetManager` (explicado em seguida) como o provedor do novo conteúdo que será enviado para o componente de aplicativo via instância `RemoteViews`.

→**28** Obtém uma instância de `AppWidgetManager` na chamada estática `AppWidgetManager.getInstance()`. A classe `AppWidgetManager` é responsável por atualizar o estado do componente de aplicativo e fornece outras informações sobre o componente de aplicativo instalado. Você usa-a para atualizar o estado do componente de aplicativo.

→ **29** Atualiza o componente de aplicativo com uma chamada para `updateAppWidget()`. Esta chamada precisa de dois componentes: o `ComponentName` do Android que está fazendo a atualização e o objeto `RemoteViews` usado para atualizar o componente de aplicativo. O `ComponentName` é criado na linha 27. O objeto `RemoteViews` usado para atualizar o estado do componente de aplicativo na tela Inicial é um pouco mais complicado (como explicado em seguida).

→ **30** Esta definição de método para `buildUpdate()` retorna um novo objeto `RemoteViews`, que é usado na linha 29. A lógica para o que deve acontecer e as ações necessárias para continuar o são incluídas nesse método.

→ **32** Esta linha constrói um objeto `RemoteViews` com o nome do pacote atual, assim como o layout que será retornado deste método. O layout, `R.layout.widget`, é mostrado na Listagem 7-3.

→ **34** Esta linha obtém uma instância de `AudioManager` e verifica o estado da campainha. Se a campainha estiver silenciosa, o usuário tocou no componente de aplicativo para mudar seu estado, indicando que ele deseja que a campainha do telefone esteja no modo normal agora.

→ **40** A linha atualiza o objeto `RemoteViews`. O objeto `RemoteViews` está mudando o desenho `ImageView R.id.phoneState` para o desenho `R.drawable.phone_state_normal` (consulte o ícone à direita na Figura 7-2).

→ **45** A instrução `else` localizada acima desta linha flui para atualizar a imagem no `ImageView` para `R.drawable.phone_state_silent` porque o modo da campainha não estava no modo Silent (Silencioso) anteriormente (o usuário deseja agora silenciar o telefone).

→ **49** Cria um objeto `Intent` que inicia a classe `AppWidget` quando inicializado.

Os componentes de aplicativo não podem comunicar-se com as intenções comuns; eles requerem o uso de um `PendingIntent`. Lembre-se que como os componentes de aplicativo utilizam comunicação entre processos, os objetos `PendingIntent` são necessários para a comunicação. Nesta linha, `PendingIntent` instrui o componente de aplicativo sobre sua próxima ação através de sua intenção filha.

→ **52** Constrói o `PendingIntent` que instrui o componente do aplicativo sobre o que fazer quando alguém clica ou toca na exibição.

→ **54** Define o atendente do clique para `ImageView` no componente de aplicativo. Como você está trabalhando com `RemoteViews`, tem que reconstruir a hierarquia de eventos inteira na exibição, pois a estrutura do componente de aplicativo substitui o `RemoteViews` inteiro por um novinho, que você fornece via este método. Portanto, você tem uma tarefa restante: informar ao `RemoteViews` o que fazer quando ele é tocado ou clicado na tela Inicial. `SetOnClickPendingIntent()` configura `PendingIntent`. Esse método aceita dois parâmetros: o ID da exibição que foi clicada (neste caso, uma imagem) e o argumento `pi`, que é `PendingIntent`.

→ **56** Retorna o objeto `RemoteViews` recém-criado para que a chamada `updateAppWidget()` na linha 29 possa atualizar o componente de aplicativo.

Trabalhando com os metadados do componente de aplicativo

Depois de você ter escrito o código para lidar com a atualização do componente de aplicativo, pode se perguntar como listar o widget de aplicativo no menu Widgets (Componentes). Este processo bem simples requer adicionar um único arquivo XML ao seu projeto. Esse arquivo XML descreve os metadados básicos sobre o widget de aplicativo, de modo que a plataforma Android possa determinar como fazer o layout do componente de aplicativo na tela Inicial. Siga estas etapas:

1. **Em seu projeto, clique com o botão direito no diretório `res` e escolha New (Nova)⇨New Folder (Nova Pasta).**
2. **Nomeie a pasta como `xml` e clique em Finish (Terminar).**
3. **Clique com o botão direito na nova pasta `res/xml` e escolha New (Novo)⇨Android XML File (Arquivo XML Android).**
4. **No New Android XML File Wizard (Assistente para Novo Arquivo XML Android), digite `widget_provider.xml` para o nome de arquivo.**
5. **Selecione `AppWidgetProvider` na lista suspensa, então, clique em Finish.**
6. **Depois do arquivo abrir, abra o editor XML e digite o seguinte código no arquivo `widget_provider.xml`:**

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/
          android"
          android:minWidth="79px"
          android:minHeight="79px"
          android:updatePeriodMillis="1800000"
          android:initialLayout="@layout/widget"
/>
```

As propriedades `minWidth` e `minHeight` são usadas para definir a quantidade mínima de espaço que a exibição precisa na tela Inicial. Esses valores podem ser maiores, se você quiser, mas para um aplicativo de telefone, tudo o que realmente precisa é de 79x79 para colocar o ícone indicador que o componente exibirá.

A propriedade `updatePeriodMillis` define com que frequência o componente de aplicativo deverá tentar atualizar-se. No caso do aplicativo Silent Mode Toggle, você raramente, talvez nunca, precisará que isto aconteça. Portanto, esse valor é definido para 1800000 milissegundos — 30 minutos. A cada 30 minutos, o aplicativo tenta atualizar-se enviando uma intenção que executa a chamada do método `onUpdate()` em `AppWidgetProvider`.

A propriedade `initialLayout` define como fica o aplicativo de componente quando ele é adicionado pela primeira vez à tela Inicial, antes de qualquer trabalho ocorrer. O componente de aplicativo pode levar alguns segundos para inicializar e atualizar o seu objeto `RemoteViews` chamando o método `onReceive()`.

Um exemplo de atraso maior é um componente de aplicativo que verifica o Twitter para obter as atualizações do status. `initialLayout` é mostrada até que as atualizações sejam recebidas do Twitter. Informe ao usuário em `initialLayout` que as informações estão sendo carregadas para mantê-lo ciente do que está acontecendo enquanto o componente de aplicativo é carregado na tela inicial. Você pode fazer isso fornecendo `TextView` com o

conteúdo "Loading..." (Carregando...), enquanto AppWidgetProvider faz seu trabalho.

Registrando seus novos componentes no manifest

Sempre que você adiciona uma atividade, um serviço ou receptor de transmissão (broadcast receiver — além de outros itens) ao seu aplicativo, precisa registrá-los no arquivo manifest do aplicativo. O manifest do aplicativo apresenta informações vitais para a plataforma Android — a saber, os componentes do aplicativo. O sistema não reconhece os objetos Activity, Service e BroadcastReceiver, que não estejam registrados no manifest do aplicativo. Portanto, se você adicionou o componente de aplicativo à tela Inicial, ele paralisaria porque seu AppWidgetProvider é um BroadcastReceiver e o código no receptor está usando um serviço que não está registrado no manifest.

Para adicionar AppWidgetProvider e IntentService ao arquivo manifest de seu aplicativo, abra o arquivo AndroidManifest.xml e digite o código mostrado na Listagem 7-4 no arquivo já existente. As linhas em negrito são as linhas recém-adicionadas para os novos componentes.

Listagem 7-4: Um arquivo AndroidManifest.xml atualizado com novos componentes registrados

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.dummies.android.silentmodetoggle"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:debuggable="true">
        <activity android:name=".MainActivity"
                android:label="@string/app_name">
                <intent-filter>
                   <action android:name="android.intent.action.MAIN" />
                   <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
        </activity>
        <receiver android:name=".AppWidget"
                android:label="@string/app_name"
                android:icon="@drawable/ic_launcher">                    →18
                <intent-filter>
                    <action
               android:name="android.appwidget.action.APPWIDGET_UPDATE" />→21
```

```
                </intent-filter>
                <meta-data
                                        android:name="android.appwidget.
            provider"
                        android:resource="@xml/widget_provider" />          →25
        </receiver>
        <service android:name=".AppWidget$ToggleService" />                 →27
    </application>
    <uses-sdk android:minSdkVersion="4" />
</manifest>
```

A seguinte lista descreve rapidamente cada seção:

→ **18** O elemento de abertura registra um `BroadcastReceiver` como parte deste aplicativo. A propriedade `name` identifica o nome do receptor — neste caso, `.AppWidget`, que se relaciona ao arquivo `AppWidget.java` no aplicativo. O nome e a etiqueta ajudam a identificar o receptor.

→ **21** Identifica a qual tipo de intenção (baseada na ação da intenção no filtro de intenção) o componente de aplicativo responde automaticamente quando determinada intenção é transmitida. Conhecido como `IntentFilter`, ele ajuda o sistema Android a conhecer para quais tipos de eventos seu aplicativo deve ser notificado. Neste caso, seu aplicativo está preocupado com a ação `APPWIDGET_UPDATE` da intenção transmitida. Esse evento é disparado após um intervalo de tempo definido pela propriedade `updatePeriodMillis`, o que é definido no arquivo `widget_provider.xml`. As outras ações incluem ativado, apagado e desativado.

→ **25** Identifica o local dos metadados que você incluiu recentemente em seu aplicativo. O Android usa os metadados para ajudar a determinar os padrões e fazer o layout dos parâmetros de seu componente de aplicativo.

→ **27** O elemento `<service>` registra `AppWidget$ToggleService IntentService` em seu aplicativo. É o serviço em segundo plano que faz grande parte do trabalho de seu componente.

Neste ponto, seu aplicativo está pronto para ser instalado e testado. Para instalar o aplicativo, escolha Run (Executar)⇨Run ou pressione Ctrl+F11. Ele deverá aparecer no emulador. Volte para a tela Inicial pressionando a tecla Home. Agora, você poderá adicionar à tela Inicial o componente de aplicativo que criou recentemente.

Colocando Seu Componente na Tela Inicial

Os especialistas em usabilidade na equipe de desenvolvimento do Android fizeram um ótimo trabalho ao permitir que os componentes de aplicativo sejam adicionados facilmente à tela Inicial. Adicionar um é fácil — siga estas etapas:

1. **Abra a lista de aplicativos na tela Inicial do emulador.**
2. **Quando a lista de aplicativos ficar visível, selecione Widgets (Componentes).**
3. **Escolha Silent Mode Toggle, como mostrado na Figura 7-4.**

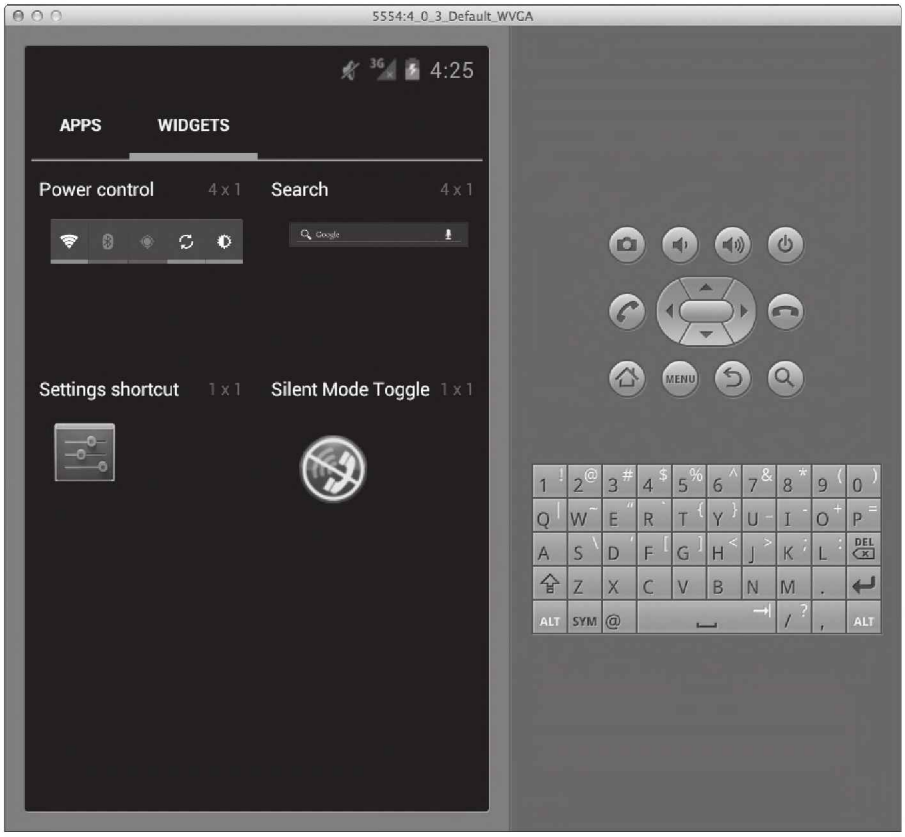


Figura 7-4: O componente Silent Mode Toggle na lista Widgets.

Agora, você adicionou o componente Silent Mode Toggle à tela inicial, como mostrado na Figura 7-5. Você pode tocar no ícone para mudar o modo da campainha e o segundo plano mudará de acordo (consulte a Figura 7-2).

Figura 7-5: O componente de aplicativo, adicionado à tela Inicial.

Capítulo 8

Publicando Seu Aplicativo na Google Play Store

Neste Capítulo

- Construindo um arquivo de pacote Android
- Criando um perfil de desenvolvedor na Google Play Store
- Escolhendo um preço para seu aplicativo
- Ilustrando seu aplicativo com uma captura de tela
- Fazendo upload e publicando seu aplicativo
- Monitorando os downloads

A Google Play Store é o mecanismo oficial de distribuição de aplicativos do Android. Publicar seu aplicativo na loja permite que ele seja baixado, instalado e usado por milhões de usuários no mundo. Os usuários também podem avaliar seu aplicativo e deixar comentários sobre ele, o que ajuda a identificar as possíveis tendências de uso e as áreas problemáticas que os usuários podem estar encontrando.

A Google Play Store também fornece um conjunto valioso de estatísticas que você pode usar para medir o sucesso de seu aplicativo.

Neste capítulo, você publicará seu aplicativo na Google Play Store. Descobrirá como fornecer algumas capturas de tela, uma captura de tela promocional e uma pequena descrição de seu aplicativo.

Criando um Arquivo de Distribuição

Então, você teve uma ótima ideia e ela o levou a desenvolver o próximo aplicativo ou jogo de sucesso para a plataforma Android. Agora, você está

pronto para colocar o aplicativo nas mãos dos usuários. A primeira coisa que você precisa fazer é empacotar seu aplicativo para que ele possa ser colocado em seus dispositivos. Para tanto, você cria um arquivo de pacote Android ou arquivo *APK*.

Nas seguintes seções, você criará um arquivo APK.

Revendo o arquivo manifest

Antes de começar a criar o arquivo APK de distribuição, você deve ter muito cuidado e assegurar que seu aplicativo esteja disponível para o maior número possível de usuários, familiarizando-se com o elemento `uses-sdk` no arquivo `AndroidManifest.xml`. Seu arquivo `AndroidManifest.xml` agora tem uma entrada `uses-sdk` (veja Capítulo 4):

```
<uses-sdk android:minSdkVersion="4" />
```

A propriedade `minSdkVersion` identifica quais versões da plataforma Android podem instalar este aplicativo — neste caso, nível 4. O aplicativo Silent Mode Toggle foi desenvolvido definindo o kit de desenvolvimento de software de destino (SDK) para a versão 15.

A plataforma Android é, em grande parte, compatível com as versões anteriores. A maioria dos recursos da versão 3 também está na versão 4. Pequenas alterações e, algumas vezes, novos componentes grandes são lançados em cada nova versão, mas tudo mais na plataforma permanece basicamente compatível com as versões anteriores. Portanto, afirmar que esse aplicativo precisa de uma versão SDK 4 mínima significa que qualquer sistema operacional Android da versão 4 ou posterior poderá executar o aplicativo.

Usando as informações `minSdkVersion`, a Google Play Store pode determinar quais aplicativos mostrar ao usuário de um dispositivo específico. Se você lançar o aplicativo agora com o valor `minSdkVersion` definido para 4 e abrir a Google Play Store em um dispositivo Android executando a versão 3 (Android 1.5) ou anterior, não encontrará o aplicativo — a Google Play Store filtra-o porque você, o desenvolvedor, especificou que esse aplicativo pode ser executado apenas nos dispositivos da API Nível 4 ou superior. Se você abrir a Google Play Store em um dispositivo executando a API Nível 4 ou superior, encontrará e instalará seu aplicativo.

Se você não fornecer um valor `minSdkVersion` no elemento `uses-sdk` do manifest do aplicativo, a Google Play Store usará como padrão `minSdkVersion` 1, significando que esse aplicativo é compatível com todas as versões do Android. Se seu aplicativo usar um componente que está indisponível nas antigas versões da plataforma (tal como o Account Manager — Gerenciador de Contas — introduzido no Android 2.0) e um usuário instalar seu aplicativo, o aplicativo irá paralisar.

Sempre defina `minSdkVersion` para a versão mais baixa do SDK na qual você testou seu aplicativo e defina `targetSdkVersion` para a versão testada mais alta.

Escolhendo suas ferramentas

Você pode construir um arquivo APK do Android de várias maneiras:

- Ferramentas de Desenvolvimento do Android (Android Development Tools — ADT) dentro do Eclipse
- Processo de construção automático, tal como, um serviço de integração contínua
- Linha de comando com Ant
- Sistemas de construção Maven ou Gradle

Você usa o ADT no Eclipse para criar um arquivo APK. O ADT fornece um conjunto de ferramentas que compila, assina digitalmente e empacota seu aplicativo Android em um arquivo APK. Aqui, ocorre o processo de assinatura digital, como analisado na próxima seção.

As outras opções, tais como o Ant e a integração contínua, são possíveis, mas são usadas em situações mais avançadas. Você pode encontrar mais informações sobre a configuração de um processo de construção Ant na documentação Android em `http://d.android.com/tools/publishing/app-signing.html` (conteúdo em inglês).

Assinando digitalmente seu aplicativo

O sistema Android requer que todos os aplicativos instalados sejam assinados digitalmente com um certificado que contém um par de chaves pública/privada. A chave privada é mantida pelo desenvolvedor. O certificado usado para assinar digitalmente o aplicativo identifica o desenvolvedor e estabelece as relações de confiança entre os aplicativos.

Você precisa conhecer algumas informações-chave sobre a assinatura dos aplicativos Android:

- Todos os aplicativos Android *devem ser assinados*. O sistema não instalará os aplicativos que não estão assinados.
- Você pode usar certificados com assinatura própria para assinar seus aplicativos; uma autoridade certificadora não é necessária.
- Quando você estiver pronto para lançar seu aplicativo na loja, deverá assiná-lo com uma chave privada. Você não pode publicar o aplicativo com a chave de depuração que assina o arquivo APK ao depurar o aplicativo durante o desenvolvimento.

- O certificado tem uma data de expiração e ela é verificada apenas na instalação. Se o certificado expirar depois que o aplicativo for instalado, ele continuará a operar normalmente.
- Se você não quiser usar as ferramentas ADT para gerar o certificado, poderá usar outras ferramentas, tais como Keytool ou Jarsigner, para gerar e assinar seus arquivos APK.

Você pode criar aplicativos modulares que podem comunicar-se entre si, caso os aplicativos tenham sido assinados com o mesmo certificado. Esta organização permite que os aplicativos sejam executados dentro do mesmo processo e o sistema poderá, se solicitado, tratá-los como um único aplicativo. Usando esta metodologia, você poderá criar seu aplicativo em módulos e os usuários poderão atualizá-los como acharem melhor — por exemplo, criar um jogo, e então lançar pacotes para atualizá-lo. Os usuários poderão decidir comprar apenas as atualizações desejadas.

O processo de certificação é descrito em detalhes na documentação Android em `http://d.android.com/tools/publishing/app-signing.html` (conteúdo em inglês).

Criando um armazenamento de chaves

Um *armazenamento de chaves* keystore no Android (e no Java) é um contêiner no qual residem seus certificados pessoais. Você pode usar algumas ferramentas no Android para criar um arquivo de armazenamento de chaves:

- **ADT Export Wizard (Assistente para Exportar ADT):** Instalado com o ADT, permite que você exporte um arquivo APK autoassinado que pode assinar digitalmente o aplicativo, assim como criar o certificado e o armazenamento de chaves (se necessário) em um processo do tipo assistente. Você criará um armazenamento de chaves quando criar seu arquivo APK na próxima seção.
- **Aplicativo Keytool:** Permite que você crie um armazenamento de chaves autoassinado via linha de comando. O Keytool, localizado no diretório bin do Java, fornece muitas opções através da linha de comando.

Protegendo seu armazenamento de chaves

O arquivo de armazenamento de chaves contém seu certificado privado, que o Android usa para identificar seu aplicativo na Google Play Store. Faça backup de seu armazenamento de chaves em um local seguro porque se você o perder, não poderá assinar o aplicativo com a mesma chave privada, nem poderá atualizar seu aplicativo porque a plataforma Google Play Store reconhece que o aplicativo não está assinado pela mesma chave e impede a atualização — a loja verá o arquivo como um novo aplicativo Android. Isto também acontecerá se você mudar o nome do pacote do aplicativo: O Android não irá reconhecê-lo como uma atualização válida porque o pacote e/ou certificado não é igual.

Criando o arquivo APK

Para criar seu primeiro arquivo APK, siga estas etapas:

1. **Abra o Eclipse, se ele ainda não estiver aberto.**

2. **Clique com o botão direito no aplicativo Silent Mode Toggle, escolha Android Tools (Ferramentas Android), e então escolha Export Signed Application Package (Exportar Pacote do Aplicativo Assinado).**

   O Export Android Application Wizard (Assistente para Exportar Aplicativo Android), mostrado na Figura 8-1, será aberto com o nome do projeto atual preenchido.

Figura 8-1: Assistente para Exportar Aplicativo Android.

3. **Clique no botão Next (Próximo).**

   A tela Keystore Selection (Seleção do Armazenamento de Chaves) será aberta, como mostrado na Figura 8-2.

4. **Selecione o botão de rádio Create New Keystore (Criar Novo Armazenamento de Chaves).**

   Se você já tiver um armazenamento de chaves, escolha a opção Use Existing Keystore (Usar Armazenamento de Chaves Existente).

5. **Escolha o local de seu armazenamento de chaves.**

   No caminho `c:\android`, escolha um local para o armazenamento de chaves, que deve terminar com a extensão `.keystore`. Por exemplo:

```
c:\android\dummies.keystore
```

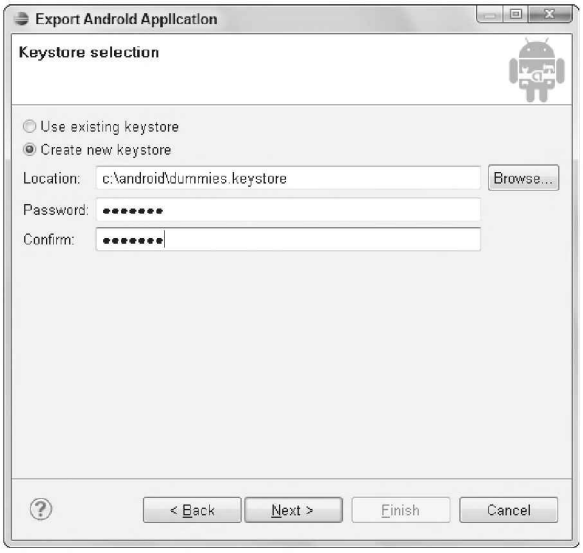


Figura 8-2: A tela Keystore Selection.

6. **Insira uma senha que você lembrará e insira-a novamente no campo Confirm (Confirmar).**

   Ao usar um armazenamento de chaves existente, você não precisará confirmar sua senha.

   Seu arquivo de armazenamento de chaves foi criado, mas agora você precisa criar uma chave.

7. **Clique no botão Next (Próximo).**

   A tela Key Creation (Criação da Chave) aparecerá, como mostrado na Figura 8-3.

Figura 8-3: A tela Key Creation.

8. **Preencha os campos a seguir:**
   - *Alias:* O apelido que você usa para identificar a chave.
   - *Password (Senha) e Confirm (Confirmar):* A senha que será usada para a chave.
   - *Validity (Validade):* Indica por quanto tempo a chave será válida. Sua chave deve expirar depois de 22 de outubro de 2033.
9. **Complete a seção de emissão do certificado, preenchendo pelo menos um dos seguintes campos:**
   - First and Last Name (Primeiro e Último Nome)
   - Organization Unit (Unidade da Organização)
   - Organization (Organização)
   - City or Locality (Cidade ou Localidade)
   - State or Province (Estado ou Província)
   - Country Code (Código do País) (XX)
10. **Clique no botão Next.**

    A tela final é Destination and Key/Certificate Checks (Destino e Verificações da Chave/Certificado), mostrada na Figura 8-4.

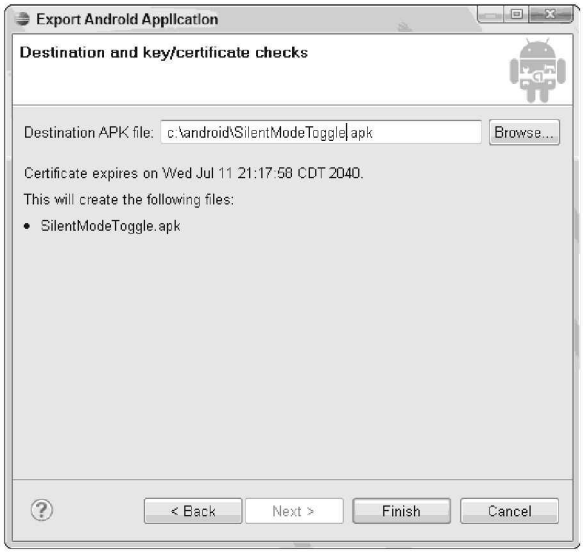


Figura 8-4: Escolhendo um nome e o destino de seu primeiro arquivo APK.

11. **Insira o local onde você deseja salvar o arquivo APK que contém seu aplicativo assinado.**

12. **Clique no botão Finish (Terminar).**

    O arquivo `.apk` é criado no local escolhido, assim como o armazenamento de chaves no local escolhido na Etapa 5. Abra esses locais e poderá ver um arquivo `.keystore`, assim como um arquivo `.apk`, como mostrado na Figura 8-5.

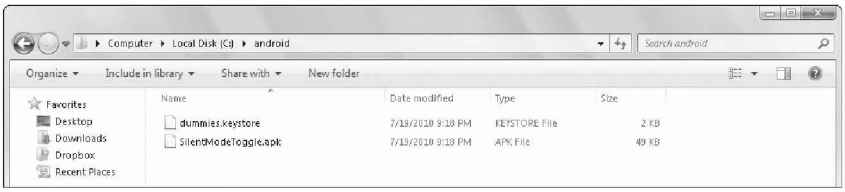


Figura 8-5: Fornecendo um destino para o arquivo APK.

Você criou um arquivo APK de distribuição e um armazenamento de chaves reutilizável para as futuras atualizações.

Criando um Perfil de Desenvolvedor Google Play

Após ter criado um arquivo APK, você poderá lançar o aplicativo na Google Play Store. Para tanto, criará um perfil de desenvolvedor Google Play. Para criar esse perfil, você precisará de uma conta Google. Qualquer conta baseada no Google, tal como uma conta Gmail, funcionará. Se você não tiver uma conta Google, poderá abrir uma gratuitamente em `www.google.com/accounts`.

Para criar o perfil de desenvolvedor Google Play, siga estas etapas:

1. **Abra seu navegador Web e navegue para** `http://play.google.com/apps/publish` **(conteúdo em inglês).**

2. **No lado direito da tela, acesse sua conta Google.**

   Se você já estiver autenticado em sua conta, vá para a Etapa 3 para preencher seu perfil do desenvolvedor.

3. **Preencha os seguintes campos para completar seu perfil de desenvolvedor, como mostrado na Figura 8-6:**

   - *Developer Name (Nome do Desenvolvedor):* O nome que aparece como o desenvolvedor dos aplicativos lançados, tais como o nome de sua empresa ou seu nome pessoal. Você pode mudá-lo mais tarde, após ter criado seu perfil de desenvolvedor.

- *E-mail Address (Endereço de E-mail):* O endereço de e-mail para o qual os usuários podem enviar e-mail com perguntas ou comentários sobre seu aplicativo.
- *Web Site:* A URL de seu site web. Se você não tiver um, tente um blog gratuito em `www.tumblr.com`.
- *Phone Number (Número de Telefone)*: Um número de telefone válido no qual contatar você para discutir problemas em seu conteúdo publicado.

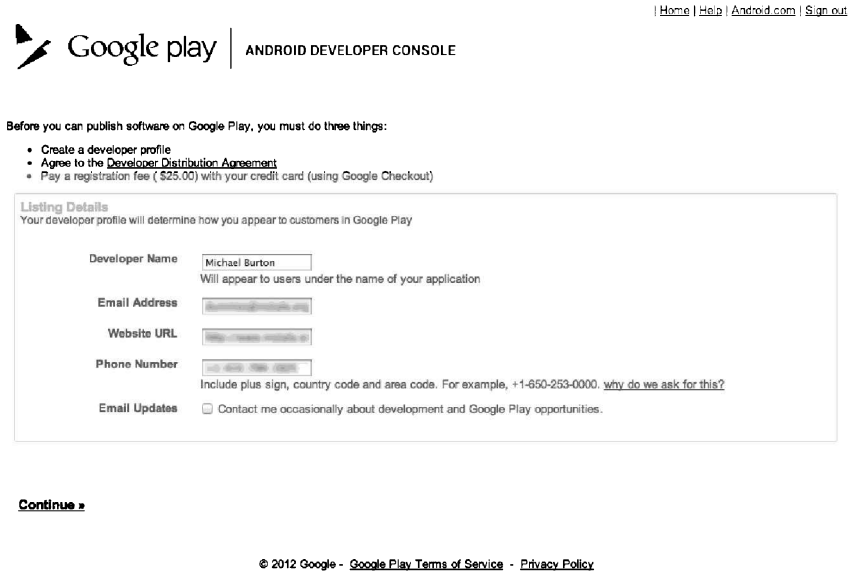


Figura 8-6: Detalhes do desenvolvedor listados.

4. **Clique no botão Continue (Continuar).**

   A página Android Developer Agreement (Acordo do Desenvolvedor Android) será aberta.

5. **Leia os termos, e então clique no link I Agree, Continue (Concordo, Continuar) para pagar a taxa de desenvolvedor através do Google Checkout (Verificação do Google).**

   Se você não pagar a taxa de desenvolvedor, não poderá publicar os aplicativos.

6. **Na página Secure Checkout (Pagamento Seguro), preencha os detalhes de seu cartão de crédito e informações do faturamento; depois, clique no botão Agree and Continue (Concordar e Continuar).**

Se você já tem um cartão de crédito cadastrado no Google, poderá não ver esta página. Se já tem uma configuração de cartão, selecione um e continue.

7. **Na página de confirmação do pedido, clique no botão Place Your Order Now (Fazer Seu Pedido Agora).**

   Dependendo da velocidade de sua conexão da Internet — e seu pedido — você poderá não ver a tela de carregamento.

   Quando o processo terminar, você verá uma mensagem confirmando que é um desenvolvedor Android. (Veja Figura 8-7.)

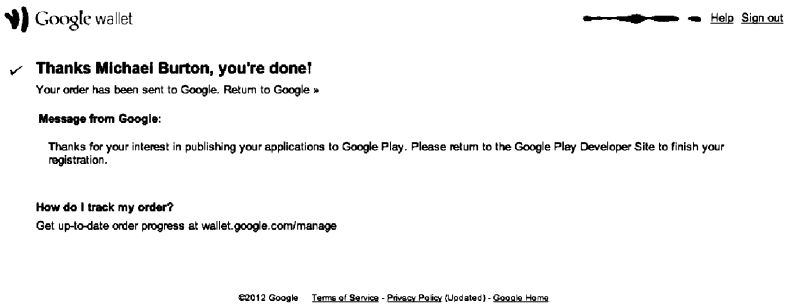


Figura 8-7: A confirmação de seu registro.

8. **Clique no link Google Play Store Developer Site (Site do Desenvolvedor na Google Play Store).**

   A home page do desenvolvedor Android será aberta, como mostrado na Figura 8-8, onde você poderá fazer o upload de seu aplicativo ou configurar uma conta comercial (que você precisará, se for cobrar por seus aplicativos). Veja a próxima seção complementar "Contas comerciais da Verificação do Google".

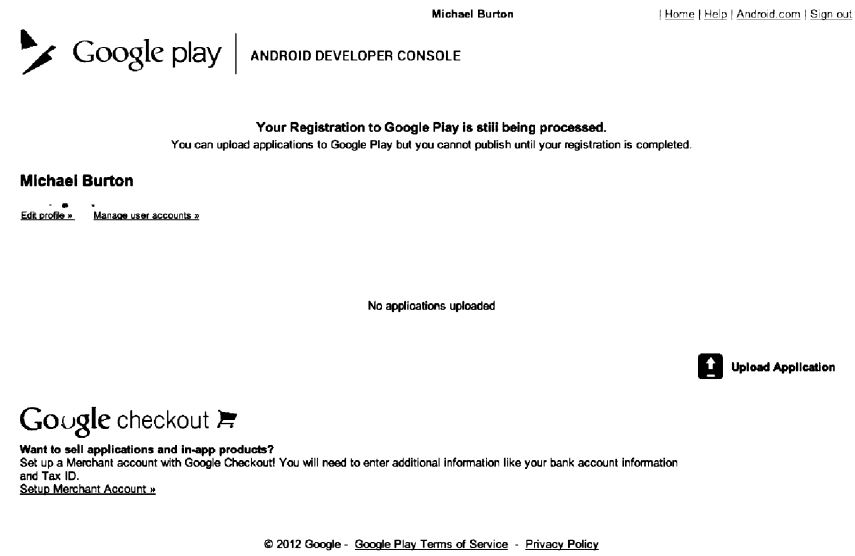


Figura 8-8: A home page do desenvolvedor Android.

Colocando Preço em Seu Aplicativo

Então, você criou um arquivo APK e é um desenvolvedor Android registrado. Agora, você está pronto para colocar seu aplicativo nas mãos dos usuários. (Finalmente!) Mas você deve responder a uma última pergunta — seu aplicativo é gratuito ou pago?

Contas comerciais da Verificação do Google

Para ter um aplicativo pago na Google Play Store, você deve configurar uma conta comercial Google Checkout. Para configurá-la, escolha Setup Merchant Account (Configurar Conta Comercial) na home page do desenvolvedor Android (consulte a Figura 8-8) e forneça estas informações:

- Nome pessoal e comercial
- Identidade da taxa (pessoal ou corporativa)
- Receita mensal esperada ($1 bilhão, certo?)

Após ter configurado uma conta comercial Google Checkout, você poderá vender seus aplicativos.

Tome esta decisão antes de lançar seu aplicativo, pois o preço tem consequências psicológicas para os clientes em potencial ou usuários, e consequências monetárias para você. Se o seu for pago, você terá que determinar a base de preço. Somente você pode tomar essa decisão, portanto, examine os aplicativos parecidos na Play Store e suas bases de preço para determinar sua estratégia. Como a maioria dos aplicativos tem preço variando entre $ 0,99 e $ 9,99, raramente você verá um com preço além de $ 10. Manter o preço de seu aplicativo competitivo com seu produto é um jogo financeiro que você tem que jogar para determinar o que funciona melhor para o seu aplicativo.

A discussão entre pago versus gratuito é um debate eterno e ambos os lados são rentáveis. Você apenas tem que descobrir o que funciona melhor para seu aplicativo, dada a situação.

Escolhendo o modelo pago

Se você escolher o modelo pago para seu aplicativo, geralmente começará a ver dinheiro em seu bolso 24 horas após a primeira venda (exceto feriados e finais de semana). Porém, seu aplicativo pago provavelmente não será instalado ativamente.

Os usuários que baixarem seu aplicativo na Google Play Store terão um período de experiência gratuito de 15 minutos para testar o aplicativo pago. Durante o período de experiência, os usuários poderão experimentar o aplicativo totalmente funcional e se não gostarem, simplesmente irão desinstalá-lo, recebendo um reembolso total. O período de experiência é extremamente útil porque os usuários não são penalizados por fazerem um pequeno teste de seu aplicativo.

Escolhendo o modelo gratuito

Se você escolher pegar a rota gratuita, os usuários poderão instalar o aplicativo gratuitamente. Entre 50 e 80 porcento dos usuários que instalam seu aplicativo gratuito o manterão no dispositivo; os outros irão desinstalá-lo. O grande problema agora é como ganhar dinheiro criando aplicativos gratuitos.

Como diz o velho ditado: nada na vida é de graça; e o ditado se aplica a ganhar dinheiro com aplicativos gratuitos. Você tem duas opções básicas:

- **Compras no aplicativo:** Você identifica diferentes "upgrades" que os usuários podem comprar ao usar seu aplicativo, que então são gerenciados via Google Play Store.
- **Publicidade:** Várias agências de publicidade móveis fornecem bibliotecas de terceiros para exibir anúncios em seu aplicativo móvel.

As principais empresas de publicidade móveis são a Google AdSense (`www.google.com/adsense`) e AdMob (`www.admob.com`). Obter uma conta gratuita em uma dessas empresas é bem simples. Elas oferecem SDKs úteis e orientam-no nas etapas para executar os anúncios em seus aplicativos Android nativos. A maioria paga um ciclo de 60 dias, portanto, você pode ter que esperar alguns meses para receber seu primeiro cheque.

Obtendo Capturas de Tela para Seu Aplicativo

As capturas de tela (screen shots) são uma parte vital do ecossistema Google Play Store porque permitem que os usuários visualizem um aplicativo antes de instalá-lo. Permitir que os usuários vejam algumas capturas de tela de seu aplicativo pode ser um fator determinante para instalar seu aplicativo. Se você passou semanas (ou meses) criando gráficos detalhados para um jogo que deseja que usuários joguem, gostará que os usuários e os compradores em potencial os vejam para que possam ter uma ideia geral de seu aplicativo.

Para capturar imagens de seu aplicativo em tempo real, você usará um emulador ou um dispositivo Android físico. Para obter capturas de tela com um emulador, siga estas etapas:

1. **Abra o emulador e coloque o componente na tela Inicial.**
2. **No Eclipse, abra a DDMS Perspective (Perspectiva DDMS).**

   Visite o Capítulo 5 para lembrar como usar o DDMS.
3. **Escolha o emulador no painel Devices (Dispositivos), como mostrado na Figura 8-9.**
4. **Clique no botão Screen Shot (Captura de Tela) para capturar uma tela.**

   Após a captura de tela ser feita, salve o arquivo em algum lugar em seu computador.

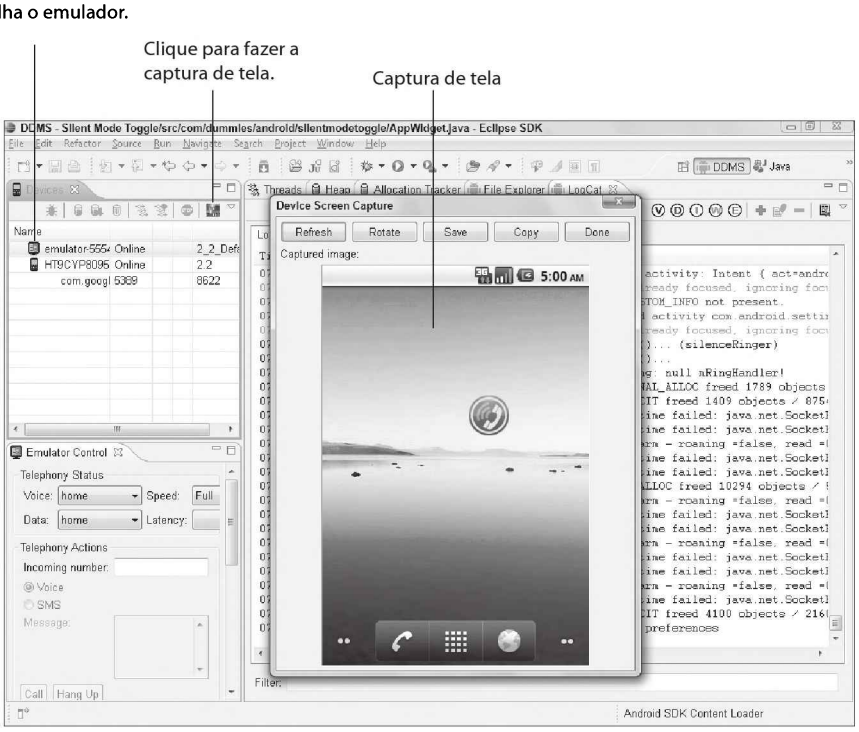


Figura 8-9: A perspectiva DDMS com a tela do emulador capturada.

Fazendo o Upload de Seu Aplicativo para a Google Play Store

Finalmente, você chegou ao ápice do desenvolvimento de aplicativos Android: Está pronto para publicar o aplicativo. Para isso, você precisará reunir as seguintes informações:

- O APK do aplicativo assinado
- As capturas de tela de seu aplicativo
- Uma descrição e um texto promocional de seu aplicativo
- Uma imagem promocional opcional usada para promover seu aplicativo, caso ele seja exibido na Google Play Store

Publicar um aplicativo é fácil — siga estas etapas:

1. **Na home page do desenvolvedor Android (consulte a Figura 8-8), clique no botão Upload Application (Fazer Upload do Aplicativo).**

   A página Upload an Application (Fazer Upload de um Aplicativo) será aberta, como mostrado na Figura 8-10.

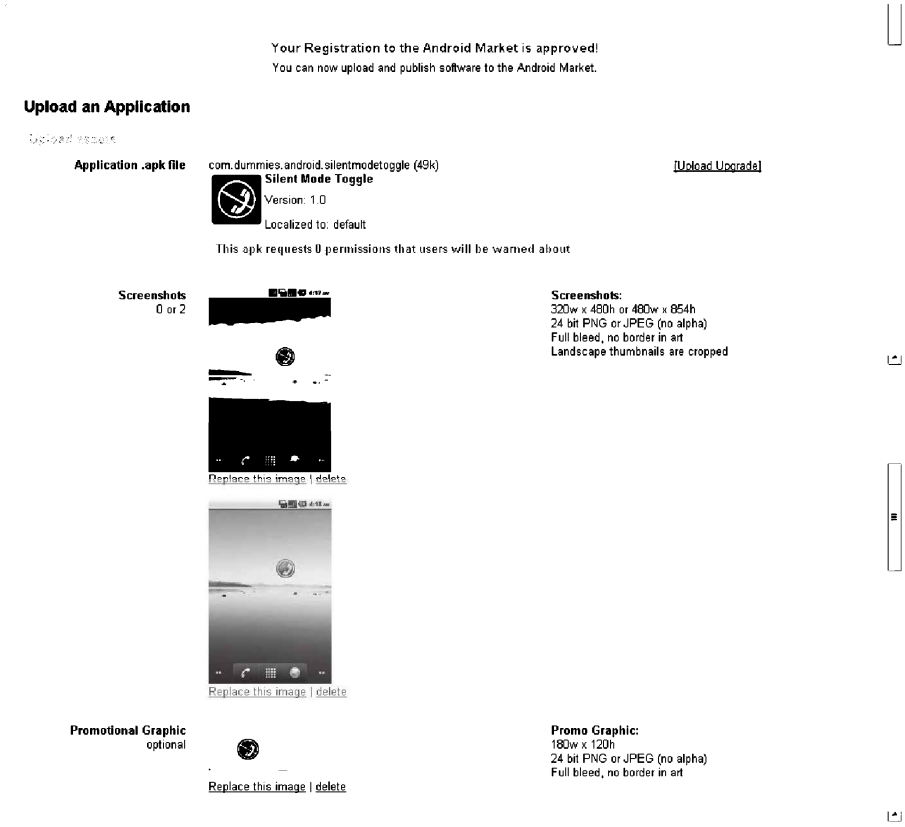


Figura 8-10: A página de upload.

2. **Para Application APK file (arquivo APK do Aplicativo), escolha o arquivo `.apk` que você criou anteriormente neste capítulo, e então clique em Upload.**

Dois aplicativos não podem ter o mesmo nome de pacote na Google Play Store (o Google usa o nome do pacote Java como o identificador). Portanto, se você tentar fazer o upload do aplicativo Silent Mode Toggle neste ponto, verá esta mensagem de erro:

```
The package name of your apk (com.dummies.android.silentmodetoggle) is the
          same as the package name of another developer's application.
          Choose a new package name.
```

Quando você fizer o upload de um aplicativo que criou, não verá esta mensagem.

3. **Na seção Screenshots (Capturas de tela), adicione duas capturas de seu aplicativo.**

   Os aplicativos com capturas de tela têm taxas de instalação mais altas do que os aplicativos sem elas. Essas capturas de tela permitem que os usuários visualizem seu aplicativo em um estado de execução sem ter que instalá-lo.

4. **Adicione uma captura promocional.**

   A captura promocional não é uma captura de tela, mas um anúncio utilizado para as promoções aleatórias que o Android escolha exibir. Uma captura promocional não é obrigatória para publicar o aplicativo.

5. **Defina o título de seu aplicativo.**

   Escolha um título adequado para seu aplicativo. Este texto é indexado para o motor de pesquisa da Google Play Store.

6. **Defina a descrição de seu aplicativo.**

   Os usuários veem esta descrição quando examinam seu aplicativo para determinar se devem instalá-lo. Todo esse texto é indexado para o motor de pesquisa da Google Play Store.

7. **Defina o texto promocional de seu aplicativo.**

   O *texto promocional* é usado quando seu aplicativo é exibido ou promovido na Google Play Store. Ter seu aplicativo exibido provavelmente é baseado na popularidade dele. Se ele for escolhido para ser exibido na área promocional da Google Play Store (geralmente na área superior da tela de cada categoria), este texto aparecerá como o componente promocional dele.

8. **Defina o tipo de aplicativo.**

   Este aplicativo é do tipo Applications (Aplicativos); se você tiver um aplicativo de jogo, escolha o tipo Games (Jogos).

9. **Defina a categoria do aplicativo.**

   A categoria é baseada em seu tipo de aplicativo.

10. **Desative a proteção contra cópias.**

   A proteção contra cópias impede que seu aplicativo seja copiado ilegalmente para outros dispositivos. Para evitar isso, use o serviço Google Play Licensing (Licenciamento Google Play) em `http://developer.android.com/google/play/licensing/index.html` (conteúdo em inglês).

11. **Selecione a lista de locais onde o aplicativo deve ficar visível.**

    Por exemplo, se seu aplicativo tiver como alvo o público italiano, cancele a seleção de All Locations (Todos os Locais) e selecione Italy (Itália) como o local de destino, para assegurar que apenas os dispositivos na região da Itália possam vê-lo na loja. Se você deixar All Locations ativado, todos os locais poderão ver seu aplicativo na loja.

12. **Preencha os campos Web Site (Site) e E-mail (e Phone — Telefone —, se quiser).**

    Estes campos são usados para contactá-lo por vários motivos, inclusive solicitações de novas funções para o aplicativo e relatórios de erros encontrados. Se você preencher o campo Phone, lembre-se de que os usuários poderão ligar para você. Se estiver escrevendo um aplicativo para uma empresa e publicando-o em sua conta de desenvolvedor, poderá mudar os campos Web Site, E-mail e Phone para que os usuários não possam contactá-lo.

13. **Verifique se seu aplicativo atende às diretrizes de conteúdo do Android e se você cumpre as leis aplicáveis, selecionando as caixas de seleção pertinentes.**

14. **Escolha uma destas opções:**

    - *Publish (Publicar):* Salva e publica o aplicativo na loja em tempo real. A página Upload an Application (Fazer Upload de um Aplicativo) será aberta (consulte a Figura 8-10).
    - *Save (Salvar):* Salva as alterações, mas não publica o aplicativo. Seu aplicativo é mostrado como salvo na home page do desenvolvedor Android, como exibido na Figura 8-11. Quando você estiver pronto para publicar seu aplicativo, selecione o título e clique no botão Upload Application.

    Você também pode escolher Delete (Apagar) neste momento, mas provavelmente não desejará. Você apagará todo o seu trabalho.

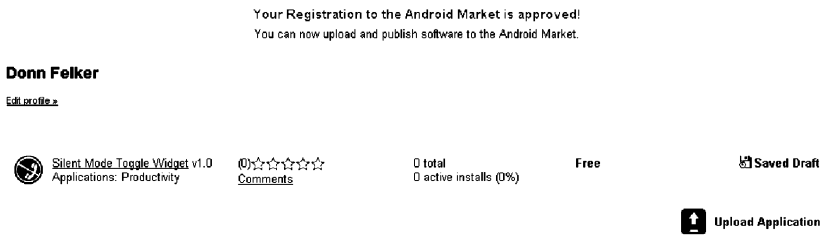


Figura 8-11: O aplicativo salvo na tela Inicial do desenvolvedor Android.

15. Vá para a parte inferior da tela e clique no botão Publish (Publicar).

Seu aplicativo será publicado na Google Play Store.

A Figura 8-12 mostra um aplicativo na Google Play Store em um dispositivo Nexus One.

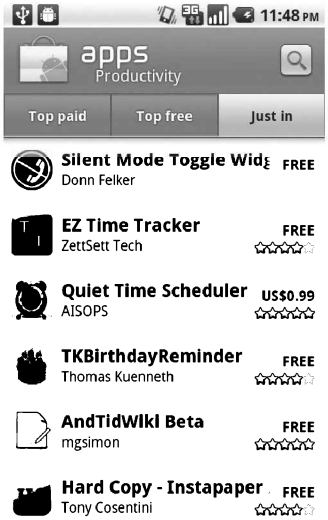


Figura 8-12: O aplicativo é lançado na Google Play Store.

Provavelmente, você notou um certo destaque no processo: não há nenhum processo de aprovação do aplicativo (como uma certa plataforma faz). Você pode criar um aplicativo agora, publicá-lo, e os usuários poderão instalá-lo imediatamente. Então, você poderá completar um ciclo de lançamento rápido e publicar novos recursos tão rapidamente quanto os conclui — muito legal.

DICA

Se você precisar remover seu aplicativo da Google Play Store, selecione o título do aplicativo na tela Inicial do desenvolvedor Android, vá para a parte inferior e clique no botão Unpublish (Cancelar Publicação).

Observando o Número de Instalações Decolar

Finalmente, você publicou seu primeiro aplicativo. Agora, é hora de ver os milhões começarem a chegar, certo? Mais ou menos. Você pode ser um desenvolvedor independente que está lançando o próximo grande jogo de tiros em primeira pessoa ou pode ser um desenvolvedor corporativo que está publicando o aplicativo Android de sua empresa. De qualquer modo, para conhecer a experiência do usuário em vários dispositivos, você pode identificar como seu aplicativo está se saindo de diversas maneiras:

- **Sistema de classificação com cinco estrelas:** A classificação média mais alta que seu aplicativo pode receber, a melhor.
- **Comentários:** Faça às pessoas a cortesia de ler os comentários que elas deixam. Você pode ficar surpreso com as excelentes ideias que elas fornecem gratuitamente. Os usuários ficam animados com os novos recursos e retornam à loja para atualizar seus comentários com classificações cada vez mais positivas.
- **Relatórios de erros:** Os usuários que foram generosos o bastante para enviar relatórios de erros desejam que você saiba que o aplicativo gerou uma exceção de execução por um motivo desconhecido. Abra esses relatórios, examine o erro, reveja o rastreamento de erros e corrija o problema. Um aplicativo que é encerrado à força com frequência pode receber rapidamente muitas críticas ruins. Os rastreamentos da pilha estão disponíveis apenas para os dispositivos executados no Android 2.2 e posterior.
- **Instalações versus instalações ativas:** Embora esta comparação não seja a melhor métrica para identificar a satisfação do usuário, é um modo não científico para determinar se os usuários que instalam seu aplicativo tenderão a mantê-lo em seus dispositivos. Os usuários que mantêm seu aplicativo obviamente gostam dele.
- **E-mail direto:** Os usuários retornarão à Google Play Store para encontrar seu endereço de e-mail ou endereço do site web, fazer perguntas sobre os recursos ou enviar comentários sobre sua experiência. Eles também podem enviar ideias sobre como melhorar seu aplicativo ou pedir que você crie outro aplicativo que faça algo que eles não podem encontrar na Google Play Store. Responda se você tiver tempo! Embora manter um diálogo ativo com os usuários seja difícil, caso seu aplicativo tenha um milhão de usuários, isto irá deixá-los felizes ao saberem que podem contar com você sobre os problemas em seu aplicativo.

Manter contato com sua base de usuários é uma árdua tarefa em si, mas fazer isso pode obter a recompensa de clientes dedicados e contentes, que encaminham seus amigos e família ao seu aplicativo.

Como a Google Play Store, a Amazon App Store para o Android (uma das maiores lojas de aplicativos não Google para os dispositivos Android) oferece aplicativos para os usuários comprarem e instalarem. Os desenvolvedores podem vender e receber uma avaliação competitiva de seus aplicativos da Amazon ou enviar aplicativos gratuitos. A Amazon também fornece ótimas avaliações de vendas para desenvolvedores e negociantes. Descubra mais em `http://developer.amazon.com` (conteúdo em inglês). Você poderá descobrir como portar seu aplicativo para a Amazon App Store no Capítulo 18.

Parte III

Criando um Aplicativo Cheio de Recursos

"Como desenvolvedor de aplicativos, eu nunca pensei em dizer isto, mas seu aplicativo precisa de mais complementos."

Nesta parte...

A Parte III expande o conhecimento que você adquiriu na Parte II, demonstrando como pode construir um aplicativo cheio de recursos. Há também alguns tópicos avançados que podem ajudar a diminuir a distância entre o desenvolvedor Android iniciante e o avançado.

Nesta parte, você criará certos recursos para melhorar a experiência dos usuários com seu aplicativo. No final da Parte III, você terá um aplicativo avançado, totalmente funcional e que interage com um banco de dados local com as preferências do cliente.

Capítulo 9

Projetando o Aplicativo Task Reminder

Neste Capítulo

Listando os requisitos do aplicativo
Desenvolvendo diversas telas
Construindo uma atividade de lista
Trabalhando com intenções

Construir aplicativos Android é divertido, mas construir aplicativos realmente robustos é animador porque você entra totalmente na plataforma Android. Este capítulo apresenta o aplicativo Task Reminder, que você construirá nos próximos capítulos.

O aplicativo Task Reminder permite que os usuários criem uma lista de itens que têm uma hora de lembrete associada a cada um deles.

Revisando os Requisitos Básicos

O aplicativo Task Reminder tem alguns requisitos básicos para que ele possa atender o que é esperado dele:

- O aplicativo deve ser capaz de aceitar a entrada do usuário (ter um aplicativo de tarefas personalizado que não permite a entrada do usuário seria estúpido!).
- As tarefas devem ser fáceis de gerenciar.
- Toda tarefa deve ter uma data e hora de lembrete para que o usuário seja notificado.
- O usuário deve ser notificado sobre a tarefa quando chegar a hora do lembrete.
- Os usuários devem ser capazes de apagar as tarefas.
- Os usuários devem ser capazes de não apenas adicionar tarefas, mas também editá-las.

Este aplicativo pede muita interação entre o usuário e o sistema Android. As seções seguintes entram nos recursos que você precisará para construir o aplicativo para dar aos usuários toda a funcionalidade necessária.

Agendando um script de lembrete (É alarmante!)

Para o aplicativo Task Reminder funcionar bem, você precisa implementar um sistema baseado em lembretes. A primeira coisa que vem à mente é uma tarefa agendada ou serviço `cron`. No sistema operacional Windows, você cria uma tarefa agendada para executar código e scripts em um certo momento. No mundo do Unix e do Linux, você usa `cron` (abreviação da palavra grega chronos, que significa tempo) para agendar os scripts ou os aplicativos.

Como o Android é executado no kernel Linux, você pode supor que o Android usa o `cron` para agendar as tarefas. Infelizmente, o Android não usa o cron; porém, ele tem a classe `AlarmManager`, que realiza a mesma tarefa. A classe `AlarmManager` permite especificar quando seu aplicativo deve iniciar. Um alarme pode ser definido como de um único uso ou de repetição. O aplicativo Task Reminder usa `AlarmManager` para lembrar os usuários sobre as tarefas pendentes.

Armazenando dados

Todas as atividades, dados de tarefa e alarmes necessários para fazer o Task Reminder funcionar estão armazenados neste locais:

- **Atividades e receptores de transmissão:** Em um único pacote Java
- **Dados da tarefa:** Em um `ContentProvider` suportado por um banco de dados SQLite
- **Informações do alarme:** Em `AlarmManager` via sistema de intenções depois de ser obtido no `ContentProvider`

Distraindo o usuário (gentilmente)

Depois que um alarme é acionado, o aplicativo tem que notificar o usuário sobre ele. A plataforma Android fornece mecanismos para trazer sua atividade para o primeiro plano quando o alarme inicia, mas este não é um método de notificação ideal porque rouba o foco daquilo que o usuário está fazendo no momento. Imagine se o usuário estiver discando um número de telefone ou respondendo a uma ligação e um alarme iniciar trazendo uma atividade para o primeiro plano. Provavelmente, o usuário ficará confuso porque uma atividade iniciou sem ser acionada manualmente.

Você tem dois modos de obter a atenção do usuário sem roubar o foco principal da atividade atual:

- **Toast:** Uma pequena exibição que contém uma breve mensagem para o usuário. A mensagem não permanece visível porque geralmente está disponível por apenas alguns segundos — um toast nunca recebe o foco. O aplicativo Task Reminder usa um toast não para *lembrar* o usuário, mas usa-o para notificá-lo quando sua atividade foi salva.
- **Gerenciador de Notificações:** A classe `NotificationManager` notifica um usuário quando ocorrem eventos. Eles podem aparecer na barra de status no topo da tela. Os itens de notificação podem conter várias exibições e são identificados pelos ícones fornecidos. O usuário rolar para baixo a lista de notificações para exibi-las. O aplicativo Task Reminder usa a classe `NotificationManager` para lidar com os alarmes (veja o Capítulo 1 se não souber como a área de notificação funciona).

Você pode obter a atenção de um usuário exibindo uma caixa de diálogo que rouba imediatamente o foco do aplicativo em execução. Apesar de sua eficácia, o usuário pode ficar irritado porque seu aplicativo está roubando o foco (possivelmente de modo contínuo para vários lembretes) da ação atual em outro aplicativo.

Criando as Telas do Aplicativo

O aplicativo Task Reminder precisa de duas telas diferentes para realizar todas as suas funções básicas com as tarefas: criar, ler, atualizar e excluir (CRUD, do inglês Create, Read, Update e Delete).

- Uma exibição lista todas as tarefas atuais no aplicativo. Essa exibição também permite ao usuário apagar uma tarefa pressionando-a por um certo tempo.
- Uma exibição para permitir que o usuário exiba (leia), adicione (crie) ou edite (atualize) uma tarefa.

Cada tela interage consequentemente com um banco de dados para que as alterações sejam mantidas no uso de longo prazo do aplicativo.

Cada tela consiste em um único fragmento de código que contém a maior parte da interface de usuário para a tela e esse fragmento está contido em uma atividade.

Você poderá reutilizar os fragmentos se — ou *quando* — construir o suporte para tablets em seu aplicativo. Veja a Parte IV para conhecer o desenvolvimento para tablets.

Iniciando o novo projeto

Para iniciar, abra o Eclipse e crie um novo projeto Android com um nome válido, pacote e atividade. A Tabela 9-1 mostra as definições do Eclipse para o aplicativo Task Reminder (se você não estiver familiarizado com a criação de um projeto Android, veja o Capítulo 3).

Se você fizer download do código-fonte no site web deste livro, poderá também abrir o exemplo do projeto Android do Capítulo 9.

Tabela 9-1	Definições do novo projeto
Propriedade	***Valor***
Project Name (Nome do Projeto)	Task Reminder
Build Target (Destino da Construção)	Android 4.0.3 (API Nível 15)
Application Name (Nome do Aplicativo)	Task Reminder
Package Name (Nome do Pacote)	`com.dummies.android.taskreminder`
Create Activity (Criar Atividade)	`ReminderListActivity`
Min SDK Version (Versão SDK Mínima)	4

Note o valor da propriedade Create Activity — `ReminderListActivity`. Normalmente, você dá à primeira atividade em um aplicativo o nome `MainActivity`; porém, a primeira tela que o usuário vê no aplicativo Task Reminder é uma lista das tarefas atuais. Portanto, essa atividade é uma instância de `ListActivity`; daí o nome `ReminderListActivity`.

O aplicativo Task Reminder usa recursos da Biblioteca de Suporte Android para suportar os dispositivos executados no Android 2.x e anteriores. Adicione a biblioteca ao seu projeto Eclipse seguindo estas etapas:

1. **Copie o arquivo** `android-support-v13.jar` **para o diretório libs de seu projeto.**

   Está em `ANDROID-SDK/extras/android/support/v13`.

   Se não puder encontrar o arquivo `android-support-v13.jar` em seu diretório SDK do Android, você pode não ter instalado ainda a biblioteca de suporte. Abra o Android SDK Manager (Gerenciador SDK do Android) e clique em Extras para instalar a biblioteca de suporte a partir dele.

2. **Escolha Project (Projeto)⇨Clean (Limpar) no menu Eclipse.**

   Assim que o projeto terminar a reconstrução, sua listagem Android Dependencies (Dependências do Android) deverá parecer com a mostrada na Figura 9-1.

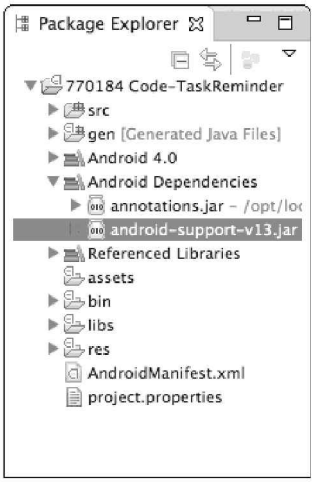


Figura 9-1: O arquivo android--support-v13.jar em sua listagem de projetos.

Criando a ReminderListActivity

A classe `ReminderListActivity` que o Eclipse gerou para você está praticamente vazia, portanto, você desejará fazer algumas alterações nela. Faça o seguinte:

- **Renomeie o arquivo** `activity_main.xml` **como** `reminder_list.xml`. O Eclipse inicia seu projeto com o arquivo `activity_main.xml` localizado no diretório `res/layout`. Para facilitar a localização de seu arquivo de layout ao abrir o diretório, renomeie-o para algo mais informativo. Para renomear o arquivo `activity_main.xml`, clique-o com o botão direito e escolha Refactor (Modificar)⇨Rename (Renomear) ou selecione o arquivo e pressione Shift+Alt+R.
- **Atualize o arquivo Java**. Depois de mudar o nome de arquivo, você precisará atualizar o nome na chamada `setContentView()` dentro do arquivo `ReminderListActivity.java`. Abra o arquivo e substitua `R.layout.activity_main` por `R.layout.reminder_list`.
- **Mude a herança.** Como `ReminderListActivity` contém fragmentos, ele precisa herdar da classe `FragmentActivity`, ao invés da atividade básica regular. Faça essa alteração também.

Sua nova classe `ReminderListActivity` agora se parece com a Listagem 9-1.

Listagem 9-1: A classe ReminderListActivity

```
public class ReminderListActivity extends FragmentActivity {
    /** Chamada quando a atividade é criada pela primeira vez. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.reminder_list);
    }
}
```

`setContentView()` usa o arquivo de layout `reminder_list`, mas você não definiu isso ainda. Abra o arquivo `res/layout/reminder_list.xml` e atualize-o para parecer com a Listagem 9-2.

Listagem 9-2: O conteúdo de reminder_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.dummies.android.taskreminder.ReminderListFragment"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

O arquivo de layout para sua atividade tem apenas um único elemento, `ReminderListFragment`, que ocupa a altura e a largura total da tela. `ReminderListFragment` faz todo o trabalho pesado para exibir a lista de tarefas para o usuário.

Criando o ReminderListFragment

Fragmentos são as partes de suas atividades que servem para ser reutilizadas em seu aplicativo. A maioria das atividades tem um ou dois fragmentos. A atividade da lista precisa de um fragmento para exibir a lista de tarefas, portanto, crie um novo arquivo chamado `ReminderListFragment` e copie o código na Listagem 9-3.

Listagem 9-3: O fragmento ReminderList

```
package com.dummies.android.taskreminder;

import android.os.Bundle;
import android.support.v4.app.ListFragment;                    → 4
import android.view.View;

public class ReminderListFragment extends ListFragment {

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {   →10
        super.onActivityCreated(savedInstanceState);
        setEmptyText(getResources().getString(R.string.no_reminders));   → 12
    }

}
```

Sabendo quando usar atividades ou fragmentos

As atividades e os fragmentos são partes centrais do código de sua interface de usuário (IU). Portanto, como você decide se é para colocar certa funcionalidade em um fragmento ou uma atividade?

Se as atividades são a lancheira do código IU, os fragmentos são o Tupperware. Você pode inserir seu código IU diretamente em sua lancheira, mas ficaria um pouco bagunçado e tornaria o código da lancheira difícil de reutilizar. Coloque seu código IU no Tupperware de seu fragmento, onde você pode movê-lo entre as lancheira quando precisar usá-lo de novo.

Se você estiver absolutamente seguro de que o código que está escrevendo é específico de certa atividade, coloque-o diretamente na atividade. Mas se não estiver certo, coloque seu código IU em um fragmento. Na maioria dos aplicativos, os fragmentos contêm todo seu código IU e suas atividades contêm apenas a cola que une os fragmentos.

Eis uma rápida explicação do código na Listagem 9-3:

→ **4** Esta linha assegura que você está usando as importações `android.support.v4.app.*` e não seus equivalentes do `android.app`.

Importe `android.support.v4.app.Fragment` e `android-support.v4.app.FragmentTransaction`, não seus equivalentes do `android.app`. Usar as classes `android.support.v4.*` da Biblioteca de Suporte Android assegura que seu aplicativo funcionará nos dispositivos com as versões do Android até v4 (Android 1.6). Se você não se importar com as versões do Android anteriores a 3.x, sinta-se à vontade para dispensar a biblioteca de suporte.

→ **10** O método `onCreate()` da atividade (não do fragmento) retorna e chama o callback `onActivityCreated()`.

Se você precisar fazer algo com as exibições do fragmento, `onActivityCreated()` será um ótimo lugar para fazer isso porque as exibições de seu fragmento estarão certamente construídas nesse ponto.

→ **12** `ListFragment` suporta exibir uma mensagem quando a lista está vazia. A chamada usa um valor `setEmptyText()` e a mensagem usa um valor `R.string.no_reminders`.

Adicione `<string name="no_reminders">No Reminders Yet</string>` ao seu arquivo `strings.xml`. Veja o Capítulo 6 para obter mais informações sobre como adicionar strings ao seu arquivo `strings.xml`.

Usando uma atividade para criar e editar lembretes

O aplicativo Task Reminder precisa de uma tela adicional que permite ao usuário editar uma tarefa e suas informações. Essa nova atividade e fragmento permitirão que os usuários criem, leiam e atualizem as tarefas.

No Eclipse, siga estas etapas:

1. **Crie uma nova atividade que possa lidar com os papéis de criação, leitura e atualização.**

   Clique com o botão direito no nome do pacote na pasta `src` e escolha New (Nova)⇨Class (Classe) ou pressione Shift+Alt+N, e então escolha Class. Nomeie-a como `ReminderEditActivity`.

2. **Na janela de nova classe Java, defina a superclasse para `android.support.v4.app.FragmentActivity` e clique em Finish (Terminar).**

   Um novo arquivo de classe da atividade será aberto.

3. **Substitua o conteúdo do arquivo de classe da atividade pela Listagem 9-4.**

   Este código cria a atividade, define sua exibição do conteúdo, e então configura o fragmento da atividade.

Listagem 9-4: ReminderEditActivity

```
package com.dummies.android.taskreminder;

import android.os.Bundle;
import android.support.v4.app.Fragment;                                  → 3
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentTransaction;

public class ReminderEditActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.reminder_edit_activity);                 → 11

        Fragment fragment = getSupportFragmentManager().findFragmentByTag(
                ReminderEditFragment.DEFAULT_EDIT_FRAGMENT_TAG);         → 15

        if (fragment == null) {                                          → 17
            fragment = new ReminderEditFragment();
            Bundle args = new Bundle();
            args.putExtra(ReminderProvider.COLUMN_ROWID, getIntent()
```

```
                .getLongExtra(ReminderProvider.COLUMN_ROWID, 0L));      →21
        fragment.setArguments(args);                                    →22

        FragmentTransaction transaction = getSupportFragmentManager()
                .beginTransaction();                                    →25
        transaction.add(R.id.edit_container, fragment,
                ReminderEditFragment.DEFAULT_EDIT_FRAGMENT_TAG);        →27
        transaction.commit();                                           →28
    }
  }
}
```

Eis uma pequena explicação do código na Listagem 9-4:

→ 3 Esta linha assegura que você está usando as importações `android.support.v4.app.*`, e não seus equivalentes do `android.app`.

→ 11 Esta linha define o layout para este fragmento. A Listagem 9-5 mostra o código para `R.layout.reminder_edit_activity`.

→ 15 Antes de um fragmento ser adicionado pela primeira vez, esta linha verifica se um já existe.

Se uma atividade for recriada a partir de outra atividade — digamos, após uma uma rotação da tela — o fragmento anterior também teria que ser recriado e seria usado, ao invés de criar um a partir de zero.

→ 17 Se a atividade não puder encontrar um fragmento anterior, esta linha criará um novo.

→ 21 As intenções podem ter extras, permitindo que as atividades transmitam informações entre si. Na linha 21, a intenção usa `getLongExtra()` para recuperar o `long` denominado `COLUMN_ROWID`, caso exista. Se não, a intenção usará o valor `0L` (ou `0` como um `long`).

→ 22 Os fragmentos precisam de argumentos. Diferentemente das classes Java normais, você não pode transmitir argumentos para um fragmento via construtor. Ao contrário, a linha 22 usa um objeto Bundle chamado `Fragment.setArguments()`.

→ 25 Sempre que você quiser interagir com um fragmento, deve usar `FragmentTransaction`. Esta linha chama `FragmentActivity. getSupportFragmentManager()` para obter o `FragmentManager` e a partir dele, chama `FragmentManager.beginTransaction()` para iniciar uma transação. Todos os fragmentos operam entre as chamadas `FragmentTransaction.beginTransaction()` e `FragmentTransaction.commit()`.

→ **27** Esta linha adiciona o fragmento à atividade. Ela coloca o fragmento em um espaço reservado `FrameLayout` chamado `R.id.edit_container` (que você definirá na Listagem 9-5) e nomeia-o como `DEFAULT_EDIT_FRAGMENT_TAG` para que o aplicativo possa encontrá-lo novamente pelo nome.

→ **28** Esta linha termina a transação.

Você pode notar que o fragmento está configurado de modo diferente nesta classe, em relação à classe `ReminderListActivity`. Em `ReminderListActivity`, o elemento `<fragment>` é adicionado diretamente ao XML porque o fragmento não precisava de parâmetros. A atividade `ReminderEditFragment` precisa que o fragmento seja adicionado manualmente usando o código Java para transmitir o ID do lembrete.

O arquivo de layout `reminder_edit_activity.xml` mostrado na Listagem 9-5 consiste em um único elemento de espaço reservado de tela cheia chamado `edit_container` que o código Java na Listagem 9-1 usa para anexar `ReminderListFragment`.

Listagem 9-5: R.layout.reminder_edit_activity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

    <FrameLayout
            android:id="@+id/edit_container"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
        />

</LinearLayout>
```

`ReminderEditActivity` usa uma constante em uma classe chamada `ReminderProvider`, que não existe ainda, portanto, crie essa classe agora e edite-a como a seguir:

```
package com.dummies.android.taskreminder;

public class ReminderProvider {
    public static final String COLUMN_ROWID = "_id";
}
```

Você também precisa informar à plataforma Android sobre a existência de `ReminderEditActivity` adicionando-a ao Manifest do Android. Você pode fazer isso adicionando-a ao elemento `Application` do arquivo `AndroidManifest.xml`, como mostrado aqui em negrito:

```
<application android:icon="@drawable/ic_launcher" android:label="@string/
          app_name">
    <activity android:name=".ReminderListActivity"
              android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ReminderEditActivity"
              android:label="@string/app_name" />
</application>
```

Se você não adicionar a atividade ao arquivo `AndroidManifest.xml`, receberá uma exceção de execução informando que o Android não pode encontrar a classe (a atividade).

Adicionando um fragmento à atividade

Depois de ter criado uma atividade para manter `ReminderEditFragment`, é hora de criar o fragmento. Crie uma nova classe Java, nomeie-a como **ReminderEditFragment** e copie o seguinte código para o arquivo:

```
package com.dummies.android.taskreminder;

import android.os.Bundle;
import android.support.v4.app.Fragment;                                  →4
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

public class ReminderEditFragment extends Fragment {
    public static final String DEFAULT_EDIT_FRAGMENT_TAG = "editFragmentTag";→12

    private EditText mTitleText;
    private EditText mBodyText;
    private Button mDateButton;
    private Button mTimeButton;
    private Button mConfirmButton;
    private long mRowId;                                                 →19

    @Override
    public void onCreate(Bundle savedInstanceState) {                    → 22
        super.onCreate(savedInstanceState);

        Bundle arguments = getArguments();                               → 25
        if (arguments != null) {
            mRowId = arguments.getLong(ReminderProvider.COLUMN_ROWID);
        }
    }
```

(Continua)

(Continuação)

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {                                  → 33

        View v = inflater.inflate(R.layout.reminder_edit, container, false);→35

        mTitleText = (EditText) v.findViewById(R.id.title);

          mBodyText = (EditText) v.findViewById(R.id.body);
          mDateButton = (Button) v.findViewById(R.id.reminder_date);
          mTimeButton = (Button) v.findViewById(R.id.reminder_time);
          mConfirmButton = (Button) v.findViewById(R.id.confirm);

          return v;
    }
}
```

Eis como o código funciona:

→ **4** Certifica-se de que está usando as importações `android.support.v4.app.*`, não seus equivalentes do `android.app`.

→ **12** As atividades precisam de uma tag ou um ID para referenciar quando elas trabalham com fragmentos. Esta linha fornece a `ReminderEditFragment` uma tag para que possa ser encontrada novamente mais tarde.

→ **19** Toda instância de `ReminderEditFragment` tem um ID para o lembrete. Row ID (ID da Linha) corresponde a uma linha no banco de dados. Ao editar os lembretes existentes, `mRowId` é o ID da linha no banco de dados para esse lembrete. Os novos lembretes obtêm um `mRowId 0`.

→ **22** Os fragmentos têm métodos `onCreate()`, exatamente como as atividades. `OnCreate()` é chamado quando o fragmento é criado e você geralmente faz a maior parte de sua inicialização em `onCreate`.

Diferentemente das atividades, porém, você não faz a inicialização relacionada às exibições em `onCreate`. Elas têm que aguardar até `onCreateView()` na linha 33.

→ **25** O fragmento descobre qual lembrete o usuário está editando ou criando chamando `getArguments()`.

Os argumentos vêm do objeto Bundle retornado por `getArguments()`, não do Bundle transmitido para `onCreate()` (um erro fácil de cometer).

→ **33** Diferentemente das atividades, você inicia seus layouts XML em `Fragment.onCreateView()`, ao invés de usar `Activity.setContentView()`. Esta linha cria o layout `R.layout`.

`reminder _edit`, e então chama `findViewById()` para configurar os objetos `View`, muito parecido com o que você faria ao inicializar uma atividade.

Você poderia configurar os objetos `View` em `onActivityCreated()`, mas é conveniente fazer isso em `onCreateView()` neste caso porque você não está manipulando as exibições. Veja a seção complementar "O ciclo de vida do fragmento" para obter mais informações sobre `onActivityCreated()` versus `onCreateView()`.

→ **35** Esta linha chama `inflate()` com `attachToRoot` definido para `false`, pois o fragmento anexa a exibição.

O ciclo de vida do fragmento

Como as atividades (veja Capítulo 5), os fragmentos têm seu próprio ciclo de vida.

Como uma atividade, um fragmento pode existir em três estados:

- **Resumed (Retomado):** O fragmento é visível na atividade em execução.
- **Paused (Pausado):** Outra atividade está em primeiro plano e tem o foco, mas a atividade na qual reside este fragmento ainda é visível (a atividade em primeiro plano está parcialmente transparente ou não cobre a tela inteira).
- **Stopped (Parado):** O fragmento não está visível. A atividade host foi parada ou o fragmento foi removido da atividade, mas adicionado à pilha. Um fragmento parado ainda está ativo (todas as informações do estado e do membro são mantidas pelo sistema). Porém, não está mais visível para o usuário e será encerrado se a atividade for encerrada.

Grande parte dos callbacks do fragmento é muito parecida com os callbacks da atividade. Contudo, existem algumas diferenças importantes. Os três callbacks de fragmento mais comuns são:

- `onCreate`: Diferentemente de uma atividade, os fragmentos não têm um método `setContentView()`. Diferentemente das atividades, as exibições não são criadas no método `onCreate()` de um fragmento, portanto, não há nenhum modo de manipular as exibições em `onCreate()`.
- `onCreateView`: Para criar uma exibição em um fragmento, sobrescreva o método `onCreateView()` e gere a exibição você mesmo, então, retorne-a no final da função. Veja a Listagem 9-1 para obter um exemplo. Algo importante a notar: Mesmo que as exibições sejam criadas, elas não estão totalmente construídas ainda. Se qualquer estado salvo precisar ser restaurado para a exibição (por exemplo, se a atividade foi destruída e recriada por causa de uma rotação da tela), esse estado não estará disponível até a próxima etapa.
- `OnActivityCreated`: `onActivityCreated()` é a etapa final, chamada antes de seu fragmento estar totalmente criado. Neste ponto, seu fragmento está totalmente configurado. Por isso, geralmente é melhor colocar grande parte do código que envolve as exibições ou o estado salvo em `onActivityCreated()`.

Criando o layout para adicionar/editar fragmentos

O layout para adicionar e editar é bem simples porque o formulário contém apenas alguns campos:

- **Title (Título):** O título da tarefa como será mostrado na exibição da lista
- **Body (Campo):** O corpo da tarefa, onde o usuário digitaria os detalhes
- **Reminder Date (Data do Lembrete):** A data na qual o usuário deve ser lembrado sobre a tarefa
- **Reminder Time (Hora do Lembrete):** A hora na qual o usuário deve ser lembrado na data do lembrete

Quando o aplicativo estiver completo e em execução em um dispositivo ou emulador, a tela ficará como a Figura 9-2.

Figura 9-2: A tela Add/Edit Task Reminder

Para criar este layout, crie um arquivo de layout no diretório `res/layout` com um nome apropriado; por exemplo, `reminder_edit.xml`. Para criar esse arquivo, siga estas etapas:

1. **Clique com o botão direito no diretório `res/layout` e escolha New (Novo)⇨Android XML File (Arquivo XML Android).**
2. **Forneça o nome no campo File (Arquivo).**
3. **Deixe o tipo padrão do recurso selecionado — Layout.**
4. **Deixe a pasta definida para `res/layout`.**
5. **Defina o elemento-raiz para ScrollView.**
6. **Clique no botão Finish (Terminar).**

Agora, você precisa fornecer todas as definições da exibição para construir a tela. (Consulte a Figura 9-2). Para tanto, digite o código mostrado na Listagem 9-6.

Listagem 9-6: O arquivo reminder_edit.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">                          →5
<LinearLayout                                                         →6
    android:orientation="vertical"                                     7
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="@string/title" />                         →12
    <EditText android:id="@+id/title"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content" />                 →15
    <TextView android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="@string/body" />                          →18
    <EditText android:id="@+id/body"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:minLines="5"
              android:scrollbars="vertical"
              android:gravity="top" />                                →24
    <TextView android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="@string/date" />                          →27
    <Button
              android:id="@+id/reminder_date"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"/>                   →31
    <TextView android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="@string/time" />                          →34
    <Button
              android:id="@+id/reminder_time"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content" />                  →38
    <Button   android:id="@+id/confirm"
              android:text="@string/confirm"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />                 →42
</LinearLayout>
</ScrollView>
```

Eis uma pequena explicação do código na Listagem 9-6:

→ **5** A view-mãe é `ScrollView`, que cria uma barra de rolagem e permite à exibição ser paginada quando o conteúdo dela é grande demais para caber na tela. A tela mostrada na Figura 9-2 é exibida no modo Retrato. Porém, se o dispositivo for girado em 90 graus, a exibição mudará e mais da metade será cortada. A `ScrollView` mãe permite que o conteúdo restante da tela seja paginável. Portanto, o usuário pode mover o dedo para cima na tela para paginar o conteúdo e ver o resto da exibição.

→ **6** Uma `ScrollView` pode ter apenas um filho — neste caso, o `LinearLayout` principal que mantém o resto do layout.

→ **7** A orientação do layout linear é definida para vertical para indicar que as exibições dentro desse layout devem ser empilhadas umas sobre as outras.

→ **12** É a etiqueta para o campo `Title`.

→ **15** O `EditText` que permite ao usuário fornecer um título para a tarefa. Você adiciona `<string name="title">Title</string>` a `strings.xml`.

→ **18** A etiqueta para o campo `Body` e adiciona `<string name="body"> Body</string>` a `strings.xml`.

→ **24** O `EditText` que define o campo `Body`. A exibição `EditText` definiu a propriedade `minLines` para 5 e a propriedade `gravity` para `top`, para informar à plataforma Android que `EditText` tem, pelo menos, cinco linhas de altura e que o texto digitado pelo usuário deverá ser vinculado ao topo da exibição (a gravidade).

→ **27** A etiqueta da data do lembrete também usa um recurso de string. Você precisa adicionar um recurso de string com o nome `"date"` e um valor `"Reminder Date"`.

→ **31** Quando o botão da data do lembrete é tocado, `DatePickerDialog` é inicializado. O usuário pode escolher uma data com um seletor de datas predefinido do Android. Quando a data é escolhida via `DatePicker`, o valor da data é colocado no texto do botão.

→ **34** Esta etiqueta da hora do lembrete usa um recurso de string. Você precisa adicionar um recurso de string com o nome `"time"` e um valor `"Time"`.

→ **38** Quando este botão do lembrete da hora é clicado, `TimePicker` é inicializado. O usuário pode escolher uma hora com um seletor de hora predefinido do Android. Quando a hora é escolhida via `TimePickerDialog`, o valor da hora é colocado no texto do botão.

→ **42** Este botão de confirmação salva os valores do formulário quando clicado. Adicione `<string name="confirm">Save</string>` a `strings.xml`.

Completando seu Fragmento de Lista

A classe `ListFragment` exibe uma lista de itens através da vinculação à uma fonte de dados, tal como, um array ou um cursor, e exibe os métodos de callback quando o usuário seleciona um item. Contudo, para construir uma relação de itens para exibir em uma lista, você precisa adicionar um layout que defina como será cada linha.

Um cursor fornece acesso aleatório de leitura e gravação ao conjunto de resultados retornado por uma consulta ao banco de dados.

Adicione um novo layout ao diretório `res/layout` com um elemento-raiz `TextView` e forneça-lhe um nome apropriado a um item do tipo linha; por exemplo, `reminder_row.xml`. Dentro dessa exibição, digite o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text1"                                          →4
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip"/>
```

Este código simplesmente define uma linha na qual os valores de texto podem ser colocados com um preenchimento de 10 pixels independentes da densidade. A linha 4 define o ID da exibição que você precisa ao carregar a lista com dados.

A exibição adicionada é fornecida pronta no sistema Android. Se você vir a documentação Android em `Android.R.Layout` em `simple_list_item_1` e examiná-la via repositório de código-fonte Android, poderá ver praticamente a mesma definição XML. O código-fonte pode ser encontrado em

```
https://github.com/android/platform_frameworks_base/blob/
          master/core/res/res/layout/simple_list_item_1.
          xml
```

`ListFragment` requer que um adaptador preencha o conteúdo da exibição de lista. Vários adaptadores estão disponíveis, mas como você não tem ainda uma base de dados (construída com um banco de dados SQLite no Capítulo 12), poderá criar temporariamente dados falsos para que possa ver a lista em ação. Na seguinte seção, você adicionará dados falsos para que possa definir o adaptador de `ListFragment` com uma chamada para `setListAdapter()`.

Ficando cheio com dados falsos

Adicione o seguinte campo e método à sua classe `ReminderListFragment`:

```
private ListAdapter mAdapter;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    String[] items = new String[] { "Foo", "Bar", "Fizz", "Bin" };      → 7

    mAdapter = new ArrayAdapter<String>(getActivity(),                  → 9
            R.layout.reminder_row, R.id.text1, items);
    setListAdapter(mAdapter);                                           → 11
}
```

Eis uma pequena explicação do código:

→ 7 Um array de itens de tipo string que serão exibidos na lista.

→ 9 A criação de um novo `ArrayAdapter` para tipos string. Um `ArrayAdapter` gerencia um `ListView` apoiado por um número arbitrário de objetos arbitrários – neste caso, um array de strings simples. Este código está usando recurso Java generics, que permite especificar o tipo de objeto com o qual `ArrayAdapter` trabalhará. O construtor do `ArrayAdapter` contém estes elementos:

- `getActivity()`: O contexto atual (como a atividade é uma implementação da classe `Context`, você pode usar a instância atual com o contexto).
- `R.layout.reminder_row`: O layout de linha que deve ser usado para cada linha em `ListView`.
- `R.id.text1`: O ID do `TextView` dentro de `R.layout.reminder_row`, no qual colocar os valores do array.
- `items`: O array de strings a carregar em ListView.

→ 11 A chamada para `setListAdapter()` que informa a `ListFragment` como preencher o `ListView`. Neste caso, você está usando `ArrayAdapter`, criado na linha 4, para carregar `ListView`.

Inicie o aplicativo Android escolhendo Run (Executar)⇨Run ou pressionando Ctrl+F11. A tela que verá deve parecer com a Figura 9-3.

O código e exemplo anteriores ilustram como usar uma fonte de dados estática para `ListFragment`. No Capítulo 12, você o substituirá pelo código que carregará os dados a partir de um banco de dados SQLite.

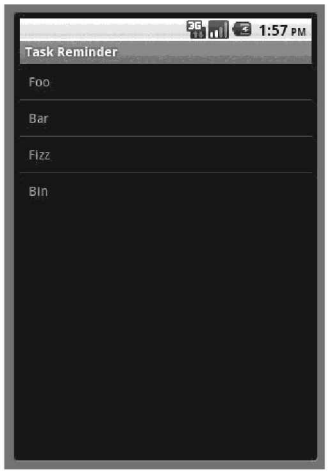


Figura 9-3: Task Reminder sendo executado com dados falsos.

Lidando com os eventos de clique do usuário

Os itens na lista expõem eventos de clique que permitem ao usuário interagir com cada item. Os objetos `View` do Android têm dois tipos principais de eventos de clique:

- **Clique:** O usuário toca rapidamente em uma exibição, tal como um botão.
- **Clique longo:** O usuário toca em um botão e segura-o por um momento.

Tanto exibições quanto atividades podem interceptar esses eventos via diversos métodos. Nas seguintes seções, você responderá a cada tipo de evento em um `ListFragment`. No Capítulo 11, irá configurar o aplicativo para responder aos eventos de clique de `Button`.

Cliques curtos

O `ListFragment` no Android faz muito tratamento de evento pesado para você (o que é bom, pois a programação não deve ser complicada).

Depois do método `onCreate()` em `ReminderListFragment`, digite este método:

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
}
```

Este código sobrescreve a implementação padrão `onListItemClick()` que é fornecida por `ListFragment`. Quando um item da lista é clicado, esse método é chamado e os seguintes parâmetros são transmitidos na chamada:

- `l`: O `ListView` onde o clique ocorreu
- `v`: O item que foi clicado com `ListView`
- `position`: A posição do item clicado na lista
- `id`: O ID da linha do item que foi clicado

Usando estas variáveis, você pode determinar qual item foi clicado e, então, executar uma ação com base nessas informações. Quando um item é clicado nesta lista, uma intenção abre `ReminderEditActivity` para permitir que o usuário edite o item, como mostrado na seção "Iniciando novas atividades com intenções" posteriormente neste capítulo.

Cliques longos

Um *clique longo* (ou *pressionar longo*) ocorre sempre que um usuário pressiona uma exibição por um tempo prolongado. Para lidar com o evento de clique longo do item da lista em `ListFragment`, adicione a seguinte linha de código ao final do método `onViewCreated()` em `ReminderListFragment`:

```
registerForContextMenu(getListView());
```

O método externo, `registerForContextMenu()`, é responsável por registrar um menu contextual para ser mostrado por determinada exibição. Diversas exibições podem mostrar um menu contextual; não é limitado a uma única exibição. Cada item da lista pode, portanto, criar um menu contextual. `RegisterForContextMenu()` aceita um objeto `View` como um parâmetro que `ListFragment` deve registrar como elegível para a criação do menu contextual. O método interno, `getListView()`, retorna um objeto `ListView` que é usado para o registro. A chamada, `getListView()`, é um membro da classe `ListFragment`.

Agora que você registrou `ListView` como elegível para criar um menu contextual, precisa responder ao evento de clique longo em qualquer item dado. Quando um item é clicado por algum tempo em `ListView`, `registerForContextMenu()` reconhece-o e chama o método `onCreateContextMenu()` quando o menu contextual está pronto para ser criado. Nesse método, você configura seu menu contextual.

No final do arquivo da classe `ReminderListFragment`, digite o seguinte método:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo
               menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
}
```

Este método é chamado com os seguintes parâmetros:

- `menu`: O menu contextual sendo construído.
- `v`: A exibição para a qual o contexto está sendo construído (a exibição que o usuário clicou por um tempo).
- `menuInfo`: As informações extras sobre o item para o qual o menu contextual deve ser mostrado (pode variar dependendo do tipo de exibição no parâmetro `v`).

Dentro deste método, você pode modificar o menu apresentado ao usuário. Por exemplo, quando um usuário pressiona por um tempo um item na lista de tarefas, ele deve ter a permissão de apagá-lo. Portanto, apresente a ele a opção Delete (Apagar) em um menu contextual (como descrito no Capítulo 10).

Identificando Sua Intenção

Até mesmo nos aplicativos que têm apenas duas telas (tal como o aplicativo Task Reminder) há muita coisa acontecendo internamente. Uma interação notável que ocorre entre o aplicativo e o usuário é a introdução de novas telas quando o usuário experimenta vários recursos do aplicativo. Como em qualquer aplicativo rico em recursos, o usuário pode interagir com cada tela independentemente. A grande pergunta que surge é: "Como um usuário abre outra tela?"

A interação com as telas é lidada via sistema de intenções do Android. Nas seguintes seções, você irá configurar as intenções que permitem ao usuário navegar de uma tela para a outra. Felizmente, é um processo simples (volte ao Capítulo 7 para descobrir mais sobre o sistema de intenções).

Iniciando novas atividades com intenções

As atividades são iniciadas via estrutura de intenções do Android. `Intent` é uma classe que representa uma mensagem colocada no sistema de intenções do Android (parecido com uma arquitetura de barramento de mensagem) e qualquer método que possa responder à intenção permite que a plataforma Android saiba, resultando em uma atividade iniciando ou uma lista de aplicativos a escolher (este conceito de seletor é explicado posteriormente na seção "Criando um seletor"). Você pode considerar uma intenção como uma descrição abstrata de uma operação.

Iniciar certa atividade é fácil. Em seu `ReminderListFragment`, digite o seguinte código no método `onListItemClick()`:

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    Intent i = new Intent(getActivity(), ReminderEditActivity.class);      →4
    i.putExtra(ReminderProvider.COLUMN_ROWID, id);                         →5
    startActivity(i);                                                      →6
}
```

Eis uma pequena explicação de cada linha:

→ 4 Esta linha cria uma nova intenção usando o construtor `Intent` que aceita o contexto atual, que é a atividade em execução no momento, assim como uma classe que o sistema de intenções deve tentar iniciar — a atividade `ReminderEdit`.

→ 5 Esta linha coloca dados extras no objeto `Intent`. Esta intenção inclui um par de chave/valor. A chave é `RowId` e o valor é o ID da exibição que foi clicada. Esse valor é colocado na intenção para que a atividade receptora (`ReminderEditActivity`) possa obter esses dados no objeto `Intent` e usá-los para carregar as informações sobre a intenção. No Capítulo 12, você poderá ver os dados fluindo para `ReminderEditFragment`.

→ 6 Esta linha inicia a atividade de dentro da atividade atual. Esta chamada coloca a mensagem de intenção no sistema de intenções do Android e permite que o Android decida como abrir essa tela para o usuário.

Criando um seletor

Em algum ponto, você poderá encontrar uma determinada situação na qual precisa fornecer ao usuário uma lista de aplicativos que podem lidar com determinada intenção. Um exemplo comum é compartilhar dados com um amigo através de ferramentas de rede, tais como e-mail, SMS, Twitter, Facebook ou Google Latitude.

O sistema de intenções do Android foi construído para lidar com esse tipo de situação. Embora não seja usado no aplicativo Task Reminder, pode ser útil. O código para exibir as várias opções disponíveis para o usuário é mostrado na Listagem 9-7.

Listagem 9-7: Criando um seletor de intenções

```
Intent i = new Intent(Intent.ACTION_SEND);                                  →1
i.setType("text/plain");                                                    →2
i.putExtra(Intent.EXTRA_TEXT, "Hey Everybody!");                            →3
i.putExtra(Intent.EXTRA_SUBJECT, "My Subject");                             →4
Intent chooser = Intent.createChooser(i, "Who Should Handle this?");        →5
startActivity(chooser);                                                     →6
```

Eis uma pequena explicação de cada linha na Listagem 9-7:

→ **1** A criação de uma nova intenção que informa ao sistema de intenções que o usuário deseja enviar ou mandar algo por e-mail.

→ **2** O tipo de conteúdo da mensagem. Pode ser definido para qualquer tipo MIME explícito. Os tipos MIME levam em conta as letras maiúsculas e minúsculas, diferentemente dos tipos MIME RFC, portanto, digite-os com letras minúsculas para especificar o tipo da intenção. Apenas os aplicativos que podem responder a esse tipo de intenção aparecerão no seletor.

→ **3** Colocando dados extras na intenção. É o corpo da mensagem que o aplicativo usará. Se um cliente de e-mail for escolhido, essa linha irá compor o corpo do e-mail. Se o Twitter for escolhido, a mensagem do tweet será o corpo. Todo aplicativo que responde à intenção pode lidar com os dados extras de modo particular. Não espere que os dados sejam tratados como você acha que deveriam ser no aplicativo de destino. O desenvolvedor desse tipo de aplicativo determina como o aplicativo deve trabalhar com os dados extras.

→ **4** Parecida com a linha 3, mas com um assunto extra fornecido. Se um cliente de e-mail responder, essa linha normalmente irá compor o assunto do e-mail.

→ **5** Cria o seletor (o objeto `Intent` tem um método auxiliar estático que o ajuda). O seletor é, em si, uma intenção. Você simplesmente fornece a intenção de destino (a ação que deseja que aconteça), assim como um título para o seletor instantâneo que é exibido.

→ **6** Inicia a intenção. Cria um seletor a partir do qual você escolhe um aplicativo.

O seletor criado na Listagem 9-7 é mostrado na Figura 9-4.

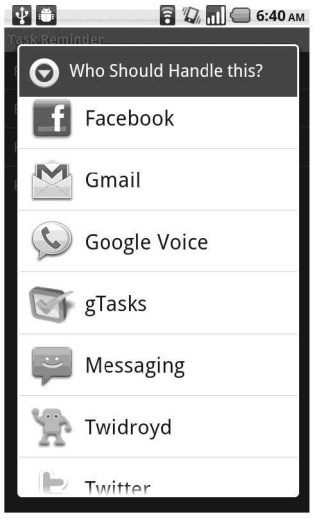


Figura 9-4: O novo seletor que foi criado.

Se o sistema de intenções não conseguir encontrar um aplicativo válido para lidar com a intenção, o seletor será criado com uma mensagem informando ao usuário que nenhum aplicativo pode realizar a ação, como mostrado na Figura 9-5.

Figura 9-5: O seletor informa ao usuário que o Android não pode encontrar um aplicativo correspondente para lidar com a intenção.

O seletor é um modo útil de aumentar a interoperabilidade de um aplicativo. Contudo, se você simplesmente chamar `startActivity()` com sua intenção sem criar um seletor, seu aplicativo poderá paralisar porque o Android está dando a você o total domínio e assume que você sabe o que está fazendo. Não incluindo um seletor, você está assumindo que o dispositivo de destino tem, pelo menos, um aplicativo para lidar com a intenção. Se não tiver, o Android irá gerar uma exceção (visível via DDMS) para informar que nenhuma classe pode lidar com a intenção. Para o usuário, seu aplicativo paralisou.

Para fornecer uma experiência satisfatória ao usuário, sempre forneça um seletor de intenção ao inicializar as intenções que são direcionadas para a interoperabilidade com os outros aplicativos. O seletor fornece um modelo de utilização suave e consistente que o resto do Android já suporta.

Capítulo 10

À la Carte com Seu Menu

Neste Capítulo

Construindo um menu de opções

Construindo um menu contextual

Todo aplicativo Android bom inclui menus. Se você tem um dispositivo Android e fez o download de alguns aplicativos na Google Play Store, provavelmente encontrou algumas implementações de menu boas e ruins.

Um menu ruim fornece muito pouco texto útil (se algum) na descrição do menu e não fornece nenhum ícone. Alguns erros de menu comuns incluem

- Um título de menu ruim
- Um menu sem ícone
- Nenhum menu

Um menu bom deve ter um apelo visual, assim como textual, para o usuário final. A aparência de um ícone de menu mostra que o desenvolvedor realmente considerou o processo de criar o menu e decidiu qual ícone é mais adequado para o aplicativo.

As atividades e os fragmentos podem ter menus e, neste caso, eles serão combinados em um. Neste capítulo, você adicionará menus de opção e contextuais aos fragmentos no aplicativo Task Reminder, mas poderia facilmente incluí-los em uma atividade também.

Compreendendo os Menus de Opções e Contextuais

O Android fornece um mecanismo simples para você adicionar menus aos seus aplicativos. Você encontrará os seguintes tipos de menus:

- **Menu de opções e menu da barra de ação:** Muito provavelmente, é o tipo de menu mais comum com o qual você irá trabalhar. É o menu primário para uma atividade ou fragmento.

  No Android 3.x e posterior, o menu Options (Opções) está na barra de ação, no topo da tela (leia mais sobre a barra de ação no Capítulo 1). No Android 2.x e anterior, o menu Options é apresentado ao usuário com o pressionar da tecla Menu no dispositivo. A Figura 10-1 mostra os mesmos menus nos dois dispositivos.

  No menu de opções, existem dois grupos:

  - *Ícone:* Estas opções de menu estão disponíveis na parte inferior da tela. O dispositivo suporta até seis itens de menu e são os únicos itens que suportam o uso de ícones. Eles não suportam caixas de seleção ou botões de rádio.
  - *Expandido:* O menu expandido é uma lista de itens de menu que vai além dos seis itens no menu Icon (Ícone). Esse menu é apresentado pelo ícone de menu More (Mais) que é colocado automaticamente na tela quando o desenvolvedor do aplicativo tem mais itens do que caberiam no menu Icon.

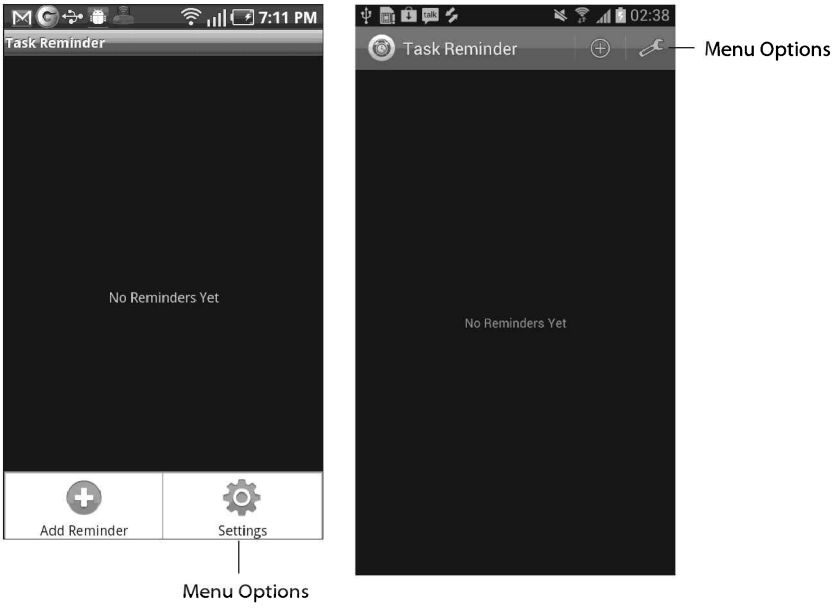


Figura 10-1: O menu de opções no dispositivo Android 2.x (esquerda) e em um dispositivo Android 4.x (direita).

- **Menu contextual:** Uma lista de itens de menu flutuante apresentada quando um usuário pressiona uma exibição por algum tempo.
- **Submenu:** Uma lista de itens de menu flutuante que o usuário abre tocando em um item no menu Options (Opções) ou em um menu contextual. Um item de submenu não pode suportar submenus aninhados.

Criando Seu Primeiro Menu

Você pode criar um menu através do código ou de um arquivo XML que é fornecido no diretório `res/menu`. O método preferido para criar um menu é defini-lo através do XML, então, inseri-lo em um objeto programável com o qual você pode interagir. Isto ajuda a separar a definição do menu do código do aplicativo.

Definindo o arquivo XML

Para definir um menu XML, siga estas etapas:

1. **Crie uma pasta menu no diretório** `res`**.**
2. **Adicione um arquivo com o nome** `list_menu.xml` **ao diretório de menus.**
3. **Digite o seguinte código no arquivo** `list_menu.xml`**.**

```
<?xml version="1.0" encoding="utf-8"?>
<menu
  xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/menu_insert"
          android:icon="@android:drawable/ic_menu_add"
          android:title="@string/menu_insert" />
</menu>
```

Note que um novo recurso de string é incluído (mostrado em negrito). Você criará isso na Etapa 4. O valor `android:icon` é um ícone Android predefinido. As versões ldpi, mdpi, hdpi e xhdi desse ícone são predefinidas na plataforma Android, portanto, você não tem que fornecer este mapa de bits em seus recursos de desenho. Para exibir os outros recursos disponíveis, confira a documentação `android.R.`drawable em (conteúdo em inglês)

```
http://developer.android.com/reference/android/R.
        drawable.html
```

Todos os recursos na classe `android.R` fornecem ao seu aplicativo uma interface de usuário e experiência de uso padronizadas com a plataforma Android.

4. **Crie um novo recurso de string com o nome** `menu_insert` **e com o valor** `Add Reminder` **no arquivo de recursos** `strings.xml`**.**

5. **Abra a classe** `ReminderListFragment` **e certifique-se de que as linhas em negrito estejam em seu** `onViewCreated()`**:**

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    setEmptyText(getResources().getString(string.no_reminders));
    registerForContextMenu(getListView());
    setHasOptionsMenu(true);
}
```

`registerForContextMenu()` informa ao Android que `ListView` deseja contribuir com o menu contextual (aquele que aparece quando um usuário pressiona a exibição por algum tempo). `setHasOptionsMenu()` informa ao Android 2.x para mostrar o menu quando o usuário pressiona o botão Menu e ao Android 3.x ou posterior para mostrar o menu na barra de ação. Volte ao Capítulo 9 para obter mais informações sobre `registerForContextMenu()` e `setHasOptionsMenu()`.

6. **Adicione o método** `onCreateOptionsMenu()` **à sua classe:**

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu,inflater);
    inflater.inflate(R.menu.list_menu, menu);
}
```

`MenuInflater` insere o layout do menu XML criado anteriormente e adiciona-o ao menu que foi transmitido como um argumento na chamada do método.

7. **Instale o aplicativo no emulador e clique no botão Menu.**

A Figura 10-2 mostra o ícone de menu Add Reminder (Adicionar Lembrete) que você acabou de criar.

Figura 10-2: O ícone de menu Add Reminder.

Lidando com as ações do usuário

Depois de você ter criado o menu, terá então que adicionar o que acontece quando um usuário clica nele. Para tanto, digite o seguinte código no final do arquivo da classe:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {                    →2
    switch (item.getItemId()) {                                          →3
    case R.id.menu_insert:                                               →4
        editReminder(0);                                                 →5
        return true;                                                     →6
    }

    return super.onOptionsItemSelected(item);                            →9
}
```

As linhas do código são explicadas em detalhes aqui:

→ 2 Este é o método chamado quando um item de menu é selecionado. O parâmetro `item` identifica em qual item o usuário tocou.

→ 3 Para determinar com qual item você está trabalhando, compare o ID dos itens de menu com os itens de menu conhecidos que você tem. Portanto, uma instrução `switch` é usada para verificar cada possível caso válido. Você obtém o ID do menu através do método `getItemId()` de `MenuItem`.

→ 4 O ID do item de menu Add Reminder (Adicionar Lembrete) verifica se o usuário selecionou esse item.

→ 5 Se o usuário selecionou o item de menu Add Reminder, o aplicativo é instruído a criar um lembrete através do método `editReminder()` (definido na próxima seção). Por convenção, chamar `editReminder()` com um ID `0` significa que o aplicativo deve criar um novo lembrete.

→ 6 Esta linha retorna true para informar ao método `onMenuItemSelected()` que uma seleção de menu foi tratada.

→ 9 Se a seleção do menu e o retorno não foram tratados anteriormente, a classe-mãe tentará tratar o item de menu.

Você poderá receber erros de compilação neste momento, mas não se preocupe! Você terminará o aplicativo na próxima seção.

Criando uma tarefa de lembrete

O método `editReminder()` permite ao usuário navegar para `ReminderEditActivity` para editar ou criar uma nova tarefa com um lembrete. Digite o seguinte método na parte inferior de seu arquivo de classe `ReminderListFragment`:

```
public void editReminder(long id) {
    Intent i = new Intent(getActivity(), ReminderEditActivity.class);
    i.putExtra(ReminderProvider.COLUMN_ROWID, id);
    startActivity(i);
}
```

Este código cria uma nova intenção que inicia `ReminderEditActivity`, e então chama `startActivity()` para iniciar a atividade.

Criando um Menu Contextual

Um menu contextual aparece quando um usuário pressiona uma exibição por um certo tempo. O menu contextual é um menu flutuante que fica sobre a tela atual e permite aos usuários escolherem uma entre várias opções relacionadas à exibição pressionada.

Felizmente, criar um menu contextual é bem parecido com criar um menu de opções. Você pode definir o menu no XML e inseri-lo usando o mesmo mecanismo utilizado na criação do menu de opções. Tudo que você precisa fazer é chamar `registerForContextMenu()` com uma exibição como destino (veja o Capítulo 9 para descobrir como criar uma exibição como destino). Depois de criar isso, você precisa sobrescrever a chamada `onCreateContextMenu()` — também demonstrada no Capítulo 9.

O aplicativo Task Reminder precisa de um mecanismo no qual apagar uma tarefa quando ela não é mais necessária. Os usuários podem pressionar uma tarefa da lista e um menu contextual aparecerá, permitindo que eles apaguem a tarefa, selecionando um item no menu.

Criando o arquivo XML do menu

Para criar este menu, crie um novo arquivo XML no diretório `res/menu`. Nomeie-o como `list_menu_item_longpress.xml`. Digite o seguinte código no arquivo XML:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_delete"
          android:title="@string/menu_delete" />
</menu>
```

Note que a propriedade `title` usa um novo recurso de string `menu_delete`. Você precisa criar um novo recurso de string com o nome `menu_delete` e o valor `Delete Reminder`. Note também que você não precisa de um ícone associado a esse menu.

Um menu contextual não suporta ícones; ele é simplesmente uma lista de opções de menu que flutua acima da atividade atual.

Carregando o menu

Para carregar o XML do menu e exibi-lo ao usuário, digite o seguinte código no método `onCreateContextMenu()`:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    MenuInflater mi = getActivity().getMenuInflater();
    mi.inflate(R.menu.list_menu_item_longpress, menu);
}
```

Este código executa a mesma função da chamada `onCreateOptionsMenu()`, mas, desta vez, você está carregando e inserindo o conteúdo do menu contextual. Agora, se um usuário pressionar um item da lista por algum tempo, verá um menu contextual, como mostrado na Figura 10-3.

Figura 10-3: O menu contextual no aplicativo Task Reminder.

Lidando com as seleções do usuário

Lidar com a seleção dos itens do menu contextual é muito parecido com lidar com um menu de opções. Digite o seguinte código na parte inferior de seu arquivo de classe:

```
@Override
public boolean onContextItemSelected(MenuItem item) {                    →2
    switch(item.getItemId()) {                                           →3
        case R.id.menu_delete:                                           →4
            // Apaga a tarefa
            return true;
    }
    return super.onContextItemSelected(item);
}
```

As linhas do código são explicadas aqui:

→ **2** Este é o método chamado quando um item do menu contextual é selecionado. O parâmetro `item` é o item que foi selecionado no menu contextual.

→ **3** Uma instrução switch determina qual item foi selecionado, com base no ID definido no arquivo `list_menu_item_longpress.xml`.

→ **4** Este é o ID para o botão `menu_delete` no arquivo `list_menu_item_longpress.xml`. Se esta opção de menu for selecionada, o código seguinte executará uma ação com base nesse estímulo. Nada está acontecendo no bloco de código neste capítulo, mas isso mudará no Capítulo 12, onde você apagará a tarefa do banco de dados SQLite.

Você pode adicionar muitos itens de menu contextual diferentes ao arquivo `list_menu_item_longpress.xml` e alternar entre eles na chamada do método `onContextMenuItemSelected()` — cada um realizando uma ação diferente.

Capítulo 11

Lidando com a Entrada do Usuário

Neste Capítulo

- Trabalhando com as exibições `EditText`
- Criando seletores de data e seletores de hora
- Configurando caixas de diálogo de alerta
- Validando a entrada do usuário

Raramente um aplicativo não permite que os usuários interajam com ele. Quer usem caixa de texto, seletor de data ou hora, botão de rádio, caixa de seleção ou qualquer outro mecanismo de entrada, os usuários precisam interagir com seu aplicativo de algum modo. De modo geral, a entrada de dados também é realizada através de botões, arrastar telas, menus, pressionar por um tempo e várias outras opções. Este capítulo foca unicamente na entrada do usuário na forma de alertas, texto com forma livre, datas e horas.

Criando a Interface de Entrada do Usuário

O tipo de entrada mais comum é a exibição `EditText`, usada para a entrada de texto com forma livre. Usando uma exibição `EditText`, você pode fornecer um teclado na tela ou permitir que o usuário use o teclado físico (se o dispositivo fornecer um) para inserir a entrada.

Caso você esteja familiarizado com outras plataformas de programação, uma caixa de texto realiza a mesma função de uma exibição `EditText`.

Criando uma exibição EditText

No Capítulo 9, você criou um arquivo XML de layout da exibição, chamado `reminder_edit.xml`, que continha estas linhas de código:

```
<EditText android:id="@+id/title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

O fragmento cria um mecanismo de entrada na tela, onde o usuário pode digitar o título da tarefa. A exibição `EditText` se expande na largura da tela e ocupa apenas a altura necessária. Quando a exibição é selecionada, o Android abre automaticamente o teclado na tela para permitir a entrada do usuário.

O exemplo anterior permite adotar uma abordagem minimalista, em comparação com o seguinte exemplo `EditText`, que também está presente no arquivo de layout `reminder_edit.xml`:

```
<EditText android:id="@+id/body" android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:minLines="5"
    android:scrollbars="vertical"
    android:gravity="top" />
```

Este código cria o texto de descrição do corpo da tarefa. A largura e a altura do layout são iguais à exibição `EditText` no exemplo anterior e a exibição `EditText` expande-se na largura da tela. Estas três propriedades descrevem as diferenças nesta definição `EditText`:

- `minLines`: Especifica a altura da exibição `EditText`. Como a exibição `EditText` é uma subclasse do objeto `TextView`, elas compartilham esta propriedade. Este código especifica um mínimo de cinco linhas para o objeto `EditText` na tela para que a exibição seja um mecanismo de entrada de texto para as mensagens longas.

  Compare esta exibição com a parte do corpo de qualquer cliente de e-mail e poderá ver que são muito parecidos — o corpo é muito maior que o assunto. Neste caso, o corpo é muito maior que o título.

- `scrollbars`: Define quais barras de rolagem devem estar presentes quando o texto excede a área de entrada disponível; especifica barras de rolagem verticais ao lado da exibição `EditText`.

- `gravity`: Alinha o texto (por padrão) no meio da exibição quando o usuário coloca o foco em um campo `EditText` (como mostrado à esquerda na Figura 11-1), embora não seja o que os usuários esperariam ao trabalhar com um mecanismo de entrada com diversas linhas. Para posicionar o cursor no topo da exibição `EditText`, como seria o esperado, você deve definir a gravidade da exibição `EditText` para `top`, para forçar o texto a ficar no topo da entrada `EditText`, como mostrado à direita na Figura 11-1.

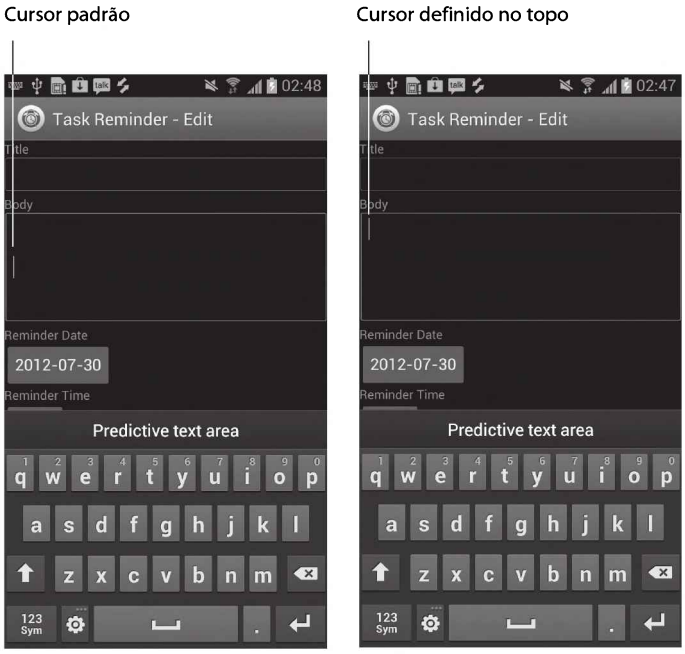


Figura 11-1: Uma exibição `EditText`, com o cursor colocado no centro (imagem à esquerda) e no topo (imagem à direita).

Exibindo um teclado na tela

A exibição `EditText` é responsável pela exibição do teclado na tela. Como alguns dispositivos não têm um teclado físico, deve haver um teclado na tela para interação com os mecanismos de entrada. Uma propriedade que a exibição `EditText` fornece é um modo de manipular o aspecto visual do teclado na tela.

Você ajusta o teclado na tela porque diferentes tipos de entrada `EditText` podem precisar de teclas diferentes. Por exemplo, se `EditText` for um número de telefone, o teclado na tela deverá exibir apenas números. Se o valor `EditText` for um endereço de e-mail, por exemplo, o teclado na tela deverá exibir os atributos comuns do estilo de e-mail — tal como o símbolo @.

Configurar devidamente o teclado na tela pode aumentar a usabilidade de seu aplicativo.

Você pode configurar o modo como o teclado aparece na tela usando a propriedade `inputType` na exibição `EditText`. Por exemplo, se você definir `android:inputType="number"` no corpo `EditText`, o teclado exibirá teclas numéricas, ao invés de teclas com letras, como mostrado na Figura 11-2.

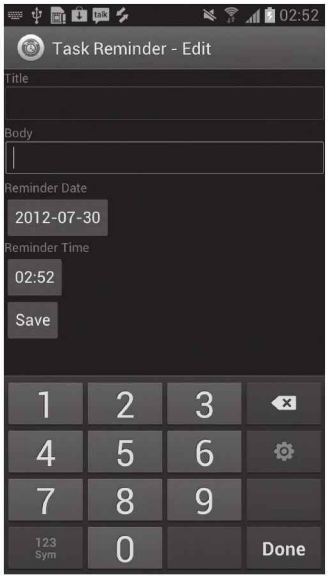


Figura 11-2: O teclado personalizado para a entrada numérica.

A propriedade `inputType` tem opções demais para o propósito deste livro, mas você pode examinar a lista completa em `http://developer.android.com/reference/android/widget/TextView.html#attr_android:inputType` (conteúdo em inglês).

Sendo Exigente com Datas e Horas

Um aplicativo Task Reminder sem um modo de definir datas e horas é um aplicativo ruim — ele seria apenas um aplicativo simples de lista de tarefas.

Se você programou datas e horas em outra linguagem de programação, percebeu que construir um mecanismo para o usuário inserir a data e a hora pode ser um processo complicado. A plataforma Android vem ao seu resgate fornecendo duas classes para ajudá-lo: `DatePicker` e `TimePicker`. Esses seletores também fornecem classes predefinidas para abrir uma caixa de diálogo, onde o usuário seleciona uma data e uma hora. Portanto, você pode incorporar `DatePicker` ou `TimePicker` nas exibições de seu aplicativo ou usar as classes `DialogFragment`.

Criando botões para selecionar

O arquivo `reminder_edit.xml` contém mecanismos para ajudar a mostrar `DatePicker` e `TimePicker` (sob as definições `EditText` descritas anteriormente). Esses dois botões têm etiquetas acima deles, como mostrado na Listagem 11-1.

Listagem 11-1: Botões de data e hora com suas etiquetas TextView correspondentes

```
<TextView android:layout_width="wrap_content"                    → 1
    android:layout_height="wrap_content"
    android:text="@string/date" />
<Button                                                          → 4
    android:id="@+id/reminder_date"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    />
<TextView android:layout_width="wrap_content"                    → 9
    android:layout_height="wrap_content"
    android:text="@string/time" />
<Button                                                          → 12
    android:id="@+id/reminder_time"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    />
```

As linhas de código são explicadas nesta lista:

→ **1** A etiqueta `TextView` para o botão `Date` (Data); exibe o valor "`Reminder Date`" de acordo com o recurso de string.

→ **4** Define o botão que o usuário clica para abrir `DatePickerDialogFragment` (como explicado na próxima seção)

→ **9** A etiqueta `TextView` do botão `Time` (Hora); exibe o valor "`Reminder Time`" de acordo com o recurso de string.

→ **12** Define o botão que o usuário clica para abrir `TimePickerDialogFragment` (explicado na seção "Criando o seletor de data").

Criando o seletor de data

Um usuário que clica o botão Date (Data) deve ser capaz de alterar a data, como descrito nas várias seções a seguir.

Configurando o atendente de clique do botão Date

Para configurar o atendente de clique do botão Date, abra a atividade onde seu código será colocado. Para o aplicativo Task Reminder, abra o arquivo `ReminderEditFragment.java`.

Adicione as linhas de código mostradas em negrito na Listagem 11-2 ao método `onCreateView()`.

Listagem 11-2: Implementando o atendente de clique do botão Date

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {

    View v = inflater.inflate(R.layout.reminder_edit, container, false);

    mTitleText = (EditText) v.findViewById(R.id.title);
    mBodyText = (EditText) v.findViewById(R.id.body);
    mDateButton = (Button) v.findViewById(R.id.reminder_date);
    mTimeButton = (Button) v.findViewById(R.id.reminder_time);
    mConfirmButton = (Button) v.findViewById(R.id.confirm);

    mDateButton.setOnClickListener(new View.OnClickListener() {        → 13
        @Override
        public void onClick(View v) {                                   → 15
            showDatePicker();                                           → 16
        }
    });

    return v;
}
```

As linhas numeradas são descritas nesta lista:

→ **13** Define `onClickListener()` para `mDateButton.onClickListener()` é executado quando o botão é clicado. A ação que ocorre no clique do botão é mostrada na linha 16.

→ **15** Sobrescreve o comportamento de clique padrão do botão para que você possa fornecer seu próprio conjunto de ações a executar. O parâmetro `v` de `View` é a exibição que foi clicada.

→ **16** Define o que você deseja que aconteça quando o botão é clicado; chama um método no fragmento `showDatePicker()`, como explicado posteriormente na seção "Criando o método `showDatePicker()`".

Criando DatePickerDialogFragment

O sistema operacional Android vem com um `DatePickerDialog` predefinido que permite aos usuários selecionarem (ao invés de digitarem) uma data. Não vem organizado em um fragmento, portanto, você tem que fazer isto sozinho.

Os antigos objetos `dialog` foram projetados para serem chamados a partir de atividades, não de fragmentos. Ao abrir uma caixa de diálogo a partir de um fragmento, você terá que usar uma subclasse da classe `DialogFragment` se quiser que a caixa de diálogo se comporte devidamente.

Crie um novo arquivo, chame-o de `DatePickerDialogFragment` e adicione o seguinte código:

```
package com.dummies.android.taskreminder;

import android.app.DatePickerDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;                          → 6

public class DatePickerDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {          → 10
        Bundle args = getArguments();                                  → 11
        Fragment editFragment = getFragmentManager()                   → 12
                .findFragmentByTag(
                        ReminderEditFragment.DEFAULT_EDIT_FRAGMENT_TAG);
        OnDateSetListener listener = (OnDateSetListener) editFragment;  → 15

        return new DatePickerDialog(getActivity(), listener,           → 17
                args.getInt(ReminderEditFragment.YEAR),
                args.getInt(ReminderEditFragment.MONTH),
                args.getInt(ReminderEditFragment.DAY));
    }
}
```

Eis como o código funciona:

→ 6 Use as importações `android.support.v4.app.*`, e não seus equivalentes de `android.app`.

→ 10 O método que é chamado quando o Android deseja exibir a caixa de diálogo `DatePicker`. Esse método cria e retorna a caixa de diálogo.

Algumas vezes, você pode precisar usar `savedInstanceState` para restaurar o estado das instâncias anteriores. Contudo, neste caso, a caixa de diálogo já faz isso por você, portanto, você pode ignorar `savedInstanceState` com segurança neste método.

→ 11 Nos fragmentos, os argumentos do tipo construtor são passados via `setArguments()`, e não via construtor do fragmento, portanto, esta linha recupera os argumentos usando `getArguments()`, que você precisará na linha 17.

→ 12 Pede a `FragmentManager` para encontrar o fragmento chamado `DEFAULT_EDIT_FRAGMENT_TAG`, que é o `ReminderEditFragment`.

→ 15 Faz a conversão de `editFragment` em um `OnDateSetListener`. A caixa de diálogo criada por `onCreateDialog` precisa do objeto `onDateSetListener` para informar a `ReminderEditFragment` quando o usuário selecionar uma data.

→ **17** Esta linha chama o construtor `DatePickerDialog` e transmite `getActivity()`, o atendente contextual atual obtido na linha 12 e `args.getInt(ReminderEditFragment.YEAR)`, `args.getInt(ReminderEditFragment.MONTH)`, `args.getInt(ReminderEditFragment.DAY)`, o ano, mês e dia, como especificado nos argumentos para este fragmento.

Criando o método showDatePicker( )

Depois que você tiver uma classe `DatePickerDialogFragment`, poderá criar uma instância dela e mostrá-la para o usuário. O `onClickListener` do botão de data chamou `showDatePicker`, para que você possa implementar `showDatePicker()` agora. Adicione o seguinte código à classe `ReminderEditFragment` depois de `onCreateView()`:

```
//
// Constantes da caixa de diálogo
//
static final String YEAR = "year";
static final String MONTH = "month";
static final String DAY = "day";
static final String HOUR = "hour";
static final String MINS = "mins";
static final String CALENDAR = "calendar";
private void showDatePicker() {
    FragmentTransaction ft = getFragmentManager().beginTransaction();  →11
    DialogFragment newFragment = new DatePickerDialogFragment();       →12
    Bundle args = new Bundle();                                        →13
    args.putInt(YEAR, mCalendar.get(Calendar.YEAR));                   →14
    args.putInt(MONTH, mCalendar.get(Calendar.MONTH));
    args.putInt(DAY, mCalendar.get(Calendar.DAY_OF_MONTH));
    newFragment.setArguments(args);                                    →17
    newFragment.show(ft, "datePicker");                                →18
}
```

Eis como o código funciona:

→ **11** Implementa uma transação de fragmento para o novo fragmento.

→ **12** Cria uma instância `DatePickerDialogFragment`.

→ **13** Cria os argumentos do construtor de fragmentos em um objeto Bundle.

→ **14** Define o ano, mês e dia do fragmento da caixa de diálogo para o ano, mês e dia definidos no objeto `mCalendar`.

→ **17** Define os argumentos do fragmento para os valores no objeto Bundle criado na linha 4.

→ **18** Mostra o fragmento da caixa de diálogo `DatePicker`, que permite que a caixa de diálogo seja aberta na tela. Como `show()` chama `commit()`, você não precisa chamá-lo explicitamente.

Criando o seletor de hora

`TimePickerDialogFragment` permite aos usuários a seleção da hora de uma tarefa pendente a ser lembrada.

Configurando o listener de clique do botão Time

Configurar um `TimePickerDialogFragment` é quase idêntico a configurar `DatePickerDialogFragment`. Primeiro, você declara `onClickListener()` para o botão Time (Hora). Em `ReminderEditFragment.onCreateView()`, adicione o seguinte fragmento de código logo antes do retorno no final:

```
mTimeButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
       showTimePicker();

    }
});
```

Este método é igual a `onClickListener()` do botão Date (Data), exceto que você está chamando `showTimePicker()`, ao invés de `showDatePicker()`.

Criando o método showTimePicker( )

Para ajudar a criar o método `showTimePicker()`, a definição completa do método, com código, é mostrada na Listagem 11-3.

Listagem 11-3: O método showTimePicker( )

```
private void showTimePicker() {
    FragmentTransaction ft = getFragmentManager().beginTransaction();      →3
    DialogFragment newFragment = new TimePickerDialogFragment();
    Bundle args = new Bundle();
    args.putInt(HOUR, mCalendar.get(Calendar.HOUR_OF_DAY);                 →5
    args.putInt(MINS, mCalendar.get(Calendar.MINUTE));
    newFragment.setArguments(args);
    newFragment.show(ft, "timePicker");                                    →8
}
```

O código na Listagem 11-3 é bem simples porque é quase idêntico ao código do método `showDatePicker()`. Porém, você pode ver diferenças nestas linhas:

→ **3** Cria uma nova instância de `TimePickerDialogFragment`.

→ **5** Define os argumentos de `TimePickerDialogFragment` para serem os componentes de hora e minuto do calendário.

→ **8** Mostra o fragmento usando a tag "`timePicker`", que não é visível para o usuário.

Criando TimePickerDialogFragment

O sistema operacional Android vem com um `TimePickerDialog` predefinido que permite aos usuários selecionarem (ao invés de digitarem) uma hora. Como `DatePickerDialog`, `TimePickerDialog` não vem empacotado em um fragmento, portanto, você tem que fazer isto sozinho.

O código para `TimePickerDialogFragment` é quase idêntico a `DatePickerDialogFragment`, exceto que ele contém um `TimePickerDialog`, ao invés de um `DatePickerDialog`.

```
package com.dummies.android.taskreminder;

import android.app.Dialog;
import android.app.TimePickerDialog;
import android.app.TimePickerDialog.OnTimeSetListener;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;

public class TimePickerDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        Bundle args = getArguments();
        OnTimeSetListener listener = (OnTimeSetListener) getFragmentManager()
                .findFragmentByTag(
                        ReminderEditFragment.DEFAULT_EDIT_FRAGMENT_TAG);
        return new TimePickerDialog(getActivity(), listener,
                args.getInt(ReminderEditFragment.HOUR),
                args.getInt(ReminderEditFragment.MINS), false);
    }
}
```

Adicionando o fragmento para lidar com os callbacks dos seletores de data e hora

A etapa restante para fazer seus seletores de data e hora funcionarem é implementar `OnDateSetListener` e `OnTimeSetListener`, para que a caixa de diálogo armazene a nova data e hora escolhidas pelo usuário. Você armazena esse valor em um objeto `Calendar`. A operação requer três etapas:

1. **Adicione as interfaces** `OnDateSetListener` **e** `OnTimeSetListener` **a** `ReminderEditFragment`.

2. **Adicione um objeto `Calendar` chamado `mCalendar` para armazenar os valores que o usuário define.**

3. **Defina o objeto `mCalendar` quando o usuário seleciona uma data ou hora, e então atualize os botões de data e hora na interface do usuário com os novos valores.**

Na Listagem 11-4, adicione o código em negrito à classe `ReminderEditFragment` para fazer com que seus seletores de data e hora funcionem.

Listagem 11-4: Adicionando as interfaces OnDateSetListener e OnTimeSetListener à classe ReminderEditFragment

```
public class ReminderEditFragment extends Fragment     implements
        OnDateSetListener, OnTimeSetListener {                          →2

    private static final String DATE_FORMAT = "yyyy-MM-dd";
    private static final String TIME_FORMAT = "kk:mm";

    private Calendar mCalendar;                                         →7

    @Override
    public void onDateSet(DatePicker view, int year, int monthOfYear,
            int dayOfMonth) {                                           →11
        mCalendar.set(Calendar.YEAR, year);
        mCalendar.set(Calendar.MONTH, monthOfYear);
        mCalendar.set(Calendar.DAY_OF_MONTH, dayOfMonth);
        updateButtons();                                                →15
    }

    @Override
    public void onTimeSet(TimePicker view, int hour, int minute) {      →19
        mCalendar.set(Calendar.HOUR_OF_DAY, hour);
        mCalendar.set(Calendar.MINUTE, minute);
        updateButtons();
    }

    private void updateButtons() {                                      →25
        // Defina o texto do botão da hora
        SimpleDateFormat timeFormat = new SimpleDateFormat(TIME_FORMAT);
        String timeForButton = timeFormat.format(mCalendar.getTime());
        mTimeButton.setText(timeForButton);

        // Defina o texto do botão da hora
        SimpleDateFormat dateFormat = new SimpleDateFormat(DATE_FORMAT);
        String dateForButton = dateFormat.format(mCalendar.getTime());
        mDateButton.setText(dateForButton);
    }
}
```

A Listagem 11-4 funciona assim:

→ **2** Implementa os objetos de callback `OnDateSetListener` e `OnTimeSetListener`.

→ **7** Armazena as seleções de data e hora do usuário em um objeto `Calendar` Java.

→ **11** Implementa o método `onDateSet()` de `OnDateSetListener`. Quando o usuário define a data na caixa de diálogo, o objeto `Calendar` é atualizado para refletir a nova data.

→ **15** Atualiza os botões de interface do usuário para refletirem a nova data e hora.

→ **19** Define a hora e os minutos do objeto `Calendar` para o método `onTimeSet()` de `OnTimeSetListener`.

→ **25** Atualiza os botões para refletirem os valores selecionados pelo usuário. O objeto `Calendar` é convertido em algumas strings de texto – uma para a botão da data e outra para o botão da hora – que são, então, usadas para definir o texto nesses botões. O primeiro `SimpleDateFormat` usa `TIME_FORMAT` para produzir uma string de hora no formato `kk:mm`; o segundo `SimpleDateFormat` usa `DATE_FORMAT` para produzir uma string de data no formato `yyyy-MM-dd`.

Visite `http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html` (conteúdo em inglês) para obter mais detalhes sobre a formatação da data e da hora `SimpleDateFormat`.

Para adicionar o objeto `mCalendar`, acrescente as seguintes linhas a `onCreate()`:

```
if (savedInstanceState != null
        && savedInstanceState.containsKey(CALENDAR)) {                  →2
    mCalendar = (Calendar) savedInstanceState.getSerializable(CALENDAR);→3
} else {
    mCalendar = Calendar.getInstance();                                 →5
}
```

O código funciona assim:

→ **2** Esta linha verifica `savedInstanceState` do campo `CALENDAR`, indicando que uma atividade anterior foi salva e destruída temporariamente, tal como girar do modo Paisagem para o modo Retrato.

Se você estiver criando uma instância novinha deste fragmento, `savedInstanceState` será nula e o aplicativo irá para a linha 5.

→ **3** Se `savedInstanceState` incluir um campo `CALENDAR`, esta linha obterá o objeto `Calendar` associado e definirá `mCalendar` para esse valor.

→ **5** Quando você está criando um `ReminderEditFragment` a partir do zero e não está utilizando uma instância anterior, esta linha define `mCalendar` para uma nova instância `Calendar` (obtida chamando `Calendar.getInstance()`). As novas instâncias `Calendar` sempre têm como padrão a data e a hora atuais.

Agora, a instância `mCalendar` é inicializada.

Como `onCreate()` está procurando uma instância do calendário em `savedInstanceState` nas linhas 1-2, você tem que salvar seu objeto `mCalendar` em `savedInstanceState` sempre que sua atividade é destruída. Assim, `onCreate()` poderá selecionar o valor salvo se a atividade for recriada. Para tanto, adicione o seguinte método à sua classe:

```
@Override
public void onSaveInstanceState(Bundle outState) {                    →2
    super.onSaveInstanceState(outState);

    // Salva a instância do calendário no caso do usuário tê-la alterado  →4
    outState.putSerializable(CALENDAR, mCalendar);
}
```

O código funciona assim:

→ **2** `onSaveInstanceState()` é um método especial que o Android chama sempre que está para destruir um fragmento. Você pode armazenar qualquer dado que possa estar usando, portanto, se o Android precisar recriar a mesma atividade, terá todas as informações necessárias.

A maioria das exibições já sabe como armazenar seu estado e restaurar a si mesmas, e fazem isso na chamada para `super.onSaveInstanceState()`.

→ **4** A única tarefa com a qual você tem que lidar manualmente é o campo `mCalendar`, portanto, esta linha armazena-o no objeto bundle para usá-lo posteriormente quando a atividade for recriada.

Os objetos do calendário podem ser serializados (armazenados como dados), portanto, esta linha usa o método `putSerializable()` para salvá-los.

Salvando os nomes do campo no Android

As atividades e os fragmentos do Android não são como os objetos Java, onde você pode armazenar informações em um campo no objeto e esperar que elas sempre estejam lá. Normalmente no Java, se um objeto `person` (pessoa) for definido para o nome "`Michael`", você poderá esperar que o nome sempre seja "`Michael`", mas de modo surpreendente, este nem sempre é o caso no Android.

Diferentemente no Java, o Android pode destruir as atividades e os fragmentos a qualquer momento. Esses elementos podem também ser recriados depois – e uma atividade recriada precisa parecer igual àquela que nunca foi destruída e recriada. O Android se reserva o direito de destruir os objetos quando a memória está ficando baixa, mas ele mantém a capacidade de recriá-los depois, para oferecer ao usuário uma experiência uniforme.

Se você armazenar a string "`Michael`" em um campo chamado `name`, esse campo não será salvo automaticamente se a atividade ou o fragmento for destruído e recriado. Você terá que salvar o campo manualmente, armazenando-o em um bundle em `onSaveInstanceState()` e restaurando-o a partir do grupo `savedInstanceState` em `onCreate()`.

Lembre-se: Sempre que você adiciona um campo a uma atividade ou um fragmento, deve adicionar o devido código aos métodos `onSavedInstanceState()` e `onCreate()` para salvá-lo e restaurá-lo – do contrário, seu aplicativo se comportará de modo estranho em algumas circunstâncias, mas não em outras.

Descubra mais informações sobre a serialização Java em `http://java.sun.com/developer/technicalArticles/Programming/serialization` (conteúdo em inglês).

Você pode salvar todos os outros tipos, tais como ints, longs, strings, parcelables e outros elementos exóticos em bundles, portanto, verifique `http://d.android.com/reference/android/cs/Bundle.html` para ver a lista completa (conteúdo em inglês).

Criando uma Caixa de Diálogo de Alerta

De tempos em tempos, pode ser necessário alertar o usuário sobre algo que aconteceu. No aplicativo Task Reminder, talvez você queira exibir uma mensagem de boas-vindas e oferecer instruções sobre como criar uma tarefa. O sistema Android tem uma estrutura, construída em torno das caixas de diálogo, que fornece a implementação da qual você poderá precisar.

Vários tipos de caixas de diálogo estão disponíveis:

- **Alerta:** Notifica o usuário sobre uma ocorrência importante. Também permite que você defina o valor de texto de um botão e a ação a ser executada quando ele é clicado. Como desenvolvedor, você pode

  fornecer a `AlertDialog` uma lista de itens a exibir, permitindo que o usuário selecione um item em uma lista.

- **Progresso:** Usada para exibir um círculo ou barra de progresso. Este tipo de caixa de diálogo é criada via classe `ProgressDialog`.

- **Personalizada:** Uma caixa de diálogo personalizada, criada e programada por você, o desenvolvedor Android mestre. Você cria uma classe de caixa de diálogo personalizada estendendo a classe `Dialog` de base ou usando arquivos XML de layout personalizados.

Vendo por que você deve trabalhar com caixas de diálogo

Se você nunca trabalhou com um aplicativo que falhou em alertá-lo ou avisá-lo devidamente sobre algo que tenha ocorrido, considere o exemplo de um cliente de e-mail não notificando que você tem um novo e-mail. Isso te chatearia? Alertar os usuários quanto a questões importantes ou escolhas que precisam ser feitas é parte integrante de qualquer experiência do usuário.

Esta lista fornece alguns exemplos de como usar uma caixa de diálogo para informar ao usuário sobre uma mensagem ou uma ação necessária:

- Algo está acontecendo em segundo plano (`ProgressDialog` faz isto.)
- Os valores em uma exibição `EditText` são inválidos.
- A rede ficou indisponível.
- O usuário precisa selecionar uma data ou uma hora (como no aplicativo Task Reminder).
- O estado do telefone é incompatível com o aplicativo (pode precisar ter o GPS ativado ou um cartão SD adicionado, por exemplo).
- O usuário precisa escolher um item em uma lista.

Embora essa lista não seja completa, dá uma ideia do que é possível fazer com as caixas de diálogo.

Quando você trabalha com qualquer tipo de processo bloqueante (comunicação da rede ou tarefas de longa execução, por exemplo), sempre forneça ao usuário uma caixa de diálogo ou indicador de progresso. Um usuário que não percebe que um elemento de seu aplicativo precisa de atenção, provavelmente acreditará por engano que ele parou de responder e poderá até desinstalá-lo. A estrutura Android fornece vários indicadores de progresso, tais como as classes de progresso comuns `ProgressDialog` e `ProgressBar`.

Embora uma análise da classe `AsyncTask` esteja além do escopo deste livro, você usaria essa classe para gerenciar as tarefas de longa execução enquanto atualiza a interface do usuário. Veja o tutorial útil "Painless Threading" (Threading sem Problemas) em `http://android-developers.blogspot.com/2009/05/painless-threading.html` (conteúdo em inglês). Você também pode criar um novo thread no código — a classe `AsyncTask` ajuda a simplificar este processo.

Escolhendo a devida caixa de diálogo para uma tarefa

Embora você determine qual caixa de diálogo usar para certa situação, você pode fazer uma série lógica de perguntas para escolher a adequada.

1. **Esta é uma tarefa de longa execução?**

   - *Sim*: Use `ProgressDialog` para permitir que o usuário saiba que algo está acontecendo em segundo plano e que o aplicativo não está congelado. Um ótimo recurso que explica como fazer isto está localizado aqui: `http://d.android.com/guide/topics/ui/dialogs/html#ProgressDialog` (conteúdo em inglês)

   - *Não:* Continue na Etapa 2.

2. **O usuário precisa ser capaz de executar uma ação avançada na caixa de diálogo?**

   Uma *ação avançada* não é suportada pela classe `AlertDialog`.

   - *Sim:* Crie uma classe `Dialog` personalizada estendendo a classe `Dialog` de base ou criando uma a partir do arquivo XML de layout personalizado. Você pode encontrar mais informações sobre as caixas de diálogo personalizadas em `http://d.android.com/guide/topics/ui/dialogs/html#CustomDialog` (conteúdo em inglês).

   - *Não:* Continue na Etapa 3.

3. **O usuário precisa responder a uma pergunta, tal como, "Tem certeza?", com um valor Yes (Sim) ou No (Não)?**

   - *Sim:* Crie uma `AlertDialog` e reaja aos botões em `AlertDialog` usando as chamadas `onClickListener()`.

   - *Não:* Continue na Etapa 4.

4. **O usuário precisa fazer uma seleção em uma lista simples de itens?**

   - *Sim:* Crie uma `AlertDialog`.

- *Não:* Continue na Etapa 5.

5. **O usuário simplesmente precisa ser alertado?**

   - *Sim:* Crie uma `AlertDialog` simples.
   - *Não:* Você pode não estar precisando de uma caixa de diálogo, se puder notificar o usuário de outro modo.

Criando sua própria caixa de diálogo de alerta

Às vezes, você precisa notificar o usuário sobre informações importantes apresentando uma caixa de diálogo. O Android facilitou essa tarefa com a introdução da classe `AlertDialog.Builder`, que permite criar facilmente uma `AlertDialog` com várias opções e botões. Seu aplicativo pode reagir a esses cliques do botão via `onClickListener()` de cada botão.

Você não precisa usar a classe `AlertDialog.Builder` em um aplicativo simples, tal como o Task Reminder. Porém, a Listagem 11-5 mostra como criar uma quando você cria aplicativos mais complexos.

Suponha que o usuário tenha tocado no botão Save (Salvar) no aplicativo Task Reminder e você deseja abrir uma janela (parecida com a mostrada na Figura 11-3) para que ele possa confirmar a operação.

Na Listagem 11-5, você cria um objeto `AlertDialog` usando a classe `AlertDialog.Builder`, e então adiciona `AlertDialogFragment` (que funciona de modo parecido com `DatePickerDialogFragment` e `TimePickerDialogFragment`).

Listagem 11-5: Criando um AlertDialogFragment com a classe AlertDialog.Builder

```
public class AlertDialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder
        = new AlertDialog.Builder(getActivity());                    → 5
    builder.setMessage("Are you sure you want to save the task?")    → 6
            .setTitle("Are you sure?")                               → 7
            .setCancelable(false)                                    → 8
            .setPositiveButton("Yes",                                → 9
             new DialogInterface.OnClickListener() {                 →10
                public void onClick(DialogInterface dialog, int id) {
                        //  Execute alguma ação, tal como salvar o item  →12
                }
             })
            .setNegativeButton("No", new DialogInterface.OnClickListener() {→15
                public void onClick(DialogInterface dialog, int id) {
                                dialog.cancel();                     →17
             }
    });
    return builder.create();                                         → 20
    }
}
```

Figura 11-3: A janela `AlertDialog` de confirmação.

O código é explicado nesta lista:

→ **5** Configura a classe `AlertDialog.Builder` com o contexto de `AlertDialog.Builder` como a atividade em execução atual.

→ **6** Especifica a mensagem a mostrar no centro de `AlertDialog` (como mostrado na Figura 11-3). O valor pode ser uma string ou um recurso de string.

→ **7** Define o título de `AlertDialog`. O valor pode ser uma string ou um recurso de string.

→ **8** Define o valor de cancelable (cancelável) para `false`, requerendo que o usuário selecione um botão em `AlertDialog`. Se este flag for definido para `false`, o usuário não poderá tocar no botão Back (Voltar) no dispositivo para sair de `AlertDialog`. Defina-o para `true` e o usuário poderá tocar no botão Back.

→ **9** Especifica o texto no botão positivo. O usuário clica no botão Yes (Sim) para realizar a ação indicada na linha 10. Este valor pode ser uma string ou um recurso de string.

→ **10** Um bloco de código (terminando na linha 12) que define `onClickListener()` para o botão Yes (Sim). O código na linha 12 é executado quando o botão é tocado.

→ **15** Especifica o texto no botão negativo. Esse botão indica que o usuário não deseja executar a ação sendo solicitada via `AlertDialog`. O valor de texto nesse botão é definido para No (Não). Pode ser uma string ou um recurso de string.

→ **17** Define `onClickListener()` para o botão negativo. O atendente fornece uma referência para a caixa de diálogo que está sendo mostrada. É chamado no método `cancel()` do objeto `Dialog` para fechar a caixa de diálogo quando o usuário clica em No (Não) em `AlertDialog`.

→ **20** Notifica o Android para criar `AlertDialog` via método `create()`.

Para mostrar a caixa de diálogo, você inicia uma transação de fragmento da maneira usual:

```
FragmentTransaction ft = getFragmentManager().beginTransaction();
DialogFragment newFragment = new AlertDialogFragment();
newFragment.show(ft, "alertDialog");
```

Criar uma caixa de diálogo através da classe `AlertDialog.Builder` é mais fácil do que ter que derivar sua própria classe `Dialog`. Se possível, crie sua caixa de diálogo com a classe `AlertDialog.Builder` porque

ela fornece ao seu aplicativo uma experiência de usuário consistente que é familiar para a maioria dos usuários Android.

Quando o usuário toca no botão Save (Salvar ou qualquer botão ao qual o código está anexado), um `AlertDialog` é aberto para que o usuário possa confirmar a gravação da tarefa. Esse dado provavelmente será armazenado em um banco de dados, como tratado no Capítulo 12.

Você pode encontrar exemplos úteis de uso de outras opções na classe `Dialog` em `http://d.android.com/guide/topics/ui/dialogs.html` (conteúdo em inglês).

Validando a Entrada

Depois de ter criado seu aplicativo para que os usuários possam inserir informações (e talvez você já tenha criado o mecanismo para salvar o conteúdo em um banco de dados ou servidor remoto), o que acontece quando o usuário insere um texto inválido ou nenhum texto? Agora, a validação da entrada entra em cena.

A *validação da entrada* verifica a entrada antes que a gravação ocorra. Se um usuário deixar de inserir o texto para o título ou a mensagem e tentar salvar, deverá ele ser bem sucedido? Claro que não.

Cabe a você fornecer o método responsável pela validação da entrada. Eis alguns métodos comuns:

- `EditText.setError()`: Se você detectar que o usuário tentou inserir um texto inválido em um campo, simplesmente chame `setError()` e transmita a mensagem de erro. Então, o Android colocará em `EditText` um ícone de erro e exibirá uma mensagem de notificação do problema. Essa mensagem ficará na tela até o usuário mudar o valor do campo ou até você chamar `setError(null)`.
- `TextWatcher`: Implemente um `TextWatcher` na exibição `EditText`. Esta classe fornece callbacks para você sempre que o texto muda na exibição `EditText`. Portanto, você pode examinar o texto em cada tecla pressionada.
- `On Save`: Quando o usuário tentar salvar um formulário, examine todos os campos dele nesse momento e informe ao usuário sobre qualquer problema encontrado.
- `OnFocusChanged()`: Examine os valores do formulário quando o evento `onFocusChanged()` for chamado — o que ocorre quando a exibição ganha e quando perde o foco. Geralmente, é um bom lugar para configurar a validação.

O aplicativo Task Reminder não fornece nenhuma validação de entrada. Contudo, você pode adicionar a validação via um dos métodos descritos anteriormente.

Informando o usuário

O modo mais comum para informar ao usuário sobre um problema em potencial, tal como um erro no valor de entrada, é exibir uma mensagem `Toast`. Esse tipo de mensagem aparece na tela por apenas alguns segundos por padrão.

Fornecer uma mensagem `Toast` é tão simples quanto implementar o código a seguir, onde você informa ao usuário sobre um erro de entrada:

```
Toast.makeText(getActivity(), "Title must be filled in", Toast.LENGTH_SHORT).
              show();
```

Você pode mostrar essa mensagem quando o usuário falha em inserir um título no campo de título e clica no botão Save (Salvar).

O único problema com uma mensagem `Toast` é que ela tem curta duração por padrão. Um usuário que estiver olhando para outro lado na hora errada, provavelmente não irá vê-la. Você pode configurar suas mensagens `Toast` para serem exibidas por mais tempo usando `Toast.LENGTH_LONG`, ao invés de `Toast.LENGTH_SHORT`.

Usando outras técnicas de validação

Uma mensagem `Toast` não é a única maneira de informar aos usuários sobre um problema com sua entrada. Algumas outras técnicas de validação populares são descritas nesta lista:

- `AlertDialog`: Crie uma instância de um `AlertDialog` que informa ao usuário sobre os erros. Esse método assegura que o usuário verá a mensagem de erro porque o alerta deve ser cancelado ou aceito.
- **Destaque do campo de entrada:** Se o campo for inválido, a cor do segundo plano do campo de entrada (exibição `EditText`) poderá mudar para indicar que o valor está incorreto.
- **Validação personalizada:** Se você estiver querendo arriscar, poderá criar uma biblioteca de validação personalizada para lidar com todo tipo de validação. Ela pode destacar o campo e desenhar pequenas exibições com setas apontando para o erro, por exemplo, parecido com a validação da janela de conexão Google quando você conecta um dispositivo pela primeira vez.

Você pode usar esses métodos comuns para exibir as informações de validação da entrada ou pode imaginar novas maneiras de informar aos usuários sobre os erros. Por exemplo, o Capítulo 14 apresentará a barra de notificação, que você pode usar para informar aos usuários sobre um problema com um serviço em segundo plano.

Capítulo 12

Sendo Persistente com o Armazenamento de Dados

Neste Capítulo

Descobrindo a mídia de armazenamento de dados
Obtendo as permissões do usuário
Criando um banco de dados SQLite
Consultando seu banco de dados

Em certos tipos de aplicativos, o Android requer que os desenvolvedores usem a persistência de dados, em que as informações sobre as preferências de um usuário, tais como, as cores favoritas do segundo plano ou as estações de rádio, são salvas no dispositivo para reutilização posterior, depois do dispositivo ser desligado e ligado mais tarde. Por exemplo, o aplicativo Task Reminder não seria útil se não salvasse as tarefas, não é? Felizmente, a plataforma Android (em combinação com o Java) fornece um conjunto robusto de ferramentas que você pode usar para armazenar os dados do usuário.

Este capítulo entra profundamente na criação e na atualização de um banco de dados SQLite e na produção de um `ContentProvider` para acessá-lo. Você precisa estar familiarizado em certo nível com a teoria de banco de dados para cuidar das tarefas do armazenamento de dados neste capítulo.

Se você não estiver familiarizado com o SQL (Linguagem de Consulta Estruturada) ou o banco de dados SQL, veja o site Web SQLite em `www.sqlite.org` (conteúdo em inglês) para obter mais informações.

Este capítulo utiliza muito código — se você começar a se sentir perdido, poderá fazer download do código-fonte completo do aplicativo no site web deste livro.

Encontrando Lugares para Colocar os Dados

Dependendo dos requisitos de seu aplicativo, você pode precisar armazenar os dados em vários lugares. Por exemplo, se um aplicativo interage com arquivos de música e um usuário deseja reproduzi-los em mais de um programa de música, você terá que armazená-los em um local onde todos os aplicativos possam acessá-los. Um aplicativo que precisa armazenar dados sensíveis, tais como nomes de usuário e senhas criptografados, não deve compartilhar dados — colocá-los em um ambiente de armazenamento local e seguro será a melhor estratégia. Independentemente de sua situação, o Android fornece várias opções para armazenar os dados.

Examinando suas opções de armazenamento

O ecossistema Android fornece vários lugares onde os dados podem ser mantidos:

- **Preferências compartilhadas:** Dados privados, armazenados em pares de chave-valor (veja o Capítulo 15 para descobrir como lidar com as preferências compartilhadas).
- **Armazenamento interno:** Um local para salvar os arquivos no dispositivo. Os arquivos armazenados no armazenamento interno são exclusivos de seu aplicativo por padrão e os outros aplicativos não podem acessá-los (nem o usuário, exceto usando seu aplicativo). Quando o aplicativo é desinstalado, os arquivos privados são apagados também.
- **Cache local:** O diretório interno de dados para armazenamento em cache, ao invés de armazená-los de modo permanente. Os arquivos em cache podem ser apagados a qualquer momento. Você usa o método `getCacheDir()`, disponível nos objetos `Activity` ou `Context` no Android.

Se você armazenar os dados em um diretório interno de dados e o espaço do armazenamento interno ficar pequeno, o Android poderá apagar os arquivos para obter espaço. Porém, não conte com o Android para apagar seus arquivos! Você mesmo deve apagar seus arquivos em cache dentro de um limite razoável (cerca de 1 MB) de espaço consumido no diretório de cache.

- **Armazenamento externo:** Todo dispositivo Android suporta o armazenamento externo compartilhado para os arquivos — armazenamento removível, tal como um cartão Secure Digital (cartão SD) ou um armazenamento não removível. Os arquivos salvos no armazenamento externo são *públicos* (qualquer pessoa ou aplicativo pode alterá-los) e nenhum nível de segurança é aplicado. Os usuários podem modificar os arquivos usando um aplicativo de gerenciador de arquivos ou conectando o dispositivo a um computador via cabo USB e montando o dispositivo como um armazenamento externo. Antes de você trabalhar com o armazenamento externo, verifique o estado atual dele com o objeto Environment, usando uma chamada para `getExternalStorageState()` para verificar se a mídia está disponível.

  No Android 2.2, um novo conjunto de métodos foi introduzido para lidar com os arquivos externos. O principal método é uma chamada ao objeto `Context` — `getExternalFilesDir()`. Essa chamada aceita um parâmetro de string como uma chave para ajudar a definir o tipo de mídia que você está salvando, tal como tons de chamada, música ou fotos. Para obter mais informações, veja os exemplos de armazenamento de dados externos e os documentos em `http://d.android.com/guide/topics/data/datastorage.html#filesExternal` (conteúdo em inglês).

- **Banco de dados SQLite:** Uma implementação leve de banco de dados SQL que está disponível em várias plataformas (inclusive Android, iPhone, Windows, Linux e Mac) que é totalmente suportada pelo Android. Você pode criar tabelas e realizar consultas SQL nelas. Você implementará um banco de dados SQLite neste capítulo para lidar com a persistência das tarefas no aplicativo Task Reminder.

- **Provedor de conteúdo:** Um "componente" (wrapper) em torno de outro mecanismo de armazenamento. Um provedor de conteúdo é usado por um aplicativo para ler e gravar os dados do aplicativo que podem ser armazenados nas preferências, arquivos ou banco de dados SQLite, por exemplo.

- **Conexão de rede:** (Também conhecida como armazenamento remoto). Qualquer fonte de dado remota à qual você tem acesso. Por exemplo, como o Flickr exibe uma API que o permite armazenar imagens em seus servidores, seu aplicativo pode trabalhar com o Flickr para armazenar imagens. Se seu aplicativo trabalhar com uma ferramenta popular na Internet (tal como, o Twitter, Facebook ou Basecamp), ele poderá enviar informações via HTTP — ou qualquer outro protocolo considerado necessário — para as APIs de terceiros para armazenar os dados.

Escolhendo uma opção de armazenamento

Vários locais de armazenamento de dados oferecem muitas opções. Contudo, você tem que descobrir qual usar e pode até querer utilizar *diversos* mecanismos de armazenamento.

Suponha que seu aplicativo se comunica com uma API remota de terceiros, tal como, o Twitter, e a comunicação de rede seja lenta e não totalmente confiável. Você poderá querer manter no servidor uma cópia local de todos os dados, desde a última atualização, para permitir que o aplicativo permaneça útil (de algum modo) até a próxima atualização. Quando você armazena dados em uma cópia local de um banco de dados SQLite e o usuário inicia uma atualização, as novas atualizações renovam o banco de dados SQLite com novos dados.

Se seu aplicativo contar unicamente com a comunicação de rede para a recuperação e o armazenamento de informações, use o banco de dados SQLite (ou qualquer outro mecanismo de armazenamento) para deixar o aplicativo útil quando o usuário não puder conectar uma rede e tiver que trabalhar off-line — uma ocorrência comum. Se seu aplicativo não funcionar quando uma conexão de rede estiver indisponível, provavelmente você terá avaliações negativas na Google Play Store — assim como solicitações de recursos para fazer seu aplicativo funcionar offline. Esta estratégia gera muito trabalho extra no processo de desenvolvimento do aplicativo, mas vale a pena dez vezes em termos de experiência do usuário.

Criando o SQLite ContentProvider de Seu Aplicativo

O melhor lugar para armazenar e recuperar as tarefas de um usuário no aplicativo Task Reminder é dentro de um banco de dados SQLite. Seu aplicativo precisa ser capaz de executar as tarefas CRUD — create, read, update e delete (criar, ler, atualizar e excluir) — no banco de dados usando `ContentProvider`.

Compreendendo como o SQLite ContentProvider funciona

Os dois fragmentos no aplicativo Task Reminder precisam realizar várias tarefas para operar. O `ReminderEditFragment` precisa completar estas tarefas:

1. Criar um novo registro.
2. Ler um registro para que ele possa exibir os detalhes para edição.
3. Atualizar o registro existente.

O `ReminderListFragment` precisa realizar estas tarefas:

1. Ler todas as tarefas para exibi-las na tela.
2. Apagar uma tarefa, respondendo ao evento de clique a partir do menu contextual após o usuário ter pressionado um item por um tempo.

Para trabalhar com um banco de dados SQLite, você se comunicará com o banco de dados via `ContentProvider`. Os programadores comumente removem tanto quanto for possível a comunicação com o banco de dados dos objetos `Activity` e `Fragment`. Os mecanismos do banco de dados são colocados em um `ContentProvider` para ajudar a separar o aplicativo em camadas de funcionalidade. Portanto, se você precisar alterar o código que afeta o banco de dados, saberá que precisa alterar o código em apenas um local.

Criando um ContentProvider para manter o código do banco de dados

Para criar um `ContentProvider` em seu projeto Android que manterá o código centrado no banco de dados, primeiro nomeie o arquivo como `ReminderProvider.java`.

Definindo os principais elementos de um banco de dados

Antes de criar e abrir um banco de dados, você precisa definir alguns campos-chave. Substitua o código em sua classe `ReminderProvider` pelo código da Listagem 12-1.

Listagem 12-1: As constantes, campos e construções da classe RemindersDbAdapter

```
package com.dummies.android.taskreminder;

import android.content.ContentProvider;
import android.database.sqlite.SQLiteDatabase;

public class ReminderProvider extends ContentProvider {
    // Constantes relacionadas ao banco de dados
    private static final int DATABASE_VERSION = 1;                      →8
    private static final String DATABASE_NAME = "data";                 →9
    private static final String DATABASE_TABLE = "reminders";           →10

    // Colunas do banco de dados
    public static final String COLUMN_ROWID = "_id";                    →13
    public static final String COLUMN_DATE_TIME = "reminder_date_time"; →14
    public static final String COLUMN_BODY = "body";                    →15
    public static final String COLUMN_TITLE = "title";                  →16
```

```
    private static final String DATABASE_CREATE = "create table "          →18
         + DATABASE_TABLE + " (" + COLUMN_ROWID
         + " integer primary key autoincrement, " + COLUMN_TITLE
         + " text not null, " + COLUMN_BODY + " text not null, "
         + COLUMN_DATE_TIME + " integer not null);";

    private SQLiteDatabase mDb;                                            →24

    @Override
    public boolean onCreate() {                                            →27
       mDb = new DatabaseHelper(getContext()).getWritableDatabase();       →28
       return true;
    }

}
```

As linhas numeradas são explicadas nesta lista:

- **→ 8** A versão do banco de dados. Se você fosse atualizar o esquema em seu banco de dados, aumentaria a versão e forneceria uma implementação do método `onUpgrade()` do `DatabaseHelper`.
- **→ 9** O nome físico do banco de dados que existirá no sistema de arquivos do Android.
- **→ 10** O nome da tabela do banco de dados que manterá as tarefas.
- **→ 13-16** Define os nomes das colunas da tabela do banco de dados.
- **→ 18** Define o script de criação do banco de dados. Os nomes de coluna das linhas anteriores são combinados em uma única instrução SQL que criará o banco de dados.
- **→ 24** A instância em nível de classe do objeto do banco de dados SQLite que permite criar, ler, atualizar e apagar os registros.
- **→ 27** Cria `ContentProvider` e chama `onCreate()`.
- **→ 28** Chama `getWritableDatabase()` em um objeto `DatabaseHelper`.

O banco de dados SQL está pronto para ser criado!

Para obter informações sobre a tabela do banco de dados ou componentes do script, veja a próxima seção "Visualizando a tabela SQL". Para obter informações sobre o auxílio do banco de dados ou a tabela do banco de dados, veja a seção "Criando a tabela do banco de dados".

Visualizando a tabela SQL

O *objeto de tabela* no SQL é a construção que mantém os dados que você gerencia. Visualizar uma tabela no SQLite é parecido com ver uma planilha: Cada linha consiste em dados e cada coluna representa os dados dentro da linha. Anteriormente neste capítulo, a Listagem 12-1 definiu os nomes das colunas para o banco de dados. Esses nomes são iguais aos valores do

cabeçalho em uma planilha, como mostrado na Figura 12-1. Cada linha contém um valor para cada coluna, e essa é a forma como os dados são armazenados no SQLite.

Figura 12-1: Visualizando os dados no aplicativo Task Reminder.

_id	title	body	reminder_date_time
1	Order Flight Tickets	Go to travel site and order ti...	2010-11-15 16:15
2	Schedule Time Off	Email manager at work to ...	2010-11-17 15:00
3	Take Vacation	YES! Finally take that much n...	2010-12-10 14:30
4	Pay Bills	Pay the bills through bill pay.	2010-11-10 12:15

A linha 18 na Listagem 12-1 reúne o script `create` do banco de dados, que concatena várias constantes de dentro do arquivo para criar um script de criação do banco de dados. Quando você executa esse script no SQLite, ele cria uma tabela chamada `reminders` em um banco de dados chamado `data`. As colunas e como elas são construídas no script de criação são descritas nesta lista:

- `create table DATABASE_TABLE`: Esta parte do script notifica o SQLite que você deseja criar uma tabela do banco de dados chamada `reminders`.
- `COLUMN_ROWID`: Esta propriedade age como o identificar da tarefa. Esta coluna tem os atributos `integer primary key autoincrement` aplicados. O atributo `integer` especifica que a linha é um inteiro. O atributo `primary key` determina que `COLUMN_ROWID` é o identificador primário de uma tarefa. O atributo `autoincrement` notifica o SQLite, sempre que uma nova tarefa é inserida, para simplesmente definir automaticamente o ID da linha para o próximo inteiro disponível. Por exemplo, se as linhas 1, 2 e 3 existirem e você inserir outro registro, o valor de `COLUMN_ROWID` na próxima linha será 4.
- `COLUMN_TITLE`: O usuário fornece o título da tarefa, tal como *Schedule Vacation* (Agendar Férias). O atributo de texto informa ao SQLite que a coluna é uma coluna de texto. O atributo `not null` determina que o valor dessa coluna não pode ser nulo — o usuário deve fornecer um valor.
- `COLUMN_BODY`: Este é o corpo ou a descrição da tarefa. Os atributos desta coluna são os mesmos de `COLUMN_TITLE`.
- `COLUMN_DATE_TIME`: A data e a hora do lembrete são armazenadas neste campo. Armazena um inteiro porque o SQLite não tem classes de armazenamento associadas a datas ou horas, portanto, você converte o objeto `Calendar` em um `long` Java, que pode ser representando — sem problemas — como um inteiro SQL.

Para obter mais informações sobre as datas e as horas no SQLite, visite `www.sqlite.org/datatype3.html#datetime` (conteúdo em inglês).

Criando a tabela do banco de dados

Quando você estiver pronto para criar a tabela do banco de dados, fornecerá uma implementação de `SQLiteOpenHelper`. O tipo de classe `ReminderProvider`, mostrado na Listagem 12-1, permite que você crie uma classe Java aninhada dentro da classe `ReminderDbAdapter`.

Atualizando seu banco de dados

Suponha que você lance seu aplicativo e 10.000 usuários instalem-no e usem-no — e eles adoram! Alguns até enviam solicitações de recursos, portanto, você implementa um que requer uma alteração no esquema do banco de dados. Então, você executa instruções SQL `ALTER` dentro da chamada `onUpgrade()` para atualizar seu banco de dados.

Você poderia atualizar o banco de dados "descartando" o existente, e então criando um novo. Mas você não deseja fazer isso — descartar um banco de dados *apaga todos os dados do usuário*. Imagine atualizar seu aplicativo Task Reminder favorito, apenas para ver que a atualização apagou todas as tarefas preexistentes (um defeito *grave*).

Listagem 12-2: Criando uma tabela do banco de dados

```
private static class DatabaseHelper extends SQLiteOpenHelper {          →1
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);      →3
        }

        @Override
        public void onCreate(SQLiteDatabase db) {                       →7
            db.execSQL(DATABASE_CREATE);                                →8
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion,
            int newVersion) {                                           →12
            throw new UnsupportedOperationException();
        }
}
```

As linhas numeradas são descritas nesta lista:

- **→ 1** A implementação de `SQLiteOpenHelper`.
- **→ 3** A chamada feita para o construtor de base `SQLiteOpenHelper`. Essa chamada cria, abre e/ou gerencia um banco de dados, que não é criado ou aberto até `getReadableDatabase()` ou `getWriteableDatabase()` ser chamado na instância `SQLiteOpenHelper`.
- **→ 7** O método `onCreate()`, que é chamado quando o banco de dados é criado pela primeira vez.

→ **8** Cria seu banco de dados e tabela do banco de dados ("onde a mágica acontece"). O método `execSQL()` aceita uma string de script SQL como um parâmetro. O banco de dados SQLite executa o SQL da Listagem 12-1 para criar a tabela do banco de dados.

→ **12** Usa o método `onUpgrade()` quando você precisa atualizar um banco de dados existente.

Determinando as URLs do ContentProvider

Um `ContentProvider` Android usa URLs para identificar os dados. Geralmente, você pode usar uma URL para identificar uma parte específica dos dados, tal como, um único lembrete ou todos os lembretes em seu banco de dados. Se você armazenar outros tipos de dados nele, poderá usar URLs para eles também.

Em seu aplicativo, você usa dois tipos de URLs — `content://com.dummies.android.taskreminder.ReminderProvider/reminder` para recuperar uma lista de todos os lembretes em seu banco de dados ou `content://com.dummies.android.taskreminder.ReminderProvider/reminder/9` para recuperar um lembrete específico no banco de dados (neste caso, o lembrete com o ID 9).

Estas URLs do provedor de conteúdo são indubitavelmente parecidas com as URLs com as quais você está familiarizado. Suas principais diferenças são descritas nesta lista:

- **content://:** Um `contentProvider` começa com `content://`, ao invés de `http://`.
- **com.dummies.android.taskreminder.ReminderProvider:** A segunda parte da URL identifica a autoridade (`ReminderProviderContentProvider`) do conteúdo. Embora essa string possa ter qualquer conteúdo, convenciona-se usar um nome totalmente qualificado de seu `ContentProvider`.
- **reminder:** A terceira parte da URL identifica o caminho — neste caso, o tipo de dado que você está pesquisando. Essa string identifica qual tabela no banco de dados deve ser lida. Se o aplicativo armazenar diversos tipos no banco de dados (digamos, uma lista de usuários além de uma lista de lembretes), um segundo tipo de caminho poderia ser nomeado como `user`.
- **9:** Na primeira URL, o caminho termina com `reminder`. Porém, na segunda URL, o caminho continua para incluir o ID específico do lembrete sendo solicitado.

Antes de você poder usar `ContentProvider`, certifique-se de que ele esteja listado no arquivo `AndroidManifest.xml`, adicionando este código antes da tag `</application>`:

```
<provider
    android:name="com.dummies.android.taskreminder.ReminderProvider"
    android:authorities="com.dummies.android.taskreminder.ReminderProvider"
    android:exported="false"
/>
```

Ele informa ao Android que um `ContentProvider` chamado `ReminderProvider` lidará com as URLs que usam a autoridade específica `com.dummies.android.taskreminder.ReminderProvider`. Também indica que os dados no provedor não são exportados para outros aplicativos no telefone do usuário. Em geral, você deve definir `exported="false"`, a menos que queira tornar seu provedor disponível para outros aplicativos.

Agora, você tem que adicionar o código para suportar essas URLs em seu `ContentProvider`. Abra `ReminderProvider` e adicione as seguintes linhas à classe:

```
// Uri e autoridade do provedor de conteúdo
public static String AUTHORITY = "com.dummies.android.taskreminder.
      ReminderProvider";                                              →2
public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
    + "/reminder");                                                   →4

// Tipos MIME usados para pesquisar palavras ou procurar uma definição
public static final String REMINDERS_MIME_TYPE =
     ContentResolver.CURSOR_DIR_BASE_TYPE
    + "/vnd.com.dummies.android.taskreminder.reminder";               →8
public static final String REMINDER_MIME_TYPE = ContentResolver.CURSOR_
     ITEM_BASE_TYPE
    + "/vnd.com.dummies.android.taskreminder.reminder";

// Coisas do UriMatcher
private static final int LIST_REMINDER = 0;                           →13
private static final int ITEM_REMINDER = 1;
private static final UriMatcher sURIMatcher = buildUriMatcher();      →15

/**
* Constrói um UriMatcher para a sugestão de pesquisas e atualizar com rapidez
* as consultas.
*/
private static UriMatcher buildUriMatcher() {
     UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);        →22
     matcher.addURI(AUTHORITY, "reminder", LIST_REMINDER);            →23
     matcher.addURI(AUTHORITY, "reminder/#", ITEM_REMINDER);          →24
     return matcher;
}

/**
* Este método é requerido para consultar os tipos suportados. Também
* é útil no método query() para determinar o tipo de Uri recebido.
*/
@Override                                                             →33
```

```
public String getType(Uri uri) {
     switch (sURIMatcher.match(uri)) {
     case LIST_REMINDER:
        return REMINDERS_MIME_TYPE;
     case ITEM_REMINDER:
        return REMINDER_MIME_TYPE;
     default:
        throw new IllegalArgumentException("Unknown Uri: " + uri);
     }
}
```

Este fragmento de código pode parecer intimidador, mas consiste basicamente em constantes com um método útil (`getType( )`). Eis como as linhas numeradas funcionam:

→ 2 A autoridade para `ContentProvider` — por convenção, igual ao nome totalmente qualificado da classe. Este valor deve corresponder ao valor adicionado ao arquivo `AndroidManifest.xml` para as autoridades do provedor.

→ 4 A URL de base (ou URI) para o `ContentProvider`. Sempre que seu aplicativo solicita dados para essa URL, o Android roteia a solicitação para `ContentProvider`.

`ContentProvider` suporta dois tipos de URLs: uma para listar todos os lembretes e outra para listar um lembrete específico.

O primeiro tipo de URL é `CONTENT_URI` e o segundo é `CONTENT_URI` com o ID do lembrete anexado ao final.

→ 8 Como `ContentProvider` suporta dois tipos de dados, define dois tipos (ou tipos MINE) para esses dados. Os *tipos MIME* são apenas as strings comumente usadas na Web para identificar os tipos de dados. Por exemplo, o conteúdo HTML da web geralmente tem um tipo MIME `text/html` e os arquivos MP3 de áudio têm `audio/mpeg3`. Como os lembretes são de um tipo padrão desconhecido, você pode criar strings do tipo MIME, contanto que siga as convenções do Android e do MIME.

O tipo MIME List começa com `ContentResolver.CURSOR_DIR_BASE_TYPE` e o tipo MIME Reminder individual começa com `ContentResolver.CURSOR_ITEM_BASE_TYPE. DIR` representa a lista e `ITEM` representa o item — bem simples.

Esta linha também verifica para saber se o subtipo (que segue /) começa com `vnd`. O subtipo é seguido do nome totalmente qualificado da classe e do tipo de dado — neste caso, `com.dummies.android.taskreminder` e `reminder`. Visite `http://developer.android.com/reference/android/content/ContentResolver.html` (conteúdo em inglês) para obter mais informações sobre as convenções Android para os tipos MIME.

→ 13 Usa outra constante para identificar os tipos de lista versus os tipos de item, que são `ints`.

→ **15** `UriMatcher` é usado para determinar o tipo de URL: lista ou item. Você constrói um `UriMatcher` usando o método chamado `buildUriMatcher()` na linha 21.

→ **22** Cria `UriMatcher`, que pode indicar se certa URL é do tipo lista ou do tipo item. O parâmetro `UriMatcher.NO_MATCH` informa ao aplicativo qual valor padrão retornar para uma correspondência.

→ **23** Define o tipo de lista. Qualquer URL que usa a autoridade `com.dummies.android.taskreminder.ReminderProvider` e tem um caminho denominado `"reminder"` retorna o valor `LIST_REMINDER`.

→ **24** Define o tipo de item. Qualquer URL que usa a autoridade `com.dummies.android.taskreminder.ReminderProvider` e tem um caminho parecido com `reminder/#` (onde # é um número) retorna o valor `ITEM_REMINDER`.

→ **33** Usa `UriMatcher` na linha 15 para determinar qual tipo MIME retornar. Se a URL for uma URL de lista, retornará `REMINDERS_MIME_TYPE`. Se for uma URL de item, retornará `REMINDER_MIME_TYPE`.

Criando e Editando Tarefas com o SQLite

Depois de ter um `ContentProvider`, você pode criar uma tarefa para ele: Insira um registro, e então liste todas as tarefas no `ReminderListFragment`. O usuário poderá, então, tocar em uma tarefa para editá-la ou pressioná-la por um tempo para apagá-la. Estas interações do usuário cobrem as operações para criar, ler, atualizar e excluir (CRUD) necessárias para fazer o aplicativo Task Reminder funcionar.

Inserindo uma entrada de tarefa

Inserir tarefas é simples, depois de você pegar o jeito. Para inserir sua primeira tarefa no banco de dados SQLite, construa o listener de clique do botão Save (Salvar) para:

1. Recuperar os valores das exibições `EditText`.
2. Armazenar os valores no banco de dados `ReminderProvider` usando um `ContentResolver`.
3. Atualizar a interface do usuário exibindo um toast e fechando a atividade de edição.

Depois de inserir sua primeira tarefa, você deve ter compreendido bem a interação da classe `ReminderProvider` para executar mais tarefas. As próximas seções irão apresentar a implementação completa de `ReminderProvider`, que descreve as operações CRUD.

Salvando os valores da tela no banco de dados

Quando o usuário cria uma tarefa, a operação ocorre em `onClickListener` do `mConfirmButton` de `ReminderEditFragment`. Lá, o aplicativo responde ao clique do botão Save (Salvar) do usuário. Se `mRowId` para o fragmento for 0, o usuário deseja adicionar uma tarefa. Se `mRowId` for maior que 0, o usuário deseja editar uma tarefa existente. Primeiro, você configura alguns parâmetros, solicita um `ContentResolver` para completar uma criação ou atualização, e então processa o resultado e notifica o usuário.

Adicione o seguinte elemento `OnClickListener` a `mConfirmButton` em `ReminderEditFragment.onCreateView()`:

```
mConfirmButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        ContentValues values = new ContentValues();                 →4
        values.put(ReminderProvider.COLUMN_ROWID, mRowId);          →5
        values.put(ReminderProvider.COLUMN_TITLE, mTitleText.
    getText().toString());                                          →7
        values.put(ReminderProvider.COLUMN_BODY, mBodyText.
    getText().toString());
        values.put(ReminderProvider.COLUMN_DATE_TIME,
    mCalendar.getTimeInMillis());                                   →11

        if (mRowId == 0) {                                          →13
            Uri itemUri = getActivity().getContentResolver().
    insert(ReminderProvider.CONTENT_URI, values);                   →15
            mRowId = ContentUris.parseId(itemUri);                  →16
        } else {
            int count = getActivity().getContentResolver().update(
    ContentUris.withAppendedId(
            ReminderProvider.CONTENT_URI, mRowId),
    values, null, null);                                            →21
            if (count != 1)                                         →22
    throw new IllegalStateException("Unable to update "
        + mRowId);
        }

        Toast.makeText(getActivity(),
    getString(R.string.task_saved_message),
    Toast.LENGTH_SHORT).show();                                     →29
        getActivity().finish ();                                    →30
}
    });
```

Eis como o código funciona:

→ 4 Uma consulta é executada por `ContentResolver` que, finalmente, é atendido por `ReminderProvider`. Especificamente, esta linha insere ou atualiza um lembrete. O objeto `ContentValues` indica o registro a atualizar e os dados a usar.

→ 5 O ID da linha para o lembrete — `0` se o usuário está criando um novo registro ou o ID específico da tarefa se o usuário está atualizando uma tarefa existente.

→ 7 O título da tarefa, que é o valor indicado pela entrada do usuário na exibição `EditText mTitleText`. O aplicativo chama

`mTitleText.getText()` para obter um `CharSequence`, então, chama `toString()` para produzir a string.

→ **11** Mantém a data e a hora que o usuário selecionou para o lembrete. Também chama `Calendar.getTimeInMillis()` e coloca o resultado em um objeto `ContentValues` (SQLite não representa datas diretamente, portanto, você deve converter o objeto para um `long`).

→ **13** Verifica se o usuário está adicionando uma nova tarefa ou atualizando uma existente. Se `mRowID` for `0`, o usuário está adicionando uma nova tarefa. Do contrário, o usuário está atualizando uma tarefa existente.

→ **15** Chama `getContentResolver()` na atividade, e então chama `insert()` para executar a consulta de inserção. Uma inserção requer dois parâmetros: os dados que você deseja inserir e a URI da tabela na qual você está inserindo-os. Em resposta, `insert()` fornece a URI completa dos dados inseridos.

→ **16** Obtém o ID da URI do item que foi inserido, usando `ContentUris.parseId()` e define `mRowID` para esse valor.

→ **21** Chama `update()` em `ContentResolver` para executar uma atualização. Ao invés de transmitir a URI de base, transmite a URI do item sendo atualizado, que é obtida chamando `ContentUris.withAppendedId()` e transmitindo a URI de base e o ID do lembrete. Os dois últimos parâmetros nulos indicam que nenhuma consulta SQL especial precisa atualizar mais do que uma única linha de uma vez.

→ **22** Indica quantos registros foram atualizados. Nunca deve ser diferente de 1, portanto, se nenhuma linha ou diversas linhas forem atualizadas, uma mensagem de erro será gerada.

→ **29** Chama `Toast.makeText()` para exibir uma mensagem de sucesso por um curto período de tempo. Então, chama `show()` para que a mensagem apareça na tela.

→ **30** Chama `finish()` na atividade para fechar a atividade de edição e mostrar a lista de atividades novamente.

A implementação ReminderProvider completa

Algumas vezes, ver todos os elementos de uma vez é melhor do que vê-los aos poucos. Trabalhar com o SQLite na classe `ReminderProvider` não é diferente. A Listagem 12-3 mostra a implementação completa de `ReminderProvider` para que você possa ter uma ideia do trabalho sendo realizado.

Listagem 12-3: A implementação completa de ReminderProvider

```
package com.dummies.android.taskreminder;

import android.content.ContentProvider;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.net.Uri;

public class ReminderProvider extends ContentProvider {
    // Uri e autoridade do provedor de conteúdo
    public static String AUTHORITY = "com.dummies.android.taskreminder.
        ReminderProvider";
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
       + "/reminder");

    // Tipos MIME usados para pesquisar palavras ou procurar uma única definição
    public static final String REMINDERS_MIME_TYPE = ContentResolver.CURSOR_DIR_
        BASE_TYPE
       + "/vnd.com.dummies.android.taskreminder.reminder";
    public static final String REMINDER_MIME_TYPE = ContentResolver.CURSOR_ITEM_
        BASE_TYPE
       + "/vnd.com.dummies.android.taskreminder.reminder";

    // Colunas do banco de dados
    public static final String COLUMN_ROWID = "_id";
    public static final String COLUMN_DATE_TIME = "reminder_date_time";
    public static final String COLUMN_BODY = "body";
    public static final String COLUMN_TITLE = "title";

    // Constantes relacionadas ao banco de dados
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "data";
    private static final String DATABASE_TABLE = "reminders";

    private static final String DATABASE_CREATE = "create table "
           + DATABASE_TABLE + " (" + COLUMN_ROWID
           + " integer primary key autoincrement, " + COLUMN_TITLE
           + " text not null, " + COLUMN_BODY + " text not null, "
           + COLUMN_DATE_TIME + " integer not null);";

    // Coisas do UriMatcher
    private static final int LIST_REMINDER = 0;
    private static final int ITEM_REMINDER = 1;
    private static final UriMatcher sURIMatcher = buildUriMatcher();
```

(continua)

Listagem 12-3: *(continuação)*

```
private SQLiteDatabase mDb;

/**
* Constrói um UriMatcher para pesquisar a sugestão e atualizar com rapidez
* as consultas.
*/
private static UriMatcher buildUriMatcher() {
    UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
    // para obter definições...
    matcher.addURI(AUTHORITY, "reminder", LIST_REMINDER);
    matcher.addURI(AUTHORITY, "reminder/#", ITEM_REMINDER);

    return matcher;
}

@Override
public boolean onCreate() {
    mDb = new DatabaseHelper(getContext()).getWritableDatabase();
    return true;
}

@Override
public Cursor query(Uri uri, String[] ignored1, String ignored2,
   String[] ignored3, String ignored4) {                                  →75

    String[] projection = new String[] { ReminderProvider.COLUMN_ROWID,
           ReminderProvider.COLUMN_TITLE, ReminderProvider.COLUMN_BODY,
           ReminderProvider.COLUMN_DATE_TIME };                           →79

   // Use UriMatcher para ver o tipo de consulta e o formatar apropriadamente
    // a consulta de banco de dados
    Cursor c;
    switch (sURIMatcher.match(uri)) {                                     →84
    case LIST_REMINDER:                                                   →85
        c = mDb.query(ReminderProvider.DATABASE_TABLE, projection, null,
      null, null, null, null);                                            →87
        break;
    case ITEM_REMINDER:                                                   →89
        c = mDb.query(ReminderProvider.DATABASE_TABLE, projection,        →90
     ReminderProvider.COLUMN_ROWID + "=?", new String[] { Long
         .toString(ContentUris.parseId(uri)) },                           →92
      null, null, null, null);
        if (c != null && c.getCount() > 0) {                              →94
            c.moveToFirst();                                              →95
        }
        break;
    default:                                                              →98
        throw new IllegalArgumentException("Unknown Uri: " + uri);
    }
```

(continua)

```
    c.setNotificationUri(getContext().getContentResolver(), uri);      →102
    return c;                                                          →103
}

@Override
public Uri insert(Uri uri, ContentValues values) {                     →107
    values.remove(ReminderProvider.COLUMN_ROWID);                      →108
    long id = mDb.insertOrThrow(ReminderProvider.DATABASE_TABLE, null, →109
     values);
    getContext().getContentResolver().notifyChange(uri, null);         →111
    return ContentUris.withAppendedId(uri, id);                        →112
}

@Override
public int delete(Uri uri, String ignored1, String[] ignored2) {       →116
    int count = mDb.delete(ReminderProvider.DATABASE_TABLE,
     ReminderProvider.COLUMN_ROWID + "=?",
    new String[] { Long.toString(ContentUris.parseId(uri)) });         →119
    if (count > 0)                                                     →120
       getContext().getContentResolver().notifyChange(uri, null);
    return count;                                                      →122
}

@Override
public int update(Uri uri, ContentValues values, String ignored1,
       String[] ignored2) {                                            →127
    int count = mDb.update(ReminderProvider.DATABASE_TABLE, values,
     COLUMN_ROWID + "=?",
    new String[] { Long.toString(ContentUris.parseId(uri)) });         →130
    if( count>0 )
       getContext().getContentResolver().notifyChange(uri, null);      →132
    return count;                                                      →133
}

/**
* Este método é necessário para consultar os tipos suportados. Também
* é útil em nosso próprio método query( ) para determinar o tipo de Uri recebido.
*/
@Override
public String getType(Uri uri) {
    switch (sURIMatcher.match(uri)) {
    case LIST_REMINDER:
       return REMINDERS_MIME_TYPE;
    case ITEM_REMINDER:
       return REMINDER_MIME_TYPE;
    default:
       throw new IllegalArgumentException("Unknown Uri: " + uri);
    }
```

(continua)

Listagem 12-3: *(continuação)*

```
        }

        // O código DatabaseHelper do SQLite foi omitido para abreviar.
 }
```

Eis como o código funciona:

→ **75** O método `query()` recupera um lembrete ou uma lista de lembretes. Sua URI pode ser do tipo lista ou do tipo item.

→ **79** Cria uma variável chamada `projection` que informa às várias consultas quais colunas de dados o usuário deseja recuperar. As várias consultas incluem `ID`, `title`, `body`, `date` e `time`.

→ **84** Chama `URIMatcher.match()` para determinar se é uma URI de lista ou de item.

→ **85-87** Para uma URI de lista, esta linha chama `query()` no SQLiteDatabase para recuperar todas as colunas de todos os lembretes.

O uso do método `query()` de SQLiteDatabase e seus parâmetros é explicado em detalhes na seção "Compreendendo a operação de consulta (leitura)", posteriormente neste capítulo.

→ **89-90** Para uma URI de item, esta linha chama `query()` no SQLiteDatabase para recuperar todas as colunas do lembrete especificado. Especifica a tabela do banco de dados e as colunas que devem ser recuperadas.

→ **92** O primeiro parâmetro é `COLUMN_ROWID=? String`, onde o valor de ? é fornecido no próximo parâmetro. O próximo parâmetro é um array de strings, onde cada uma mapeia um ponto de interrogação. Esse parâmetro tem um ponto de interrogação, portanto, precisa de uma string no array, que é o ID da tarefa (da URI).

→ **94-95** Se a chamada para a consulta for bem-sucedida e retornou pelo menos um valor, irá para o primeiro valor. O método `moveToFirst()` no objeto `Cursor` direciona o cursor para o primeiro registro no conjunto de resultados. Esse método será chamado apenas se `Cursor` não for nulo. O motivo do cursor não ser posicionado imediatamente no primeiro registro é que é um conjunto de resultados. Antes que o aplicativo possa trabalhar com o registro, ele deve direcionar para ele. Considere o conjunto de resultados como uma caixa de itens: você não pode trabalhar com um item até que ele saia da caixa.

→ **98** Exibe uma mensagem de erro se a URI não for uma URI de lista nem uma URI de item.

→ **102** Chama `setNotificationUri()` no objeto `Cursor` para associar o resultado da consulta à URI. É transmitido para `ContentResolver` e para a URI que foi atualizada.

Um `ContentProvider` pode notificar os usuários sobre as alterações em seus dados. Por exemplo, se um fragmento abriu uma exibição de lista para seus lembretes e outro fragmento alterou e salvou um lembrete específico, a exibição de lista será notificada de que seu conteúdo mudou e irá atualizá-lo automaticamente.

→ **103** Retorna o objeto `Cursor` para que ele possa ser usado pelo fragmento para processar os resultados.

→ **107** O método que é chamado quando um usuário insere um lembrete no banco de dados.

→ **108** Remove o parâmetro ID se ele existir, pois você não pode especificar um ID quando insere lembretes. O banco de dados fornece o próximo ID disponível.

→ **109** Insere o lembrete no banco de dados usando qualquer valor transmitido.

O uso do método `insertOrThrow()` e seus parâmetros serão explicados em detalhes na seção "Compreendendo a operação de inserção", posteriormente neste capítulo.

→ **111** Chama `notifyChange()` em `ContentResolver` para essa URI; corresponde à chamada `setNotificationUri()` da linha 102. Quando você modifica a tabela inserindo um item, notifica qualquer pessoa que tem uma consulta aberta nessa tabela informando que a consulta pode ter mudado.

→ **112** Constrói a URI do item final para este lembrete anexando à URI de base o ID que foi criado.

→ **116** O método `delete` de `ContentProvider`.

→ **119** Usando o método `ContentUris.parseId()` para recuperar o id da tarefa a partir da URI, chama o método `delete()` no banco de dados SQLite para apagar uma tarefa do banco de dados.

O uso e os parâmetros do método `delete()` serão descritos em detalhes na seção "Compreendendo a operação de exclusão", posteriormente neste capítulo.

→ **120** Se o banco de dados apagou algo, notifica a todos que ele foi atualizado para que possam atualizar suas consultas.

→ **122** Diferentemente de `insert()`, `delete()` retorna o número de linhas que foram apagadas.

→ **127** O método `update()` é parecido com o método `delete()`.

→ **130** O método `update()` é responsável por atualizar uma tarefa existente com novas informações.

O uso do método `update()` e seus parâmetros serão explicados em detalhes na seção "Compreendendo a operação de atualização", posteriormente neste capítulo.

→ **132** Se o banco de dados atualizou algo, notifica a todos que ele foi atualizado para que possam atualizar suas consultas.

→ **133** Retorna a quantidade de linhas que foram modificadas.

Uma rotina CRUD aceita vários parâmetros, que serão explicados em detalhes nas seções "Compreendendo a operação de inserção", "Compreendendo a operação de consulta (leitura)", "Compreendendo a operação de atualização " e "Compreendendo a operação de exclusão".

Compreendendo a operação de inserção

A operação de inserção é simples de completar porque você está simplesmente inserindo um valor no banco de dados. Você chama `insertOrThrow()`, ao invés de `insert()` porque deseja obter uma exceção, caso um problema ocorra ao inserir no banco de dados. O método `insertOrThrow()` aceita estes parâmetros:

- `table`: O nome da tabela que receberá os dados. Usa a constante `DATABASE_TABLE` para o valor.
- `nullColumnsHack`: O SQL não permite inserir uma linha vazia, portanto, se o parâmetro `ContentValues` (o próximo parâmetro) estiver vazio, o valor `NULL` será explicitamente atribuído a esta coluna. Está transmitindo `null` para esse valor.
- `values`: Este parâmetro define os valores iniciais, como definidos no objeto `ContentValues`. Está fornecendo a variável local `initialValues` como o valor deste parâmetro. Essa variável contém as informações do par de chave-valor para definir uma nova linha.

Compreendendo a operação de consulta (leitura)

A operação de consulta também é conhecida como operação de leitura porque na maioria das vezes, o aplicativo está *lendo* os dados no banco de dados com o método `query()`. O método de consulta é responsável por fornecer um conjunto de resultados com base em uma lista de critérios fornecidos. Esse método retorna um Cursor que fornece um acesso aleatório de leitura-gravação ao conjunto de resultados retornado pela consulta.

O método de consulta aceita estes parâmetros:

- `table`: O nome da tabela do banco de dados que será consultada. O valor vem da constante `DATABASE_TABLE`.
- `columns`: Uma lista de colunas retornadas pela consulta. Transmitir `null` retorna todas as colunas, o que é normalmente desencorajado para evitar ler e retornar dados desnecessários. Se você precisar de todas as colunas, será válido transmitir `null`. Para este aplicativo, você pode transmitir um array de string com as colunas a retornar.

- `selection`: Um filtro descrevendo quais linhas retornar, formatado como uma cláusula `WHERE` do SQL (excluindo o próprio `WHERE`). Transmitir um `null` retorna todas as linhas na tabela. Se for uma operação de lista, você fornecerá `null` porque deseja todas as linhas retornadas. Se for uma operação get para retornar um único lembrete, você fornecerá uma string de consulta `COLUMNS_ROWID=?`. O SQL sabe que um ponto de interrogação indica um valor específico de `COLUMN_ROWID` no parâmetro `selectionArgs`.
- `selectionArgs`: É permitido incluir pontos de interrogação (?) na string de seleção. Esses pontos são substituídos pelos valores de `selectionArgs` para que apareçam na seleção. Esses valores são passados no tipo string. Dependendo da situação, você transmite `null` ou um array `String` contendo o ID a ser obtido.
- `groupBy`: Um filtro que mostra apenas as linhas formatadas como uma cláusula SQL `GROUP BY` (excluindo `GROUP BY`). Transmitir `null` faz com que as linhas não sejam agrupadas. Um valor `null` é transmitido aqui porque não é necessário agrupar os resultados.
- `having`: Um filtro que mostra apenas os grupos de linha que devem ser incluídos no cursor, caso o grupo de linhas esteja sendo usado. Transmitir `null` faz com que todos os grupos de linha sejam incluídos e é obrigatório quando o grupo de linhas não está sendo usado. É por isso que um valor `null` é requerido aqui.
- `orderBy`: A ordem das linhas, formatadas como uma cláusula SQL `ORDER BY` (excluindo o próprio `ORDER BY`). Transmitir null usa a ordem de classificação padrão, que pode ser desordenada. Você pode transmitir um valor `null` porque a ordem na qual os resultados são retornados não é importante.
- `limit`: Limita o número de linhas retornadas pela consulta usando uma cláusula `LIMIT`. Transmitir valores `null` determina que você não tem uma cláusula `LIMIT`. Para evitar limitar o número de linhas retornadas, você pode transmitir `null` para retornar todas as linhas que correspondem à sua consulta.

Compreendendo a operação de atualização

Atualizar um registro em um banco de dados simplesmente substitui os valores na célula de destino que está dentro da linha especificada pelos parâmetros de entrada (ou nas linhas, se muitas linhas são atualizadas). Como na operação de exclusão seguinte, a atualização pode afetar muitas linhas. Você deve entender os parâmetros do método de atualização e como eles podem afetar os registros no banco de dados. O método `update()` aceita estes parâmetros:

- `table`: A tabela a atualizar. O valor é fornecido pela constante `DATABASE_TABLE`.

- `values`: O objeto `ContentValues`, que contém os campos que devem ser atualizados.
- `whereClause`: A cláusula `WHERE`, que limita quais linhas devem ser atualizadas. Você informa ao banco de dados para atualizar uma linha com um ID específico fornecendo o valor de string `COLUMN_ROWID + "=?"`. O ID é especificado em `whereArgs`.
- `whereArgs`: Os argumentos `whereClause` adicionais. Você fornece o ID que está ligado a `?` no argumento `whereClause`. Neste caso, `whereArgs` é um array `String` contendo o ID do lembrete a ser atualizado. Sempre deve haver exatamente um valor no array para cada `?` em `whereClause`.

Compreendendo a operação de exclusão

Ao usar o método `delete()`, vários parâmetros são utilizados para definir o critério de exclusão no banco de dados. Uma instrução de exclusão pode afetar nenhum registro no banco de dados ou todos eles. Você deve entender os parâmetros da chamada de exclusão para assegurar que não apagará os dados por engano. Os parâmetros para o método `delete()` são

- `table`: A tabela que terá linhas apagadas. O valor deste parâmetro é fornecido pela constante `DATABASE_TABLE`.
- `whereClause`: A cláusula `WHERE` opcional aplicada ao apagar as linhas. Se você transmitir `null`, todas as linhas serão apagadas. Esse valor é fornecido manualmente criando a cláusula `WHERE` com a string `COLUMN_ROWID + "=?"`.
- `whereArgs`: Os argumentos da cláusula `WHERE` opcional. Você fornece o ID que está ligado a `?` no argumento `whereClause`. Neste caso, `whereArgs` é um array String contendo o ID do lembrete a ser apagado. Sempre deve haver exatamente um valor no array para cada `?` em `whereClause`.

Carregadores

Quando você está fazendo qualquer tipo de operação de E/S, tal como, ler de uma rede ou de um disco (ler um banco de dados, por exemplo), deve fazer este trabalho a partir de um thread em segundo plano. Se você trabalhar a partir do thread principal da interface do usuário, correrá o risco de bloqueá-lo por um período de tempo desconhecido, podendo fazer com que pareça tolo e não responsivo. Em circunstâncias particularmente ruins, pode até levar a exibir a temida caixa de diálogo Aplicativo Não Responsivo, que pode deixar muitos usuários acreditando que seu aplicativo paralisou.

O *carregador* foi introduzido no Android 3.x para ajudar a resolver este problema — ele fornece um mecanismo pelo qual você pode inicializar as tarefas em segundo plano (tais como ler a partir de seu banco de dados),

e então obter um callback quando essas tarefas terminam, para que possa atualizar a interface do usuário.

Um exemplo típico de carregador é `CursorLoader`. Você usa um `CursorLoader` para carregar os dados de um banco de dados SQLite usando um cursor. Para adicionar um `CursorLoader` a um de seus fragmentos de lista, você implementa a interface `LoaderCallback` em seu callback e implementa três métodos `LoaderCallback`:

- `onCreateLoader()`: Este método é chamado em um thread em segundo plano quando você cria um carregador usando `initLoader()`. Nesse método, você é responsável por criar um objeto `CursorLoader` e retorná-lo. `CursorLoader` usa uma URI para solicitar dados a `ContentProvider`.
- `OnLoadFinished()`: Este método é chamado quando o objeto `CursorLoader` termina de carregar seus dados do banco de dados. Nesse método, você é responsável por atualizar a IU para mostrar novos dados para o usuário.
- `OnLoaderReset()`: Este método é chamado quando o carregador está sendo redefinido ou finalizado. Quando isto acontece, você é responsável por assegurar que seu fragmento não usará mais o carregador ou seu cursor.

Para iniciar um carregador, primeiro você obtém um `LoaderManager` a partir de sua atividade chamando `getLoaderManager()`, então, `initLoader()`. `initLoader()` começa a carregar os dados em segundo plano chamando `onCreateLoader()` e quando termina, executa `onLoaderFinished()` em seu objeto `LoaderCallback`.

Você pode usar carregadores para outras tarefas além de carregar dados a partir de um banco de dados, mas todos os carregadores devem implementar os mesmos três métodos, independentemente de estarem carregando seus dados de um banco de dados, rede ou algum outro lugar.

Visite `http://developer.android.com/guide/components / loaders.html` (conteúdo em inglês) para obter informações sobre os carregadores.

Retornando todas as tarefas com um cursor

Você pode criar uma tarefa, mas para que serve isso se não puder ver a tarefa na lista de tarefas? Para nada, realmente. Você tem que listar as tarefas que existem atualmente no banco de dados na `ListView` em `ReminderListFragment`.

A Listagem 12-4 descreve o `ReminderListFragment` inteiro com o novo código que pode ler a lista de tarefas a partir do banco dados e colocá-la em `ListView`.

Listagem 12-4: ReminderListFragment completa com acesso ao banco de dados

```
package com.dummies.android.taskreminder;

import android.database.Cursor;
import android.os.Bundle;
import android.support.v4.app.ListFragment;
import android.support.v4.app.LoaderManager.LoaderCallbacks;
import android.support.v4.content.CursorLoader;
import android.support.v4.content.Loader;
import android.support.v4.widget.SimpleCursorAdapter;
import android.view.View;
import android.widget.ListView;

import com.dummies.android.taskreminder.R.string;

public class ReminderListFragment extends ListFragment implements
      LoaderCallbacks<Cursor> {                                          →16

   private SimpleCursorAdapter mAdapter;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      // Crie um array para especificar os campos a serem exibidos na lista
            // (apenas TITLE)
      String[] from = new String[] { ReminderProvider.COLUMN_TITLE };  →26

      // e um array dos campos aos quais vincular esses campos (neste
      // caso, apenas text1)
      int[] to = new int[] { R.id.text1 };                             →30

      // Agora, crie um adaptador de curso simples e defina-o para exibição
      mAdapter = new SimpleCursorAdapter(getActivity(),
            R.layout.reminder_row, null, from, to, 0);                 →34
      setListAdapter(mAdapter);                                        →35

      getLoaderManager().initLoader(0, null, this);                    →37
   }

   @Override
   public void onViewCreated(View view, Bundle savedInstanceState) {
      super.onViewCreated(view, savedInstanceState);
      setEmptyText(getResources().getString(string.no_reminders));
      registerForContextMenu(getListView());
      setHasOptionsMenu(true);
   }

   @Override
   public void onListItemClick(ListView l, View v, int position, long id) {
      super.onListItemClick(l, v, position, id);
      startActivity(new Intent(getActivity(), ReminderEditActivity.class) →51
            .putExtra(ReminderProvider.COLUMN_ROWID, id));
   }

   @Override
   public boolean onContextItemSelected(MenuItem item) {               →56
      switch (item.getItemId()) {
```

(continua)

```
        case R.id.menu_delete:
            AdapterContextMenuInfo info = (AdapterContextMenuInfo) item
           .getMenuInfo();                                                  →60
            getActivity().getContentResolver().delete(
           ContentUris.withAppendedId(ReminderProvider.CONTENT_URI,
               info.id), null, null);                                       →63
            return true;
        }
        return super.onContextItemSelected(item);
    }

    @Override
    public Loader<Cursor> onCreateLoader(int ignored, final Bundle args) {  →70
        return new CursorLoader(getActivity(), ReminderProvider.CONTENT_URI,
         null, null, null, null);                                           →71
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {      →75
        mAdapter.swapCursor(cursor);
    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {                      →80
       // Isto é chamado quando o último Cursos fornecido para onLoadFinished()
        // acima está para ser fechado. Certifique-se de que o adaptador
        // não esteja mais usando o cursor definindo-o para null
        mAdapter.swapCursor(null);
    }

    // Código do menu removido para simplificação.
}
```

Esta lista descreve as linhas numeradas do código que lê a lista de tarefas:

→ 16 Adiciona a interface `LoaderCallbacks` à lista de interfaces.

Para esta linha funcionar, você implementa alguns callbacks relacionados ao carregador posteriormente na classe para que possa lidar com o carregamento dos dados em um thread assíncrono em segundo plano. Veja a seção "Carregador" anteriormente para obter detalhes.

→ 26 Vê o campo de título do lembrete.

→ 30 Define o array de exibições a vincular como a exibição da linha. Então, o título corresponde a um ID da tarefa em particular, sendo por isso que a variável na linha 26 é nomeada como `from` e a variável nesta linha é nomeada como `to`. Os valores da linha 26 mapeiam os valores na linha 30.

→ 34 Cria um `SimpleCursorAdapter` que mapeia as colunas de um Cursor para `TextView`, como definido em um arquivo XML de layout. Usando esse método, você pode especificar quais colunas exibir e especificar o arquivo XML que define a aparência dessas exibições. Inicialmente, `SimpleCursorAdapter` está vazio porque não tem dados ainda.

O uso de um `SimpleCursorAdapter` e seus parâmetros associados é descrito na seção a seguir.

→ **35** Transmite `SimpleCursorAdapter` como o parâmetro do adaptador para o método `setListAdapter()` para informar à exibição de lista onde encontrar seus dados.

→ **37** Pede a `LoaderManager` para começar a carregar o carregador com um ID `0`. Quando o carregador é terminado, ele executa os métodos de callback na classe `LoaderCallback`, que você transmite como o parâmetro final desta chamada (neste caso, `"this"`).

→ **51** Coloca na intenção o ID da tarefa a ser editada. `ReminderEditActivity` examina essa intenção e se encontrar o ID, tenta permitir que o usuário edite a tarefa.

→ **56** Define o método que lida com os eventos do menu contextual do usuário, que ocorrem quando ele pressiona pela primeira vez a tarefa (pressionamento longo) na exibição de lista, e então seleciona um item de menu no menu contextual.

→ **60** Usa o método `getMenuInfo()` do item que foi clicado para obter uma instância de `AdapterContextMenuInfo`. Essa classe exibe pequenas porções de informação sobre o item de menu e o item no qual o usuário pressionou por algum tempo na exibição de lista.

→ **63** Obtém um `ContextResolver` e solicita que apague a tarefa cujo ID é recuperado no campo Id do objeto `AdapterContextMenuInfo`. Esse campo contém o ID da linha na exibição de lista, que é usada para construir a URI da linha usando `ContentUris.withAppendedId()`. Quando o lembrete é apagado, `ListView` atualiza-se automaticamente para mostrar os últimos dados.

→ **70** Quando você chama `initLoader()` na linha 37, o método `onCreateLoader()` é chamado. Aqui, você cria o carregador que o Android começa a executar em um thread em segundo plano. Os fragmentos complicados podem ter diversos carregadores de diferentes tipos, mas você pode usar um único carregador neste fragmento simples e ignorar com segurança o parâmetro ID do método `onCreateLoader()`.

→ **71** Cria um novo `CursorLoader`, que é usado quando você carrega itens de um `ContentProvider` ou `SQLiteDatabase`. `CursorLoader` carrega os dados usando `ReminderProvider`, portanto, especifique o `CONTENT_URI` de `ReminderProvider`. É uma operação de lista simples, portanto, você pode ignorar a projeção opcional, seleção e os parâmetros `selectionArgs` e `sortOrder` para o construtor `CursorLoader`.

→ **75** Quando um carregador termina, chama o callback `onLoadFinished()` e retorna o cursor que contém os dados carregados. Então, o adaptador troca o antigo cursor pelo novo e atualiza automaticamente `ListView`.

→ **80** Chama `onLoaderReset()` quando o último `Cursor` fornecido para `onLoadFinished()` está para ser fechado. O cursor do adaptador é definido para null porque não é mais necessário.

Compreendendo o SimpleCursorAdapter

A linha 34 na Listagem 12-4 cria um `SimpleCursorAdapter`. Ele faz muito por você quando você deseja vincular os dados de um objeto `Cursor` a uma exibição de lista. Para configurar um `SimpleCursorAdapter`, forneça estes parâmetros:

- `getActivity() - Context`: O contexto associado ao adaptador.
- `R.layout.reminder_row - layout id`: O identificador de recurso do layout que define o arquivo a usar para este item de lista.
- `null - Cursor`: O cursor do banco de dados que o adaptador usa para carregar os dados. Você não tem nenhum cursor ainda, portanto, use `null`.
- `from - from`: Um array de nomes de coluna que é usado para vincular os dados do cursor à exibição; definido na linha 26.
- `to - to`: Um array de IDs de exibição que devem exibir as informações da coluna a partir do parâmetro `from`. O campo To (Para) é definido na linha 30.
- `0 - flags`: Qualquer flag opcional a ser transmitida para `SimpleCursorAdapter`. Você não precisa de flags opcionais, portanto, use 0.

Os parâmetros `to` e `from` criam um mapa informando a `SimpleCursor Adapter` como mapear os dados no cursor para as exibições no layout de linha.

Quando você iniciar o aplicativo agora, verá uma lista de itens que estão sendo lidos a partir do banco de dados SQLite. Se você não vir essa lista, crie uma pressionando no menu e selecionando o item para adicionar uma nova tarefa.

Apagando uma tarefa

Para o usuário, apagar uma tarefa é tão simples quanto pressionar por algum tempo um item em `ReminderListFragment` e selecionar a ação de exclusão. Para realmente apagar a tarefa do banco de dados, você usa o método `delete()` no objeto do banco de dados SQLite. Esse método é chamado na Listagem 12-3 na linha 119.

O método `delete()` de `ContentResolver` é chamado de dentro da chamada do método `onContextSelectedItem()` na linha 63 da Listagem 12-4. O único item que é necessário antes de apagar a tarefa do banco de dados é o ID da tarefa no banco. Para obter o ID, você deve usar o objeto

`AdapterContextMenuInfo`, que fornece informações de menu extras. Essas informações são fornecidas para a seleção do menu contextual quando um menu é aberto para `ListView`. Como você está carregando a lista com um cursor do banco de dados, `ListView` contém o ID que está procurando. A linha 60 da Listagem 12-4 obtém o objeto `AdapterContextMenuInfo` e a linha 63 chama o método `delete()` com o ID como um parâmetro.

Agora, quando você iniciar o aplicativo no emulador, poderá criar, ler, atualizar e apagar as tarefas!

Capítulo 13

Alertando o Usuário com AlarmManager

Neste Capítulo

- Compreendendo as tarefas agendadas
- Planejando as permissões
- Configurando alarmes
- Vendo como as reinicializações do dispositivo afetam os alarmes

Muitas tarefas precisam ocorrer diariamente, certo? Acordar, tomar banho, tomar o café da manhã — fazemos todas essas coisas todos os dias. Essas tarefas compõem a rotina matinal pré-trabalho de segunda a sexta (ou uma variante dela). Você pode ter um relógio interno e acordar todo dia na hora, mas a maioria das pessoas tem que usar alarmes para isso. No trabalho, os funcionários têm calendários que os lembram dos eventos futuros aos quais eles precisam ir, tais como, reuniões e atualizações importantes do servidor. Os lembretes e os alarmes fazem parte da maioria das rotinas diárias e as pessoas contam com eles de uma maneira ou de outra.

Construir seu próprio sistema de tarefas agendadas seria difícil. Felizmente, o Windows tem tarefas agendadas, o Linux tem cron e o Android tem a classe `AlarmManager`. Embora o Android seja baseado no Linux, ele não tem acesso a cron, portanto, você tem que configurar as ações agendadas via `AlarmManager` do Android.

Vendo Por Que Você Precisa do AlarmManager

O aplicativo Task Reminder tem uma palavra-chave em seu nome — *Reminder*. O usuário pode definir o título da tarefa, a descrição e a data e hora de lembrete para ser lembrado sobre uma tarefa, como neste exemplo: Um usuário adiciona algumas tarefas no aplicativo Task Reminder (todas para

hoje mais tarde), afasta seu dispositivo e sai para resolver seus assuntos. Se ele não for lembrado sobre as tarefas, poderá esquecê-las, portanto, ele precisa de um modo de ser lembrado sobre o que deve acontecer — é onde a classe `AlarmManager` entra em cena.

A classe `AlarmManager` permite que os usuários agendem a hora em que o aplicativo Task Reminder deve ser executado. Quando um alarme dispara, uma intenção é transmitida pelo sistema. Então, seu aplicativo responde a essa intenção de transmissão e executa uma ação, tal como abrir o aplicativo, notificar o usuário via notificação da barra de status (que você poderá fazer no Capítulo 14) ou realizar outro tipo de ação.

Pedindo a Permissão do Usuário

Você não deixaria seu vizinho de porta guardar as decorações de Natal em seu galpão sem permissão, deixaria? Provavelmente, não. O Android não é diferente. Realizar algumas ações no dispositivo Android de um usuário requer permissão, como explicado nas seguintes seções.

Vendo como as permissões afetam a experiência do usuário

Quando um usuário instala um aplicativo da Google Play Store, o arquivo manifest do aplicativo é examinado para obter as permissões requeridas. Sempre que seu aplicativo precisar acessar componentes diferenciados (tais como um armazenamento externo, Internet ou informações do dispositivo), o usuário é notificado e decide se continua ou não a instalação.

Não solicite permissões desnecessárias para seu aplicativo — os usuários paranoicos com segurança provavelmente irão rejeitá-las. Por exemplo, o aplicativo Silent Mode Toggle (descrito na Parte II) não precisa de localização por GPS, acesso à Internet ou informações relacionadas ao hardware.

Se seu aplicativo não precisar de uma permissão, retire-a. Quanto menos permissões seu aplicativo requerer, mais provavelmente o usuário irá instalá-lo.

Definindo as permissões solicitadas no arquivo AndroidManifest.xml

Quando você precisar solicitar permissões, adicione-as ao arquivo `AndroidManifest.xml` em seu projeto. Você precisa adicionar as seguintes permissões ao aplicativo Task Reminder:

- `android.permission.RECEIVE_BOOT_COMPLETED`: Permite que o aplicativo saiba quando o dispositivo reinicializa para que possa registrar de novo seus alarmes no Alarm Manager (Gerenciador de Alarmes).
- `android.permission.WAKE_LOCK`: Permite que o dispositivo permaneça ativo durante o processamento em segundo plano por períodos curtos de tempo.

Como `AlarmManager` mantém um relógio da CPU ativo, contanto que o método `onReceive()` do receptor de alarme esteja em execução, ele assegura que o dispositivo não entre no modo hibernação até que o aplicativo termine de trabalhar com a transmissão.

Se seu aplicativo precisasse de acesso à Internet ou precisasse gravar dados no cartão SD (o aplicativo Task Reminder não precisa), você adicionaria estas permissões respectivamente:

- **Internet:** `android.permission.INTERNET`
- **Cartão SD:** `android.permission.WRITE_EXTERNAL_STORAGE`

Há duas maneiras de adicionar permissões ao arquivo `AndroidManifest.xml`:

- **Editor de Permissões** `Android.Manifest.xml`: Escolha Add (Adicionar) → Use Permission (Usar Permissão), e então selecione a permissão na lista suspensa.
- **Arquivo XML:** Edite o arquivo manualmente para adicionar o elemento `uses-permission` a element do manifest. A solicitação de permissão XML é assim:

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Se você ainda não o fez, adicione as permissões `WAKE_LOCK` e `RECEIVE_BOOT_COMPLETED` ao aplicativo Task Reminder. Para ver uma lista completa das permissões disponíveis, examine a documentação de permissão Android em `http://d.android.com/reference/android/ Manifest.permission.html` (conteúdo em inglês).

Se você não declarar as permissões que seu aplicativo precisa, ele não funcionará como o esperado em um dispositivo ou um emulador, e qualquer exceção de execução gerada poderá paralisar seu aplicativo. Sempre assegure que suas permissões estejam presentes.

Ativando um Processo com AlarmManager

Para ativar um processo com `AlarmManager`, você primeiro tem que definir o alarme. No aplicativo Task Reminder, o melhor lugar para fazer isso é logo depois de salvar uma tarefa no `onClickListener()` do botão para salvar. Antes de adicionar o código, porém, você precisa acrescentar quatro arquivos de classe ao seu projeto:

- `ReminderManager.java`: Esta classe é responsável por configurar lembretes usando `AlarmManager`. O código para essa classe é mostrado na Listagem 13-1.
- `OnAlarmReceiver.java`: Esta classe é responsável por lidar com a transmissão (o broadcast) quando o alarme dispara. O código desta classe é mostrado na Listagem 13-2 (veja a seção "Criando a classe OnAlarmReceiver", posteriormente neste capítulo). Você precisa adicionar a seguinte linha de código ao elemento do aplicativo no arquivo `AndroidManifest.xml` para seu aplicativo reconhecer este receptor:

```
<receiver android:name=".OnAlarmReceiver" />
```

A sintaxe de ponto como prefixo informa ao Android que o receptor está no pacote atual — aquele que é definido no elemento do aplicativo do arquivo `AndroidManifest.xml`.

- `WakeReminderIntentService.java`: Esta classe abstrata é responsável por adquirir e liberar o bloqueio de ativação. O código desta classe é mostrado na Listagem 13-3 (veja a seção "Criando a classe WakeReminderIntentService", posteriormente neste capítulo).
- `ReminderService.java`: Esta classe é uma implementação do `WakeReminderIntentService` que lida com a construção da notificação, como mostrado no Capítulo 14. O código para essa classe é mostrado na Listagem 13-4 (veja a seção "Criando a classe ReminderService", posteriormente neste capítulo).

Você precisa adicionar a seguinte linha de código ao elemento do aplicativo no arquivo `AndroidManifest.xml` para seu aplicativo reconhecer `WakeReminderIntentService`:

```
<service android:name=".ReminderService" />
```

Criando a classe ReminderManager

A classe `ReminderManager` é responsável por configurar os alarmes usando a classe `AlarmManager` no Android. Você coloca nessa classe todas as ações relativas à definição dos alarmes a partir de `AlarmManager`.

Adicione a seguinte linha ao final de `onClickListener()` do `mConfirmButton` na classe `ReminderEditFragment` para adicionar um alarme para essa tarefa:

```
new ReminderManager(getActivity()).setReminder(mRowId,mCalendar);
```

Esta linha de código instrui `ReminderManager` para definir um novo lembrete para a tarefa com o ID indicado na linha `mRowId`, na data e hora determinadas, como definido pela variável `mCalendar`.

A Listagem 13-1 mostra o código da classe ReminderManager.

Listagem 13-1: A classe ReminderManager

```
public class ReminderManager {

private Context mContext;
private AlarmManager mAlarmManager;
public ReminderManager(Context context) {                          →5
   mContext = context;
   mAlarmManager =
       (AlarmManager)context.getSystemService(Context.ALARM_SERVICE);   →8
}

public void setReminder(Long taskId, Calendar when) {
   Intent i = new Intent(mContext, OnAlarmReceiver.class);         →11
   i.putExtra(RemindersProvider.COLUMN_ROWID, (long)taskId);       →12
                                                                   →13
   PendingIntent pi =
   PendingIntent.getBroadcast(mContext, 0, i,
            PendingIntent.FLAG_ONE_SHOT);
                                                                   →17
   mAlarmManager.set(AlarmManager.RTC_WAKEUP, when.getTimeInMillis(), pi);
      }                                                            →19
}
```

As linhas numeradas de código são explicadas nesta lista:

→ **5** A classe `ReminderManager` é instanciada com um objeto `Context`.

→ **8** Um `AlarmManager` é obtido via chamada `getSystemService()`.

→ **11** O método `setReminder()` é declarado com o ID de banco de dados da tarefa e o objeto `Calendar` de quando o alarme deve soar.

→ **12** Um novo objeto `Intent` é criado e é responsável por especificar o que deve acontecer quando o alarme dispara. Nesta instância, a linha especifica que o receptor `OnAlarmReceiver` deve ser chamado.

→ **13** O objeto `Intent` é fornecido com informações extras – o ID da tarefa no banco de dados.

→ **17** `AlarmManager` opera em um processo separado e para `AlarmManager` notificar um aplicativo de que uma ação precisa ser executada, um `PendingIntent` deve ser criado. Contém um objeto Intent que foi criado na linha 12. Nesta linha, um `PendingIntent` é criado com um flag `FLAG_ONE_SHOT` para indicar que esse `PendingIntent` pode ser usado apenas uma vez.

→ **19** O método `set()` de `AlarmManager` é chamado para agendar o alarme. O método `set()` é fornecido com estes parâmetros:

- type: `AlarmManager.RTC_MAKEUP` é a hora do relógio em UTC. Este parâmetro ativa o dispositivo quando a hora do argumento `triggerAtTime` especificada chega.
- `triggerAtTime:when.getTimeInMillis()` é a hora na qual o alarme deve disparar. O objeto `Calendar` fornece o método `getTimeInMillis()`, que converte a hora para um *valor long*, que representa a hora em unidades de milissegundos.
- `operation:pi` é a intenção pendente que age quando o alarme dispara. O alarme dispara na hora solicitada.

Se um alarme já estiver agendado com uma intenção pendente que contém a mesma assinatura, o alarme anterior será cancelado e o novo será configurado.

Criando a classe OnAlarmReceiver

A classe `OnAlarmReceiver`, mostrada na Listagem 13-2, é responsável por tratar a intenção que é iniciada quando um alarme é gerado. Essa classe age como um gancho no sistema de alarme porque é basicamente uma simples implementação de `BroadcastReceiver` — que pode reagir aos eventos de transmissão no sistema Android.

Listagem 13-2: A classe OnAlarmReceiver

```
public class OnAlarmReceiver extends BroadcastReceiver {
     @Override
     public void onReceive(Context context, Intent intent) {
        long rowid =
           intent.getExtras().getLong(ReminderProvider.COLUMN_ROWID);   →5

           WakeReminderIntentService.acquireStaticLock(context);        →7

           Intent i = new Intent(context, ReminderService.class);       →9
           i.putExtra(ReminderAdapter.COLUMN_ROWID, rowid);             →10
           context.startService(i);                                     →11

    }
}
```

As linhas numeradas são explicadas nesta lista:

→ 5 Esta linha recupera o ID de banco de dados da tarefa a partir da intenção, após o receptor ter começado a lidar com a intenção.

→ 7 `WakeReminderIntentService` requer um bloqueio na CPU para manter o dispositivo ativo enquanto o trabalho é realizado.

→ 9 Esta linha define um novo objeto Intent que inicia `ReminderService`.

→ 10 Esta linha coloca o ID da tarefa na intenção que inicia o serviço que faz o trabalho. Isto dá à classe `ReminderService` o ID da tarefa com a qual precisa trabalhar.

→ 11 Esta linha inicia `ReminderService`.

`onReceive()` de `OnAlarmReceiver` é o primeiro ponto de entrada para o alarme definido. Neste `BroadcastReceiver`, você não deseja que o dispositivo volte a hibernar durante o processamento. Sua tarefa nunca completaria e, possivelmente, poderia deixar seu aplicativo em um estado inconsistente com dados corrompidos no banco de dados.

Quando um alarme dispara, a intenção pendente agendada com o alarme é transmitida no sistema e qualquer receptor de transmissão capaz de lidar com ela faz o serviço.

Um componente `BroadcastReceiver` nada faz, exceto receber e reagir às mensagens de transmissão do sistema. Não exibe uma interface do usuário; contudo, inicia uma atividade em resposta à transmissão. `OnAlarmReceiver` é uma instância de `BroadcastReceiver`.

Quando `AlarmManager` transmite a intenção pendente, a classe `OnAlarmReceiver` responde à intenção — pois é endereçada a essa classe, como mostrado anteriormente, na linha 12 da Listagem 13-1. Essa classe, então, aceita a intenção, bloqueia a CPU e executa o trabalho necessário.

Criando a classe WakeReminderIntentService

A classe `WakeReminderIntentService` é a classe de base para a classe ReminderService, como mostrado na Listagem 13-3. Essa classe lida com o gerenciamento da aquisição e da liberação de um bloqueio de ativação da CPU, que mantém o dispositivo (mas não necessariamente a tela) ativo enquanto ocorre o trabalho. Depois que o trabalho terminar, essa classe libera o bloqueio de ativação para que o dispositivo possa voltar a hibernar.

Listagem 13-3: A classe WakeReminderIntentService

```
public abstract class WakeReminderIntentService extends IntentService {   →2
     abstract void doReminderWork(Intent intent);                          →5
                                                                           →6
     public static final String
        LOCK_NAME_STATIC="com.dummies.android.taskreminder.Static";
     private static PowerManager.WakeLock lockStatic=null;                 →9

     public static void acquireStaticLock(Context context) {
        getLock(context).acquire();
}                                                                          →13

     synchronized private static PowerManager.WakeLock
        getLock(Context context) {
       if (lockStatic==null) {                                             →17
        PowerManager
           mgr=(PowerManager)context
              .getSystemService(Context.POWER_SERVICE);                    →20

        lockStatic=mgr.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
                 LOCK_NAME_STATIC);                                        →23

       }
       return(lockStatic);                                                 →26
     }

     public WakeReminderIntentService(String name) {
       super(name);                                                        →31
     }
     @Override                                                             →33
     final protected void onHandleIntent(Intent intent) {
        try {                                                              →35
           doReminderWork(intent);
        } finally {
           getLock(this).release();
        }

     }
}
```

As linhas numeradas são explicadas nesta lista:

→ **2** Este método abstrato é implementado em qualquer filha desta classe — tal como na filha `ReminderService`, como mostrado posteriormente neste capítulo, na linha 7 da Listagem 13-4.

→ **5** É o nome da tag do bloqueio da CPU. Esse nome da tag auxilia a depuração.

→ **6** É a variável estática privada de bloqueio da ativação, que é referenciada e definida mais tarde nesta classe.

→ **9** É chamado o método `getLock()`. Depois dessa chamada ser retornada, o método `acquire()` é chamado para assegurar que o dispositivo esteja ativado no estado solicitado, um bloqueio de ativação parcial. Esse bloqueio de ativação impede que o dispositivo fique inativo, mas não ativa a tela.

→ **13** Esta linha define o método `getLock()` que retorna `PowerManager.WakeLock`, que permite informar ao Android que você deseja que o dispositivo faça algum trabalho.

→ **17** Esta linha recupera `PowerManager` a partir da chamada `getSystemService()`. É usado para criar o bloqueio.

→ **20** Isto cria um novo `WakeLock` usando a chamada do método `newWakeLock()`. Esse método aceita os seguintes parâmetros:

- `flags: A tag PARTIAL_WAKE_LOCK` informa ao Android que você precisa da CPU ativada, mas a tela não tem que estar ativada. Você pode fornecer várias outras tags a essa chamada também.
- `tag:LOCK_NAME_STATIC` é o nome da classe ou outra string. É usado para a depuração. Esta string personalizada é definida na linha 3.

→ **23** É retornado o `WakeLock` para quem chama.

→ **26** Aqui temos o construtor com o nome da instância-filha que o criou. Esse nome é usado para a depuração apenas.

→ **31** Esta é a chamada `onHandleIntent()` de `IntentService`. Assim que o serviço é iniciado, esse método é chamado para tratar a intenção transmitida para ele.

→ **33** O serviço tenta realizar o trabalho necessário chamando `doReminderWork()`.

→ **35** Independentemente da chamada para `doReminderWork()` ser bem-sucedida, certifique-se de que `WakeLock` seja retornado. Se você não o liberar, o dispositivo poderá ficar no estado On (Ativado) até o telefone ser reinicializado, o que é indesejável devido ao consumo de bateria. É por isso que o método `release()` é chamado na parte final do bloqueio `try-finally`. A parte final do bloqueio `try-finally` é sempre chamada, sem considerar se a tentativa teve sucesso.

Você deve assegurar que sempre chamará `release()` em um bloqueio de ativação exatamente tantas vezes quanto chamar `acquire()`. Você pode colocar suas chamadas `acquire()` e `release()` em um bloco `try-finally` para assegurar que `release()` será sempre chamado se uma exceção for gerada. Visite `http://docs.oracle.com/javase/tutorial/essential/exceptions/finally.html` (conteúdo em inglês) para obter mais informações sobre os blocos `try-finally`.

Embora nenhuma implementação de `doReminderWork()` exista em `ReminderService` ainda, o aplicativo Task Reminder responde aos alarmes. Sinta-se à vontade para configurar diversas tarefas e definir pontos de interrupção no depurador para ver o caminho de execução interromper no método `ReminderServicedoReminderWork()`.

`AlarmManager` não persiste os alarmes — se o dispositivo for reinicializado, os alarmes deverão ser configurados de novo. Sempre que o telefone é reinicializado, os alarmes precisam ser configurados novamente.

A Listagem 13-3 demonstra o que é necessário para executar o trabalho em um dispositivo que pode estar desativado ou bloqueado. O código adquire o bloqueio de ativação e enquanto o dispositivo está bloqueado em um estado ativado, `doReminderWork()` é chamado, sendo implementado no `ReminderService`.

Criando a classe ReminderService

A classe `ReminderService` (veja Listagem 13-4) é responsável por fazer o trabalho quando um alarme é disparado. A implementação neste capítulo simplesmente cria uma casca indicando onde o trabalho deve ocorrer (você implementará a notificação da barra de status no Capítulo 14).

Listagem 13-4: A classe ReminderService

```
public class ReminderService extends WakeReminderIntentService {          →1
    public ReminderService() {
        super("ReminderService");
    }

    @Override
    void doReminderWork(Intent intent) {                                  →7
        Long rowId = intent.getExtras()
            .getLong(ReminderProvider.COLUMN_ROWID);                      →9

        // O código de notificação da barra de status entra aqui.
    }
}
```

As linhas numeradas do código são explicadas nesta lista:

→ **1** Esta linha define a classe `ReminderService` herdando de `WakeReminderIntentService`.

→ **7** O método abstrato `doReminderWork()` em `WakeReminderIntentService` é implementado.

→ **9** Esta linha recupera o ID da tarefa que estava dentro do objeto `Intent` transmitido nesta classe.

A classe `ReminderService` não contém nenhuma implementação — apenas recupera o ID da tarefa a partir da intenção.

Reinicializando os Dispositivos

Provavelmente, você esquece coisas de tempos em tempos. É humano. O `AlarmManager` do Android não é diferente. O `AlarmManager` não persiste os alarmes; portanto, quando o dispositivo reinicializar, você deverá configurar os alarmes novamente.

Se você não configurar seus alarmes de novo, simplesmente eles não serão disparados, pois, para o Android, eles não existem.

Criando um receptor de inicialização

A permissão `RECEIVE_BOOT_COMPLETED` deixa que seu aplicativo receba uma notificação de transmissão a partir do Android quando o dispositivo termina de inicializar e pode interagir com o usuário. Como o sistema Android pode transmitir uma mensagem quando esse evento termina, você precisa adicionar outro `BroadcastReceiver` ao seu projeto. Esse `BroadcastReceiver` é responsável por lidar com a notificação de inicialização do Android. Quando a transmissão é recebida, o receptor precisa recuperar as tarefas de `ReminderProvider`, fazer um loop pelas tarefas e agendar um alarme para cada uma delas, para assegurar que seus alarmes não fiquem perdidos na reinicialização.

Adicione um novo `BroadcastReceiver` ao seu aplicativo. Para o aplicativo Task Reminder, `BroadcastReceiver` tem o nome `OnBootReceiver`. Você também precisa adicionar as seguintes linhas de código ao elemento do aplicativo no arquivo `AndroidManifest.xml`:

```
<receiver android:name=".OnBootReceiver" android:exported="false">
   <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED" />
   </intent-filter>
</receiver>
```

Este fragmento informa ao Android que `OnBootReceiver` deve receber notificações de inicialização para a ação `BOOT_COMPLETED`. Em termos

leigos, permite que `OnBootReceiver` saiba quando o dispositivo terminou de inicializar.

A implementação completa de `OnBootReceiver` é mostrada na Listagem 13-5.

Listagem 13-5: A classe OnBootReceiver

```
public class OnBootReceiver extends BroadcastReceiver {                    →1

    private static final String TAG = OnBootReceiver.class.getSimpleName();

    @Override
    public void onReceive(Context context, Intent intent) {                →6

        ReminderManager reminderMgr = new ReminderManager(context);        →8

        Cursor cursor = context.getContentResolver().query(
               ReminderProvider.CONTENT_URI, null, null, null, null);      →11

        if (cursor != null) {                                              →14
            cursor.moveToFirst();
                                                                           →17
            int rowIdColumnIndex = cursor
           .getColumnIndex(ReminderProvider.COLUMN_ROWID);                 →19
            int dateTimeColumnIndex = cursor
           .getColumnIndex(ReminderProvider.COLUMN_DATE_TIME);             →21

            while (cursor.isAfterLast() == false) {
                                                                           →23
                long rowId = cursor.getLong(rowIdColumnIndex);
                long dateTime = cursor.getLong(dateTimeColumnIndex);       →24

                Calendar cal = Calendar.getInstance();                     →27
                cal.setTime(new java.util.Date(dateTime));

                reminderMgr.setReminder(rowId, cal);                       →29

                cursor.moveToNext();                                       →31
            }
            cursor.close();                                                →33
        }
    }
}
```

As linhas numeradas são detalhadas nesta lista:

→ 1 Esta é a definição de `OnBootReceiver`.

→ 6 Este é o método `onReceive()`, chamado quando o receptor recebe uma intenção para realizar uma ação.

→ 8 Aqui é configurado um novo objeto `ReminderManager` que permite ao usuário agendar os alarmes.

→ 11 Aqui é obtido o cursor com todos os lembretes de `ReminderProvider` via `ContentResolver`. É parecido com as chamadas usadas para atualizar e apagar os lembretes em `ReminderEditFragment` e `ReminderListFragment`.

→ **14** Cursor é movido para o primeiro registro.

→ **17-19** Cada linha no cursor contém várias colunas de dados. A linha 17 encontra o índice para o ID e a linha 19 encontra o índice para a data e a hora.

Você deseja encontrar o ID da linha, assim como a data e a hora para que possa agendar o lembrete. Para obter esta informação, você precisa encontrar o índice das colunas que contém essas informações.

→ **21** Aqui é configurado um loop `while` que verifica se o cursor passou do último registro. Se for igual a false, o cursor irá para a linha 23. Se esse valor for true, nenhum outro registro estará disponível para usar no cursor.

→ **23-24** O ID e `dateTime` são recuperados no cursor desta linha usando os índices de coluna a partir das linhas 17-19.

→ **27** Após a data ser recuperada no cursor, a variável `Calendar` precisa ser atualizada com a hora correta. Essa linha define o objeto `Calendar` local para a hora da tarefa na linha.

→ **29** Aqui é agendado um novo lembrete com o ID da linha obtida a partir do banco de dados na hora definida pela variável `Calendar` recém-construída.

→ **31** Esta linha move o cursor para o próximo registro. Se nenhum outro registro existir no cursor, a chamada para `isAfterLast()` na linha 21 retornará `true`, significando que o loop `while` termina. Do contrário, a próxima linha será processada.

→ **33** Esta linha fecha o cursor porque não é mais necessário. Geralmente, os `BroadcastReceivers` não usam carregadores, portanto, você precisa fechar o cursor.

Quando você trabalhou anteriormente com o objeto Cursor no Capítulo 12, não teve que fechá-lo. Isso ocorreu porque o objeto `Loader` estava gerenciando o cursor.

Se iniciar o aplicativo, criar alguns lembretes, e então reinicializar o dispositivo, verá que os lembretes foram mantidos.

Verificando o receptor de inicialização

Se você não estiver certo se `OnBootReceiver` está funcionando, poderá colocar as seguintes instruções de registro no loop while:

```
Log.d("OnBootReceiver", "Adding alarm from boot.");
Log.d(TAG, "Row Id - " + rowId);
```

Este fragmento imprime mensagens no registro do sistema que são exibidas via Dalvik Debug Monitor Service (Serviço do Monitor de Depuração Dalvik) ou DDMS. Você poderá, então, fechar o emulador (ou o dispositivo) e iniciar de novo. Observe o fluxo de mensagens no DDMS e procure as mensagens `OnBootReceiver`. Se você tiver duas tarefas em seu banco de dados, deverá ver dois conjuntos de mensagens informando sobre o sistema adicionando um alarme a partir da inicialização. Então, a próxima mensagem deverá ser o ID do lembrete da tarefa. Veja o Capítulo 5 para obter mais informações sobre o DDMS.

Capítulo 14

Atualizando a Barra de Status do Android

Neste Capítulo

Compreendendo a barra de status

Trabalhando com o gerenciador de notificação do Android

Atualizando e limpando as notificações

Neste livro, você descobre várias maneiras de obter a atenção do usuário usando caixas de diálogo, mensagens toast e novas atividades. Embora essas técnicas funcionem bem em suas respectivas situações, em outras vezes, você precisa informar algo ao usuário sem tirar sua atenção da atividade atual. Esta é a finalidade da barra de status.

Desconstruindo a Barra de Status

A Figura 14-1 mostra a *barra de status*, a área onde você pode notificar o usuário sobre um evento. A barra de status pode manter muitos ícones. A que é exibida na Figura 14-1 mantém, da esquerda para a direita, uma notificação de calendário informando um compromisso e um ícone indicando que a depuração USB está ativada.

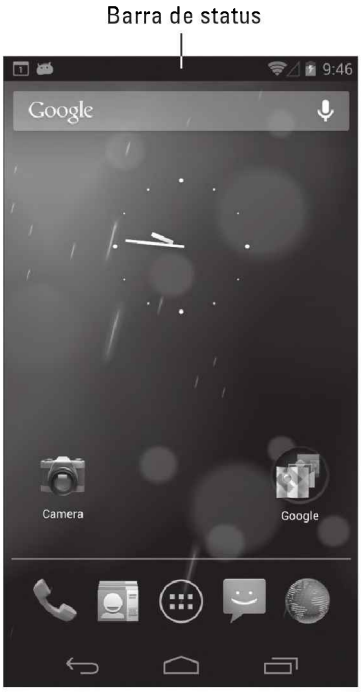


Figura 14-1: A barra de status, com alguns itens.

Os usuários também podem arrastar a barra de dados para baixo para ver mais informações, como mostrado na Figura 14-2. Agora, cada ícone da barra de status tem uma exibição expandida, onde mais informações podem ser exibidas.

Figura 14-2: Abrindo a barra de status.

Você pode informar aos usuários sobre várias atividades, tais como, o estado do dispositivo, notificações do correio e até progresso do download, como mostrado na Figura 14-3.

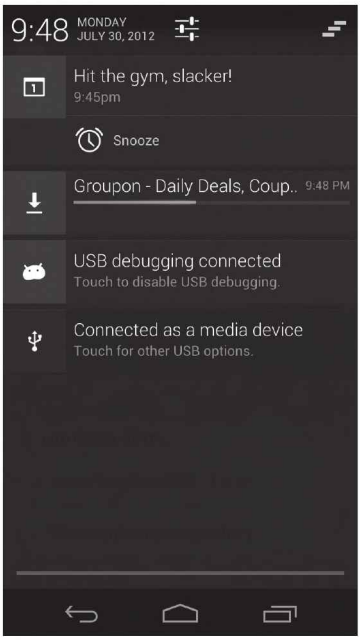


Figura 14-3: O carregador de progresso na barra de status.

Adicionar um ícone à barra de status não é sua única opção para alertar o usuário sobre uma notificação. Você pode incrementar uma notificação usando uma — ou mais — destas três opções:

- **Vibração:** O dispositivo vibra rapidamente quando uma notificação é recebida – útil quando o dispositivo está no bolso do usuário.
- **Som:** Um alarme soa quando a notificação é recebida. Um tom de chamada ou um tom gravado previamente que você instala junto com seu aplicativo é útil quando o usuário aciona o nível de som da notificação.
- **Luz:** A luz LED no dispositivo pisca em um dado intervalo de tempo, na cor especificada (muitos dispositivos contêm um LED que você pode programar). Se o LED suportar apenas uma cor, tal como branco, ele piscará ignorando a especificação de cor. Se o usuário definiu o nível do volume para silencioso, a luz fornecerá uma excelente dica de que algo precisa de atenção.

O Android 4.1 Jelly Bean introduziu melhorias drásticas nas notificações Android. Os dispositivos que executam o Jelly Bean podem ter notificações com estes recursos:

- **Visualização expansível:** O usuário pode expandir uma notificação usando o gesto 'apertar e ampliar'. A notificação expansível é um modo útil de mostrar aos usuários uma visualização expandida do conteúdo de notificação, tal como, uma visualização resumida da mensagem de um aplicativo de e-mail.
- **Botões de ação:** Um usuário sempre foi capaz de tocar em uma notificação para inicializar o aplicativo que a criou. Porém, você pode adicionar até três botões extras a um aplicativo Jelly Bean para executar qualquer operação desejada. Um excelente exemplo no aplicativo Task Reminder é fazer o botão Snooze (Soneca) descartar temporariamente a notificação e retorná-la depois.
- **Estilos de modelo variados:** O Jelly Bean vem com três novos estilos de notificações, como mostrado na Figura 14-4:
  - `BigTextStyle`: Mostra um `TextView` com diversas linhas
  - `BigInboxStyle`: Mostra uma lista de informações
  - `BigPictureStyle`: Mostra uma imagem
- **Tamanho maior:** As notificações Jelly Bean podem ser tão altas quanto 256 dp.

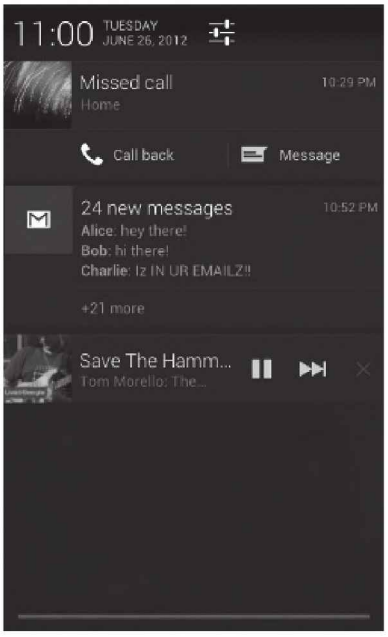


Figura 14-4: As notificações personalizadas no Jelly Bean.

Estes recursos não estão disponíveis nos antigos dispositivos, portanto, você não poderá usá-los nas versões anteriores a 4.1. Para dar às suas notificações mais impacto no Jelly Bean, visite `http://developer.android.com/guide/topics/ui/notifiers/notifications.html` (conteúdo em inglês) para obter mais informações.

Adicionar várias opções ao seu arsenal de notificações é muito útil porque permite ao usuário saber que o dispositivo precisa de atenção mesmo quando a tela está desativada.

Usando o Gerenciador de Notificações

Você usa o Notification Manager (Gerenciador de Notificações) para interagir com o mecanismo de notificações do Android. Trabalhar com o Notification Manager é tão simples quanto solicitá-lo a partir do contexto atual.

A seguinte linha de código obtém o objeto `NotificationManager` na chamada `getSystemService()` de Context:

```
NotificationManager mgr = (NotificationManager)getSystemService(NOTIFICATION_
      SERVICE);
```

Criando uma notificação

O aplicativo Task Reminder precisa notificar o usuário que uma tarefa precisa de atenção, através de um alarme indicando essa tarefa, por exemplo. Para definir essa notificação na barra de status, você usará `NotificationManager`.

No método `doReminderWork()` da classe ReminderService, digite o código mostrado na Listagem 14-1.

Listagem 14-1: A implementação de doReminderWork( )

```
Long rowId = intent.getExtras().getLong(ReminderProvider.COLUMN_ROWID);   →1

NotificationManager mgr =
            (NotificationManager)getSystemService(NOTIFICATION_SERVICE);  →4

Intent notificationIntent = new Intent(this, ReminderEditActivity.class);  →6
notificationIntent.putExtra(ReminderProvider.COLUMN_ROWID, rowId);

PendingIntent pi = PendingIntent.getActivity(this, 0, notificationIntent,  →7
      PendingIntent.FLAG_ONE_SHOT);

Notification note=new Notification(android.R.drawable.stat_sys_warning,    →9

            getString(R.string.notify_new_task_message),
               System.currentTimeMillis());

note.setLatestEventInfo(this, getString(R.string.notify_new_task_title),   →14
       getString(R.string.notify_new_task_message), pi);

note.defaults |= Notification.DEFAULT_SOUND;
note.flags |= Notification.FLAG_AUTO_CANCEL;                               →17

// Poderá ocorrer um problema se o usuário inserir mais de 2,147,483,647   →19
       tarefas (o valor int máximo).                                       →20

// Improvável, mas é bom registrar.

int id = (int)((long)rowId);                                               →28
mgr.notify(id, note);                                                      →27
```

As linhas numeradas na Listagem 14-1 são explicadas nesta lista:

→ 1 A intenção que iniciou `ReminderService` contém o ID da linha da tarefa com a qual você está trabalhando. Esse ID é necessário porque você o define como parte de `PendingIntent` para o status. Quando a notificação é selecionada na barra de status, `ReminderEditActivity` inicia com o ID da linha como parte da intenção pendente. Assim, `ReminderEditActivity` abre, lê os dados sobre esse determinado ID da linha e exibe-os para o usuário.

→ 4 Esta linha obtém uma instância de `NotificationManager`.

→ 6 Esta linha constrói uma nova intenção e define a classe para `ReminderEditActivity`. Essa atividade inicia quando o usuário seleciona a notificação.

→ 7 Esta linha coloca o ID da linha na intenção.

→ 9 Esta linha configura uma intenção pendente para ser usada pelo sistema de notificação. Como o sistema de notificação é executado em outro processo, um `PendingIntent` é requerido. O flag `FLAG_ONE_SHOT` indica que essa intenção pendente pode ser usada apenas uma vez.

→ 14 Esta linha constrói a `Notification` que aparece na barra de status. A classe `Notification` aceita estes parâmetros:

- `icon:android.R.drawable.stat_sys_warning` é o ID de recurso do ícone que deve ser colocado na barra de status. Esse ícone é um ponto de exclamação em um triângulo. Como é um ícone Android predefinido, você não tem que se preocupar em fornecer gráficos com densidade pequena, média, alta ou extra-alta — eles já estão predefinidos na plataforma.
- `tickerText:getString(R.string.notify_new_task_message)` é o texto que flui na barra de status de uma linha quando a notificação aparece pela primeira vez.
- `when:System.currentTimeMillis()` é a hora a mostrar no campo Time (Hora) da notificação.

  O Eclipse pode informá-lo que esse construtor e a chamada subsequente para `setLatestInfo()` estão deprecated (obsoletos) e que ao invés deles você deve usar `Notification.Builder`. `Notification.Builder` está disponível apenas na API nível 11 ou posterior, portanto, você não pode usar `Notification.Builder` nos antigos dispositivos.

→ 17 Esta linha define o conteúdo da exibição expandida com o layout Latest Event (Último Evento) padrão, como fornecido pelo Android. Por exemplo, você pode fornecer um layout XML personalizado para a exibição.

Nesta instância, você simplesmente fornece a exibição da notificação existente, não um layout personalizado. O método `setLastestEventInfo()` aceita estes parâmetros:

- `context:this` é o contexto a associar as informações do evento.
- `contentTitle:getString(R.string.notify_new_task_title)` é o título exibido na notificação quando ela é expandida.
- `contextText:getString(R.string.notify_new_task_message)` é o texto exibido abaixo do título na notificação quando ela é expandida.
- `contentIntent:pi` é a intenção a inicializar quando a exibição expandida é selecionada.

→ **19** Você usa este OR de bitwise ao definir o objeto `Notification` para incluir som durante o processo de notificação. Faz com que o som padrão da notificação seja reproduzido, caso o volume da notificação esteja ativado. Visite `http://en.wikipedia.org/wiki/Bitwise_operation#OR` (conteúdo em inglês) para obter mais informações sobre os operadores de bitwise.

→ **20** Você usa este OR de bitwise ao definir a propriedade do flag do objeto `Notification` que cancela a notificação depois que ela é selecionada pelo usuário.

→ **27** Esta linha faz a conversão do ID para um inteiro. O ID armazenado no banco de dados SQLite é long; porém, essa linha faz a conversão dele para um inteiro. Uma perda de precisão está acontecendo, mas esse aplicativo provavelmente nunca irá configurar mais de 2.147.483.647 tarefas (o número máximo que um inteiro pode armazenar no Java). Portanto, essa conversão é aceitável. A conversão para um inteiro é necessária porque o código na linha 20 aceita um inteiro como o ID para a notificação.

→ **28** Gera a notificação na barra de status. A chamada `notify()` aceita dois parâmetros:

- `id:` Este ID é único em seu aplicativo. Você usa o ID da linha da tarefa aqui.

- `Notification`: O objeto `note` que você acabou de criar descreve como notificar o usuário.

Exibindo o fluxo de trabalho

A Listagem 14-1 permite que este fluxo de trabalho ocorra:

1. O usuário está ativo em outro aplicativo, tal como um e-mail.
2. Uma tarefa é concluída e, portanto, o alarme inicia. A notificação é criada na barra de status.
3. O usuário pode escolher arrastar a barra de status para baixo, e então, selecionar a notificação ou ignorá-la no momento.

   Se o usuário escolher abrir deslizando na barra de status e selecionar um item, a intenção pendente na notificação será ativada. Isto, por sua vez, fará com que `ReminderEditActivity` abra com o ID da linha da tarefa fornecido.
4. A notificação é removida da barra de status.
5. As informações da tarefa são recuperadas no banco de dados e exibidas no formulário em `ReminderEditActivity`.

Adicionando recursos de string

Você pode notar que precisa adicionar estes dois recursos de string ao seu arquivo `strings.xml`:

- `notify_new_task_message`: Um valor é definido para o texto *A task needs to be reviewed!* (Uma tarefa precisa ser revista!) Esta mensagem tem duas finalidades — na exibição expandida e como o texto do registro quando a notificação chega pela primeira vez.
- `notify_new_task_title`: O valor é definido para *Task Reminder*, que é usado como o título para a exibição expandida.

Atualizando uma Notificação

Em algum ponto, você pode precisar atualizar a exibição de sua notificação, tal como, quando seu código é executado em segundo plano, para ver se as tarefas foram revistas. Esse código verifica se qualquer notificação está vencida. Suponha que depois de passar da marca de 2 horas, você queira mudar o ícone da notificação para um ponto de exclamação vermelho e piscar rapidamente o LED em vermelho. Felizmente, atualizar a notificação é um processo bem simples.

Se você chamar o método `notify()` novamente com um ID que já está ativo na barra de status, a notificação será atualizada na barra de status. Portanto, para atualizar a notificação, você simplesmente criará um novo objeto `Notification` com o mesmo ID e texto (mas, com um ícone vermelho diferente), e então chamará `notify()` de novo para atualizar a notificação.

Limpando uma Notificação

Os usuários constituem um grupo imprevisível — sejam usuários de primeira viagem, ou usuários avançados, eles podem estar localizados em qualquer lugar no mundo. Todos os usuários Android utilizam seus dispositivos à sua maneira. Em algum momento, um usuário pode ver uma notificação e decidir abrir o aplicativo usando a inicialização do aplicativo. Se isto acontecer enquanto a notificação está ativa, ela será mantida. Mesmo que o usuário esteja executando a tarefa, a notificação ainda será mantida na barra de status. Seu aplicativo deve ser capaz de reconhecer seu estado e tomar as devidas medidas para cancelar qualquer notificação existente para a tarefa. Contudo, se o usuário abrir seu aplicativo e rever uma tarefa diferente que não tem nenhuma notificação ativa, seu aplicativo não deverá limpar a notificação.

Limpe apenas a notificação que o usuário está revendo.

O `NotificationManager` simplifica o cancelamento de uma notificação existente usando o método `cancel()`. Esse método aceita um parâmetro – o ID da notificação. Você pode chamar de novo usando o ID da tarefa como o ID da nota. O ID da tarefa é único para o aplicativo Task Reminder. Fazendo isso, você poderá abrir facilmente uma tarefa e cancelar qualquer notificação existente chamando o método `cancel()` com o ID da tarefa.

Em algum momento, você poderá também precisar limpar todas as notificações mostradas anteriormente. Para tanto, simplesmente chame o método `cancelAll()` em `NotificationManager`.

Capítulo 15

Trabalhando com a Estrutura de Preferências do Android

Neste Capítulo

Vendo como as preferências funcionam no Android

Construindo uma tela de preferências

Trabalhando programaticamente com as preferências

A maioria dos programas precisa ser configurada para se adequar às necessidades do usuário através de definições ou preferências individuais. Permitir que os usuários configurem seu aplicativo Android fornece uma vantagem de utilização. Felizmente, criar e fornecer um mecanismo para editar as preferências no Android são processos bem fáceis.

O Android fornece uma estrutura (um framework) de preferências robusta que permite definir de modo declarativo — e programático — as preferências de seu aplicativo. O Android armazena as preferências como pares de chave-valor persistentes de tipos de dados primitivos para você. Você não precisa armazenar os valores em um arquivo, banco de dados ou em nenhum outro mecanismo. A estrutura de preferências do Android guarda os valores fornecidos no armazenamento interno em nome de seu aplicativo. Você pode usar a estrutura de preferências para armazenar elementos booleanos, float, int, long e de string. Os dados *persistem* entre as seções do usuário — se o usuário fechar o aplicativo e reabri-lo depois, as preferências serão salvas e poderão ser usadas, mesmo que seu aplicativo seja encerrado.

Este capítulo entra na estrutura de preferências do Android e descreve como incorporá-la em seus aplicativos. Você descobrirá como usar a `PreferenceActivity` predefinida para criar e editar as preferências, e como ler e gravar as preferências através do código de seu aplicativo. No final deste capítulo, você terá integrado totalmente as preferências no aplicativo Task Reminder.

Se você estiver usando o Android 3.x e posterior, e não quiser suportar os antigos dispositivos, considere usar o novo `PreferenceFragment`, ao invés de `PreferenceActivity`. Contudo, `PreferenceFragment` não está disponível ainda como parte da biblioteca de suporte, portanto, você não pode usar os `PreferenceFragments` nos antigos dispositivos. Por isso, este capítulo usa a consagrada `PreferenceActivity`.

Compreendendo a Estrutura de Preferências do Android

Uma excelente qualidade da estrutura de preferências do Android é a simplicidade para desenvolver uma tela que permite aos usuários modificarem suas preferências. Grande parte do trabalho pesado é feito para você pelo Android porque desenvolver uma tela de preferências é tão simples quanto defini-la no XML localizado na pasta `res/xml` de seu projeto. Embora esses arquivos XML não sejam iguais aos arquivos de layout, existem definições XML específicas que definem as telas, categorias e preferências reais. As preferências comuns que são predefinidas na estrutura incluem

- `EditTextPreference`: Armazena texto comum como uma string
- `CheckBoxPreference`: Armazena um valor booleano
- `RingtonePreference`: Permite que o usuário armazene um tom de chamada a partir dos disponíveis no dispositivo
- `ListPreference`: Permite que o usuário selecione um item em uma lista de itens na caixa de diálogo

Se as preferências predefinidas não forem adequadas às suas necessidades, você poderá criar sua própria preferência derivando-a da classe `Preference` de base ou `DialogPreference`, que é a classe de base para as preferências que são estabelecidas na caixa de diálogo. Tocar em uma dessas preferências abre uma caixa de diálogo mostrando os controles da preferência. Exemplos da `DialogPreference` predefinida são `EditTextPreference` e `ListPreference`.

O Android também fornece uma `PreferenceActivity`, na qual você pode carregar uma tela de preferências do mesmo modo como carrega um layout para uma classe `Activity` básica. A classe de base permite que você toque nos eventos `PreferenceActivity` e execute operações avançadas, tal como, definir uma `EditTextPreference` para aceitar apenas números.

Compreendendo a classe PreferenceActivity

A responsabilidade da classe `PreferenceActivity` é mostrar uma hierarquia de objetos `Preference` em listas, possivelmente estendendo-se em diversas telas, como mostrado na Figura 15-1.

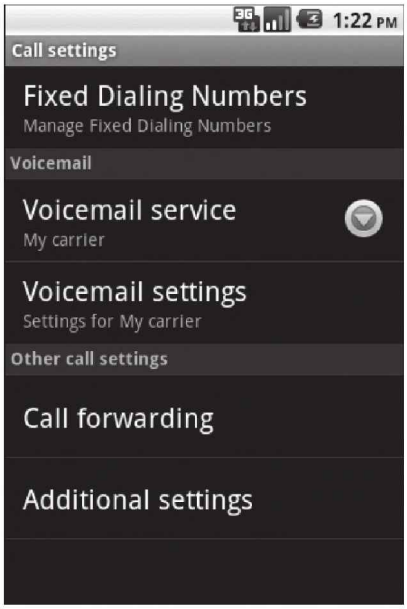


Figura 15-1: A tela de preferências para as definições da chamada no Android.

Quando as preferências são editadas, elas são armazenadas usando uma instância de `SharedPreferences`. A classe `SharedPreferences` é uma interface para acessar e modificar os dados de preferência retornados por `getSharedPreferences( )` de qualquer objeto `Context`.

`PreferenceActivity` é uma classe de base parecida com a classe de base `Activity`. Contudo, `PreferenceActivity` se comporta de modo um pouco diferente. Um dos recursos mais importantes que `PreferenceActivity` oferece é a exibição das preferências com um estilo visual que lembra as preferências do sistema. Isto dá a seu aplicativo uma aparência consistente no painel em relação aos componentes da interface de usuário Android. Você deve usar `PreferenceActivity` ao lidar com as telas de preferências em seus aplicativos Android.

Mantendo os valores de preferência

Como a estrutura Android armazena as preferências em `SharedPreferences`, que guarda automaticamente os dados de preferência no armazenamento interno, você pode criar facilmente uma preferência. Quando um usuário edita uma preferência, o valor é salvo automaticamente para você; você não precisa persisti-la manualmente.

A Figura 15-2 mostra uma preferência sendo definida no aplicativo Task Reminder. Quando o usuário toca em OK, o Android mantém o valor para `SharedPreferences`. O Android faz todo o trabalho pesado em relação a manter os valores da preferência.

Figura 15-2: Definindo uma preferência.

Layout das preferências

Trabalhar com layouts no Android algumas vezes pode ser um processo meticuloso de alinhamento, gravidade e outros fatores complicados. Construir layouts é quase como construir um site web com tabelas por toda parte. Algumas vezes, é fácil; outras, não. Felizmente, fazer o layout das preferências Android é muito mais simples do que definir um layout para a tela do aplicativo.

As telas de preferências do Android são divididas nestas categorias:

- `PreferenceScreen`: Representa uma preferência de alto nível que é a raiz de uma hierarquia de preferências. Você pode usar uma `PreferenceScreen` neste dois lugares:

- *Em uma* `PreferenceActivity`: `PreferenceScreen` não é mostrada porque exibe apenas as preferências contidas na definição `PreferenceScreen`.
- *Em outra hierarquia de preferências:* Quando presente em outra hierarquia, `PreferenceScreen` serve como um acesso a outra tela de preferências (parecido com aninhar as declarações `PreferenceScreen` dentro de outras declarações `PreferenceScreen`). Embora este conceito possa parecer confuso, você pode considerá-lo como um arquivo XML, onde você pode declarar um elemento e qualquer elemento pode conter o mesmo elemento-pai. Nesse ponto, você está aninhando os elementos. A mesma afirmação aplica-se a `PreferenceScreen`. Aninhando `PreferenceScreen`, você está informando que ela deve mostrar uma nova tela quando selecionada.

- `PreferenceCategory`: Esta preferência é usada para agrupar os objetos de preferência e fornecer, acima do grupo, um título que descreve a categoria.
- `Preference`: Uma preferência que é mostrada na tela. Essa pode ser uma preferência comum ou personalizada por você.

Fazendo o layout de uma combinação de `PreferenceScreen`, `PreferenceCategory` e `Preference` no XML, você poderá criar facilmente uma tela de preferências parecida com a Figura 15-1.

Criando Sua Tela de Preferências

Criar preferências usando `PreferenceActivity` e um arquivo XML de preferências é um processo bem simples. A primeira coisa a fazer é criar o arquivo XML de preferências, que define o layout das preferências e os valores dos recursos de string que aparecem na tela. Esses recursos de string são apresentados como `TextView` na tela para ajudar o usuário a determinar o que faz a preferência.

Sua `PreferenceScreen` deve dar aos usuários a chance de definir a hora padrão para um lembrete (em minutos) e um título padrão para uma nova atividade. Como o aplicativo está agora, o título padrão está vazio e a hora padrão do lembrete está definida para a hora atual. Essas preferências permitem que o usuário economize algumas etapas enquanto constrói novas tarefas. Por exemplo, se o usuário normalmente constrói tarefas configurando a hora de lembrete para 60 minutos após a hora atual, agora, o usuário poderá especificar este intervalo nas preferências. Esse novo valor se tornará o valor da hora do lembrete quando o usuário criar uma nova tarefa.

Construindo o arquivo de preferências

Para construir sua primeira tela de preferências, crie uma pasta `res/xml` em seu projeto. Dentro desta pasta, crie um arquivo XML e nomeie-o como `task_preferences.xml`. Adicione o código na Listagem 15-1 ao arquivo.

Listagem 15-1: O arquivo task_preferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen                                                    →2
  xmlns:android="http://schemas.android.com/apk/res/android">
     <PreferenceCategory                                             →4
       android:key="@string/pref_category_task_defaults_key"         →5
       android:title="@string/pref_category_task_defaults_title">    →6
       <EditTextPreference                                           →7
          android:key="@string/pref_task_title_key"                  →8
          android:dialogTitle="@string/pref_task_title_dialog_title" →9
          android:dialogMessage="@string/pref_task_title_message"    →10
          android:summary="@string/pref_task_title_summary"         →11
          android:title="@string/pref_task_title_title" />           →12
     </PreferenceCategory>
     <PreferenceCategory                                             →13
          android:key="@string/pref_category_datetime_key"           →14
          android:title="@string/pref_category_datetime_title">      →15
            <EditTextPreference                                      →16
              android:key="@string/pref_default_time_from_now_key"   →17
    android:dialogTitle="@string/pref_default_time_from_now_dialog_title" →18
    android:dialogMessage="@string/pref_default_time_from_now_message"   →19
    android:summary="@string/pref_default_time_from_now_summary"         →20
    android:title="@string/pref_default_time_from_now_title" />          →21
     </PreferenceCategory>
</PreferenceScreen>
```

Poucos recursos de string são introduzidos na Listagem 15-1 (veja a próxima seção). Cada linha de código numerada é explicada a seguir:

→ 2 Esta é a `PreferenceScreen` no nível da raiz; é o contêiner para a própria tela. Todas as outras preferências ficam abaixo desta declaração.

→ 4 Esta é a `PreferenceCategory` que define a categoria para os padrões da tarefa, tais como, o título ou o corpo. Como você pode ter notado, a linha 13 declara outra `PreferenceCategory` para a hora da tarefa padrão. Normalmente, você coloca esses dois itens na mesma categoria; eles estão separados aqui para mostrar como usar diversos elementos `PreferenceCategory` em uma tela.

→ 5 Esta linha define a chave usada para armazenar e recuperar a preferência em `SharedPreferences`. Essa chave deve ser única.

→ **6** Esta linha define o título da categoria.

→ **7** Esta linha contém a definição de `EditTextPreference`, que é responsável por armazenar a preferência para o título padrão de uma tarefa.

→ **8** Esta linha contém a chave para o texto do título padrão `EditTextPreference`.

→ **9** `EditTextPreference` é uma classe-filha de `DialogPreference`. Quando um usuário seleciona a preferência, vê uma caixa de diálogo parecida com a mostrada na Figura 15-2. Essa linha de código define o título dessa caixa de diálogo.

→ **10** Esta linha define a mensagem que aparece na caixa de diálogo.

→ **11** Esta linha define o texto de resumo presente na tela de preferências, como mostrado na Figura 15-1.

→ **12** Esta linha define o título da preferência na tela de preferências.

→ **13** Esta linha define `PreferenceCategory` para a hora da tarefa padrão.

→ **14** Esta linha define a chave da categoria.

→ **15** Esta linha define o título da categoria.

→ **16** Esta linha é o início da definição de `EditTextPreference`, que armazena a hora padrão em minutos (dígitos) que a hora do lembrete da tarefa tem como padrão a partir da hora atual.

→ **17** Esta linha define a chave para a preferência padrão da hora da tarefa.

→ **18** Esta linha define o título da caixa de diálogo que se abre quando a preferência é selecionada.

→ **19** Esta linha define a mensagem que está presente na caixa de diálogo.

→ **20** Esta linha define o resumo da preferência presente na tela principal de preferências, como mostrado na Figura 15-1.

→ **21** Esta linha define o título da preferência na tela de preferências.

Adicionando recursos de string

Para seu aplicativo compilar, você precisa dos recursos de string para as preferências. No arquivo `res/values/strings.xml`, adicione estes valores:

```
<!-- Preferências -->
<string name="pref_category_task_defaults_key">task_default_category</string>
<string name="pref_category_task_defaults_title">Task Title Default</string>
<string name="pref_task_title_key">default_reminder_title</string>
<string name="pref_task_title_dialog_title">Default Reminder Title</string>
<string name="pref_task_title_message">The default title for a reminder.</string>
<string name="pref_task_title_summary">Default title for reminders.</string>
<string name="pref_task_title_title">Default Reminder Title</string>
<string name="pref_category_datetime_key">date_time_default_category</string>
<string name="pref_category_datetime_title">Date Time Defaults</string>
<string name="pref_default_time_from_now_key">time_from_now_default</string>
<string name="pref_default_time_from_now_dialog_title">Time From Now</string>
<string name="pref_default_time_from_now_message">The default time from
    now (in minutes) that a new reminder should be set to.</string>
<string name="pref_default_time_from_now_summary">Sets the default
    time for a reminder.</string>
<string name="pref_default_time_from_now_title">Default Reminder Time</string>
```

Agora, você deverá ser capaz de compilar seu aplicativo.

Trabalhando com a Classe PreferenceActivity

Definir uma tela de preferências é bem simples: forneça os valores aos atributos necessários e estará pronto. Embora a tela de preferências possa ser definida no XML, simplesmente defini-la no arquivo não significa que aparecerá na tela. Para exibir sua tela de preferências, você criará uma `PreferenceActivity`.

`PreferenceActivity` mostra uma hierarquia de preferências na tela de acordo com um arquivo de preferências definido no formato XML — tal como aquele que você pode ter acabado de criar. As preferências podem estender-se por diversas telas (se diversos objetos `PreferenceScreen` estiverem presentes e aninhados). Essas preferências são salvas automaticamente em `SharedPreferences`. Como bônus, as preferências mostradas automaticamente seguem o estilo visual das preferências do sistema, permitindo que seu aplicativo tenha uma experiência de usuário similar à plataforma Android padrão.

Para inserir e exibir a `PreferenceScreen` que você acabou de construir, adicione uma atividade que derive de `PreferenceActivity` ao seu aplicativo e nomeie-a como `TaskPreferences`. Adicione o código na Listagem 15-2.

Listagem 15-2: O arquivo TaskPreferences

```
public class TaskPreferences extends PreferenceActivity {                  →1
     @Override
     protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.task_preferences);                →5

        EditTextPreference timeDefault = (EditTextPreference)
   findPreference(getString(R.string.pref_default_time_from_now_key));     →8
  timeDefault.getEditText().setKeyListener(DigitsKeyListener.getInstance()); →9
    }
}
```

Este é todo o código necessário para exibir, editar e manter as preferências no Android. As linhas numeradas de código são explicadas nesta lista:

→ 1 O arquivo da classe `TaskPreferences` é herdado da classe `PreferenceActivity` de base.

→ 5 A chamada para o método `addPreferencesFromResource()` é fornecida com o ID de recurso do arquivo `task_preferences.xml` armazenado no diretório `res/xml`. Esse método é obsoleto em favor de `PreferenceFragment`, segundo a biblioteca de suporte, portanto, o Eclipse exibirá um aviso que você pode ignorar.

→ 8 Esta linha recupera `EditTextPreference` para a hora padrão do lembrete da tarefa chamando o método `findPreference()` e fornecendo-lhe a chave definida no arquivo `task_preferences.xml`. Esse método também é obsoleto em favor de `PreferenceFragment`, segundo a biblioteca de suporte, portanto, o Eclipse exibirá um aviso que você pode ignorar.

→ 9 Esta linha obtém o objeto `EditText`, que é um filho de `EditTextPreference`, usando o método `getEditText()`. Esse objeto define o listener de teclas, que é responsável por ouvir aos eventos de tecla pressionada. O método `setKeyListener()` define o listener de teclas e, fornecendo-lhe uma instância de `DigitsKeyListener`, `EditTextPreference` permite que os dígitos sejam inseridos em `EditTextPreference` apenas para a hora padrão do lembrete.

Você não deseja que os usuários insiram valores, tais como, `foo` ou `bar` neste campo porque não são valores inteiros válidos. Usar `DigitsKeyListener` assegura que os valores transmitidos nas preferências sejam somente dígitos.

Neste ponto, você pode usar sua atividade. `PreferenceActivity` permite que os usuários editem e salvem suas preferências. Como você pode ver, esta implementação requer apenas um fragmento de código. A próxima etapa é exibir a tela de preferências adicionando um item de menu.

Adicione sua nova `PreferenceActivity` ao arquivo `AndroidManifest.xml` usando esta linha de código:

```
<activity android:name=".TaskPreferences" android:label="@string/app_name" />
```

Abrindo a classe PreferenceActivity

Para abrir esta nova atividade, você adiciona um item de menu a `ReminderListActivity`. Para adicionar um novo item de menu, você acrescenta uma nova definição do menu ao arquivo `list_menu.xml` localizado no diretório `res/menu`. Atualizar este arquivo atualiza o menu em `ReminderListActivity`. O arquivo `list_menu.xml` atualizado é mostrado aqui com a nova entrada em negrito:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
   xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_insert"
      android:icon="@android:drawable/ic_menu_add"
      android:title="@string/menu_insert" />
      <item android:id="@+id/menu_settings"
         android:icon="@android:drawable/ic_menu_preferences"
         android:title="@string/menu_settings" />
</menu>
```

O último item adiciona um item de menu para as definições, usando o ícone Settings (Definições) predefinido do Android e o recurso de string `menu_settings`. Você adiciona um novo recurso de string chamado `menu_settings` com um valor `Settings` em seus recursos de string.

Lidando com as seleções de menu

Depois de seu menu ser atualizado, o aplicativo precisa responder sempre que o usuário toca em um item de menu. Para tornar isto possível, você adiciona código ao método `onOptionsItemSelected()` em `ReminderListFragment`. O código para lidar com a seleção do menu de definições está em negrito neste fragmento:

```
@Override
public boolean onOptionsItemSelected (MenuItem item) {
        switch(item.getItemId()) {
           case R.id.menu_insert:
              ((OnEditReminder) getActivity()).editReminder(0);
              return true;
           case R.id.menu_settings:
              Intent i = new Intent(getActivity(), TaskPreferences.class);
              startActivity(i);
              return true;
           }
        return super.onOptionsItemSelected(item);
}
```

Este código cria um novo objeto `Intent` com uma classe de destino `TaskPreferences`. Um usuário que selecione o item de menu Settings (Definições) vê a tela de preferências para editar suas preferências. Se você iniciar o aplicativo e selecionar Settings, deverá ver uma tela parecida com a mostrada na Figura 15-3.

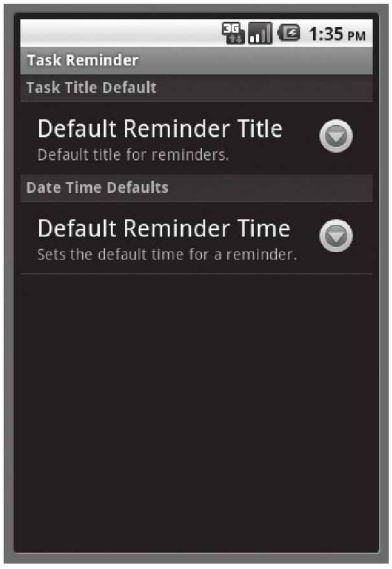


Figura 15-3: A tela de preferências.

Trabalhando com as Preferências em Suas Atividades Durante a Execução

Embora definir as preferências em uma `PreferenceActivity` seja útil, não gera nenhum valor, a menos que você possa ler as preferências a partir do objeto `SharedPreferences` durante a execução e usá-las em seu aplicativo. Felizmente, o Android simplifica muito o processo.

No aplicativo Task Reminder, você lê esses valores em `ReminderEditFragment` para definir os valores padrão quando um usuário cria uma nova tarefa. Como as preferências são armazenadas em `SharedPreferences`, você pode acessá-las em várias atividades em seu aplicativo.

Recuperando os valores de preferência

Abra `ReminderEditFragment` e navegue para o método `onCreateView()`. Ele determina se a tarefa é existente ou nova. Se a tarefa for nova, você obterá os valores padrão de `SharedPreferences` e irá carregá-los na atividade para o usuário. Se por alguma razão o usuário não especificou suas

preferências, serão strings vazias e você irá ignorar os padrões. Você usará as preferências apenas se o usuário as definiu.

Para recuperar os valores da preferência, você usa o objeto `SharedPreferences`, como mostrado na Listagem 15-3. Adicione o código em negrito à parte inferior de `onCreateView()`.

Listagem 15-3: Recuperando os valores do Shared Preferences

```
if (mRowId == 0) {                                                    →1
   // Esta é uma nova tarefa - adiciona os padrões a partir das
    preferências, se definidas.
   SharedPreferences prefs = PreferenceManager                        →3
    .getDefaultSharedPreferences(getActivity());
   String defaultTitleKey = getString(R.string.pref_task_title_key);  →5
   String defaultTimeKey = getString(R.string.pref_default_time_
    from_now_key);                                                    →7
   String defaultTitle = prefs.getString(defaultTitleKey, null);      →8
   String defaultTime = prefs.getString(defaultTimeKey, null);        →9

   if (defaultTitle != null)
      mTitleText.setText(defaultTitle);                               →12

   if (defaultTime != null && defaultTime.length()>0)
      mCalendar.add(Calendar.MINUTE, Integer.parseInt(defaultTime));  →15

   updateButtons();                                                   →17

} else {

   // Dispara um carregador em segundo plano para recuperar os
     dados no banco
   // de dados.
   getLoaderManager().initLoader(0, null, this);

}
```

Cada nova linha de código é explicada nesta lista:

→ 1 A instrução `if` lida com a lógica da nova tarefa.

→ 3 Esta linha recupera o objeto `SharedPreferences` a partir da chamada estática `getDefaultSharedPreferences()` no objeto `PreferenceManager`.

→ 5 Esta linha recupera o valor da chave para a preferência padrão do título a partir dos recursos de string. Esta mesma chave é usada na Listagem 15-1 para definir a preferência.

→ 7 Esta linha recupera o valor da chave para o deslocamento (offset) de tempo padrão, em minutos, a partir das preferências.

→ **8** Esta linha recupera o valor padrão do título a partir das preferências com uma chamada para `getString()` no objeto `SharedPreferences`. O primeiro parâmetro é a chave para a preferência e o segundo é o valor padrão, caso a preferência não exista (ou não foi definida ainda). Neste caso, o valor padrão será `null`.

→ **9** Esta linha recupera o valor da hora padrão a partir das preferências, usando o mesmo método descrito na linha 8 com uma chave diferente.

→ **12** Esta linha define o valor de texto da exibição `EditText` — que é o título da tarefa. Esse valor será definido se a preferência não for uma string vazia.

→ **15** Esta linha aumenta a hora no objeto `Calendar` local chamando o método `add()` com o parâmetro `Calendar.MINUTE`, caso o valor das preferências não seja uma string vazia. A constante `Calendar.MINUTE` informa ao objeto `Calendar` que o próximo parâmetro deve ser tratado como minuto e o valor deve ser adicionado ao campo Minute (Minuto) do calendário. Se os minutos forçarem o calendário para uma nova hora ou dia, o objeto `Calendar` atualizará os outros campos para você.

Por exemplo, se o calendário originalmente estava definido para 2012-08-31 11:45 pm e você adicionar 60 minutos a ele, o novo valor do calendário será 2012-09-01 12:45 am. Como `EditTextPreference` armazena todos os valores como strings, a string será convertida para um inteiro com o método `Integer.parseInt()`. Adicionando a hora ao objeto `Calendar` local, o seletor de hora e o texto do botão associado à abertura do seletor de hora serão atualizados também.

→ **17** Esta linha atualiza o texto do botão da hora para refletir a hora que foi adicionada ao objeto local `Calendar` existente.

Quando você iniciar o aplicativo, poderá definir as preferências e vê-las refletidas quando escolher adicionar uma nova tarefa à lista. Tente limpar as preferências, e então escolher criar uma nova tarefa. Note que os padrões não serão mais aplicados — fácil!

Definindo os valores de preferência

Embora atualizar os valores de preferência via Java não seja feito no aplicativo Task Reminder, às vezes, você poderá precisar fazê-lo em seus próprios aplicativos. Suponha que você desenvolva um aplicativo para um sistema de bilhetes de suporte que requer que os usuários entrem com seus departamentos atuais. Você tem um objeto `preference` para o departamento padrão, mas o usuário nunca usa a tela de preferências e, portanto, entra repetidamente com o departamento em seu aplicativo de modo manual. Usando uma lógica definida e programada, você determina

que o usuário está entrando com o mesmo departamento para cada bilhete de suporte (suponha que seja o departamento de Contabilidade), portanto, você solicita que ele determine se deseja definir o departamento padrão para Accounting (Contabilidade). Se ele escolher Yes (Sim), você atualizará as preferências para ele de forma programada.

Para editar as preferências programaticamente, você precisa de uma instância de `SharedPreferences`. Você pode obtê-la via `PreferenceManager`, como mostrado na Listagem 15-4. Depois de conseguir uma instância de `SharedPreferences`, você poderá editar as várias preferências obtendo uma instância do objeto Editor da preferência. Após as preferências serem editadas, você precisará aceitá-las, também demonstrado na Listagem 15-4.

Listagem 15-4: Editando programaticamente as preferências

```
SharedPreferences prefs =
   PreferenceManager.getDefaultSharedPreferences(this);          →1
Editor editor = prefs.edit();                                    →2
editor.putString("default_department", "Accounting");            →3
editor.commit();                                                 →4
```

As linhas numeradas de código são explicadas nesta lista:

→ 1 Uma instância de `SharedPreferences` é recuperada de `PreferenceManager`.

→ 2 Uma instância do objeto `Editor` de preferências é obtida chamando o método `edit()` no objeto `SharedPreferences`.

→ 3 Esta linha edita uma preferência com o valor da chave `default_department` chamando o método `putString()` no objeto `Editor`. O valor é definido para "Accounting". Normalmente, o valor da chave é recuperado nos recursos de string e o valor da string é recuperado via seu programa ou entrada do usuário. O fragmento de código permanece simples para abreviar.

→ 4 Após as alterações serem feitas para qualquer preferência, você deve chamar o método `commit()` no objeto `Editor` para persisti-las em `SharedPreferences`. A chamada substitui automaticamente qualquer valor armazenado em `SharedPreferences` pela chave dada na chamada `putString()`.

Se você não chamar `commit()` no objeto `Editor`, suas alterações não serão mantidas e seu aplicativo não funcionará como o esperado.

Parte IV

Tablets

A 5ª Onda Por Rich Tennant

"Até termos resolvido os problemas, David fornecerá a parte de áudio da demonstração de nosso aplicativo."

Nesta parte...

A Parte IV apresenta o mundo do desenvolvimento Android para tablets. Os tablets Android são diferentes dos telefones Android e esta parte irá orientá-lo nas alterações necessárias em seu aplicativo Task Reminder para executá-lo nos tablets Android.

Caso você queira ir além dos tablets Google Android padrão, encontrará tudo que precisa saber para suportar os dispositivos baseados no Android (mas não no Google), como, por exemplo, o Amazon Kindle Fire, no Capítulo 18.

Capítulo 16

Desenvolvendo para Tablets

Neste Capítulo

Entendendo porque o tablet é diferente.

Modificando seu aplicativo existente para executar nos tablets

Você precisa dominar alguns truques que fazem seus aplicativos funcionarem nos tablets e nos telefones. Neste capítulo, você terá uma visão geral das diferenças entre telefones e tablets e descobrirá como projetar o aplicativo Task Reminder para trabalhar nos dois tipos de dispositivos.

Considerando a Diferença Entre Telefones e Tablets

Os tablets e os telefones Android têm algumas diferenças óbvias e o tamanho vem imediatamente à mente, mas há algumas outras:

- Os tablets Android tendem a ser muito maiores que seus correspondentes em telefone.
- Os tablets são projetados para serem segurados com as duas mãos, ao passo que os telefones são projetados para apenas uma.
- As telas do tablet Android tendem a não passar de 7 a 10 polegadas e os maiores telefones ficam em torno de 5 polegadas.

  A linha que separa tablets e o telefones pode ser a marca das 5 polegadas. Alguns dispositivos "intermediários" são comercializados como telefones e outros com praticamente as mesmas especificações são comercializados como tablets.
- A orientação do tablet varia dependendo do uso, ao passo que quase todos os telefones Android utilizam a orientação retrato para suas telas.

Muitos tablets Android são projetados para a exibição de mídia utilizando a largura da tela, portanto, favorecem a orientação paisagem. Outros, tais como o Nexus 7 e o Kindle Fire, são projetados basicamente para serem usados no modo Retrato. Isto não quer dizer que você não possa executar um aplicativo no modo Retrato em um tablet paisagem (ou vice-versa), mas saiba que muitos usuários podem executar seu aplicativo em uma orientação diferente daquela na qual você completou grande parte de seus testes.

Os tablets e os telefones também têm algumas diferenças no projeto do hardware e operação que afetam o projeto do aplicativo. Esta lista descreve-as da perspectiva do tablet:

- Os tablets geralmente não têm as conexões de dados 3G ou 4G sempre ativadas.
- Os tablets tendem a ser maiores, usam baterias maiores e aproveitam muito mais a duração da bateria do que seus correspondentes em telefone.
- Os tablets podem ter câmeras mais baratas — ou nenhuma câmera — porque as câmeras do tablet geralmente são menos usadas do que as do telefone.
- Geralmente, os tablets não têm as capacidades comuns dos telefones, tais como o serviço de localização GPS.

E mais, não fique surpreso se você tiver que projetar seu aplicativo (ou ajustar um existente) para aceitar os novos recursos do tablet. As versões 3.0 original e 4.0 e posteriores dos tablets Android suportam os fragmentos, o elemento da interface de usuário conhecido como a barra de ação e o tema holográfico (chamado Holo). Os usuários esperam que você adicione essas novas ferramentas ao seu arsenal quando você atualiza seu aplicativo para funcionar no tablet.

Ajustando o Aplicativo Task Reminder para os Tablets

Para ajudar a implementar as diferenças, você usa algumas técnicas para atualizar o aplicativo Task Reminder para que ele possa funcionar nos tablets e nos telefones.

Use estas estratégias sempre que projetar um aplicativo Android porque é provável que a maioria de seus aplicativos tenha como alvo os usuários de ambos os tipos de dispositivos.

Antecipando o tamanho da tela com um layout de fluxo

Vá no embalo quando estiver projetando seu layout para se ajustar a vários tamanhos de tela. Um layout de fluxo evita muito esforço e frustração para o designer e o usuário.

Se você estiver familiarizado com o desenvolvimento iOS, sabe que tem apenas dois tamanhos de tela com os quais se preocupar: iPhone e iPad. Cada tamanho requer imagens com baixa e alta resoluções, mas isso é bem fácil de lidar. Projete para o iPhone primeiro, *e então* para o iPad, depois, forneça as imagens com baixa e alta resoluções nas respectivas versões e terá terminado.

O Android não é tão simples de projetar. Os layouts no Android precisam "fluir" — ou seja, redimensionar-se e reorganizar-se — para que possam aceitar as pequenas diferenças (e algumas vezes, grandes diferenças) na largura e na altura dos dispositivos dos usuários. Onde o iOS tem apenas dois tamanhos, o Android tem dezenas ou centenas.

É parecido com projetar para os sites web — quando você está construindo um site web, não pode supor que todos os usuários irão vê-lo em janelas de navegador com exatamente o mesmo tamanho (800x600 pixels). Os usuários podem ver o site a partir de janelas maiores (ou algumas vezes, menores); seu design deve ser flexível o bastante para fornecer uma boa experiência para uma boa variedade de tamanhos. Projetar para o Android tem a mesma exigência.

Portanto, como você faz esta pequena mágica? Para os iniciantes, não tente usar dimensões fixas (tais como `10px` ou `120dp`) em seus layouts. Ao contrário, prefira as dimensões *relativas*, tais como, `"wrap_content"` e `"fill_parent"`, o máximo possível. A ideia é conseguir um layout de *fluxo* que pode redimensionar-se para caber no dispositivo.

O seguinte código mostra um layout que faz suposições demais sobre o dispositivo no qual está:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="300px"
    android:layout_height="match_parent"
    android:lines="1"
    android:text="Occurrences of the word &apos;Internet&apos; in the
           Gettysburg Address: 0 (unverified)" />
```

O código tem vários problemas:

- Ele usa um `TextView` com tamanho fixo, ao invés de dimensões flexíveis, tais como, `wrap_content` e `fill_parent`.

- Usa pixels (px) para definir o tamanho, que não se dimensionam automaticamente nos diferentes dispositivos, como dp (pixels independentes do dispositivo) faria.
- Codifica especificamente o número de linhas em `TextView` para 1 e não informa ao Android o que fazer com qualquer excesso.
- Não usa `ScrollView`, portanto, se seu layout for maior que a tela do dispositivo, não haverá nenhum modo de ver as views fora da tela.

Geralmente falando, muitos de seus layouts devem ser colocados em um único `ScrollView` para lidar com o excesso inesperado que extrapola a parte inferior da tela. Exceções incluem os layouts que já lidam com a rolagem, tais como, `ListView`, que não deve ser colocada em um `ScrollView`. O aplicativo Task Reminder precisa apenas de um único `TextView`, porém, os layouts mais complicados devem considerá-los.

Embora "`fill_parent`" tenha sido renomeado para "`match_parent`" no Android 2.2, os dispositivos mais antigos não suportam "`match_parent`". Porém, contanto que sua `minSdkVersion` seja definida para um valor inferior a 8, você deverá usar "`fill_parent`", ao invés de "`match_parent`".

A Figura 16-1 mostra `TextView` na Listagem 16-1. Ele corta abruptamente a frase no meio do texto por causa do tamanho fixo. Se o desenvolvedor tivesse usado um layout de fluxo, o texto não teria sido cortado.

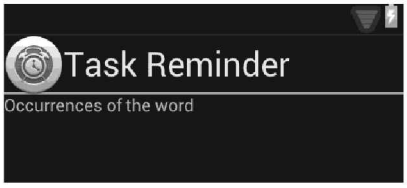


Figura 16-1: Um layout sem fluxo.

Corrigir este exemplo em particular é fácil mudando a largura de `TextView` e substituindo `android:lines="1"` por `android:maxLines="3"`:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:maxLines="3"
    android:text="Occurrences of the word &apos;Internet&apos; in the
            Gettysburg Address: 0 (unverified)" />
```

Quando você estiver projetando seus layouts, sempre considere o tamanho máximo de cada item nele. O conteúdo do aplicativo pode ocupar mais espaço do que você espera e é importante antecipar estas situações e planejá-las, ao invés de acabar com um aplicativo sem uma boa aparência.

Adicionando mais fragmentos

Se você for como muitos desenvolvedores, poderá esperar que a Google — como a fonte de todo conhecimento — saiba *tudo*. Mas, algo que as pessoas na Google não sabiam quando começaram a construir o Android era como ficariam populares os tablets em cinco anos. Mas os tablets realmente ficaram populares e trouxeram com eles todo aquele espaço na tela para ser preenchido.

O Android 3.0 Honeycomb introduziu fragmentos para ajudá-lo a lidar com este terreno livre. A ideia básica é que uma atividade típica do telefone fique centrada em um, dois ou três grupos distintos de itens reutilizáveis na tela. Coloque cada um desses grupos em seu próprio fragmento e ficará fácil reutilizar em várias atividades.

Você fará exatamente isto no Capítulo 17 — irá adicionar `ReminderEditFragment` e `ReminderListFragment` de suas duas atividades do telefone em uma atividade nova que os usuários de tablets gostarão.

A Figura 16-2 mostra como os fragmentos Reminder (Lembrete) apresentam seus layouts em um telefone e um tablet.

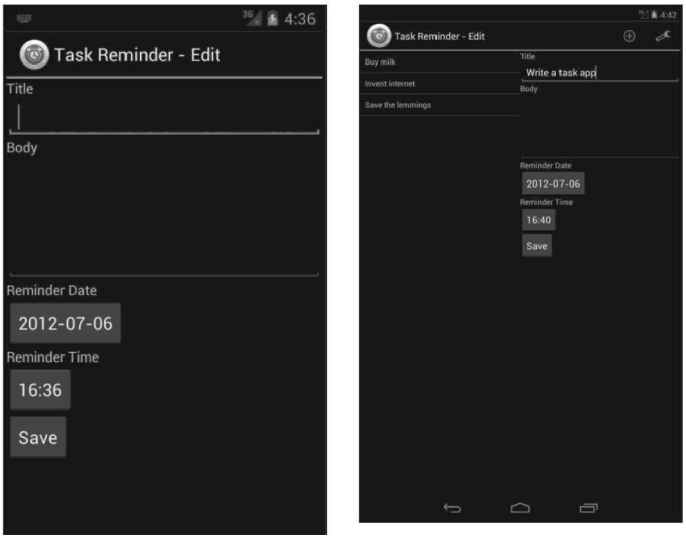


Figura 16-2: Os fragmentos podem lidar com uma atividade em um telefone (esquerda) ou duas atividades em um tablet (direta).

Sem fragmentos, você teria que reinventar a roda sempre que desejasse criar uma atividade que exibisse uma lista de tarefas. Usando fragmentos, você escreve o código uma vez e pode reutilizá-lo quantas vezes quiser.

Criando diferentes layouts para diferentes dispositivos

O fragmento é um recurso útil para o projetista, mas como você decompõe e compõe os fragmentos para mostrar a experiência correta para o dispositivo certo? O espaço de tela relativamente grande do tablet (comparado com um telefone) pode mostrar um ou mais fragmentos em uma única atividade.

Você pode usar um layout contendo uma única atividade para seu telefone e outro layout contendo diversos fragmentos para seu tablet. Por exemplo, eis o layout `ReminderListActivity` para o tamanho do telefone no aplicativo Task Reminder:

```
<?xml version="1.0" encoding="utf-8"?>
    <fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:name="com.dummies.android.taskreminder.ReminderListFragment"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
      />
```

E eis como você pode modificar o código usando dois fragmentos para criar um layout com duas colunas em um tablet:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:orientation="horizontal"
      android:layout_height="fill_parent">

    <fragment
          android:id="@+id/list_fragment"
          android:name="com.dummies.android.taskreminder.ReminderListFragment"
          android:layout_width="0dp"
          android:layout_weight="1"
          android:layout_height="fill_parent"
      />

    <fragment
          android:id="@+id/edit_fragment"
          android:name="com.dummies.android.taskreminder.ReminderEditFragment"
          android:layout_width="0dp"
          android:layout_weight="1"
          android:layout_height="fill_parent"
      />

</LinearLayout>
```

Você pode encontrar uma explicação sobre como este código funciona no Capítulo 17.

Usando a barra de ação

As interfaces de usuário mudam continuamente e o Android não é nenhuma exceção. Um bom exemplo é a barra de ação — padrão em

todos os dispositivos Android 3.0 e 4.0. Se você estiver visando os tablets, definitivamente usará este novo elemento.

Antes do Android 3.0, todos os telefones Android tinham um botão Menu dedicado (para saber mais sobre o botão Menu, veja o Capítulo 10). Esse botão não existe nos dispositivos Android 3.0 e posterior. Os itens no menu estão na barra de ação.

A barra de ação facilita muito o acesso a estas funções (por exemplo, adicionar ou apagar uma tarefa ou acessar as definições de um aplicativo). As mais importantes estão sempre na tela; as menos importantes ficam escondidas em um submenu, como mostrado na Figura 16-3.

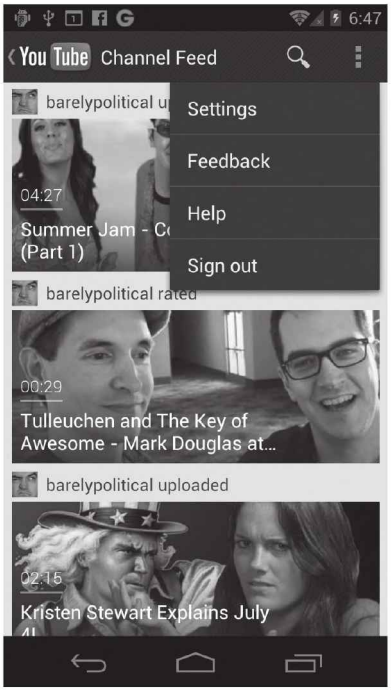


Figura 16-3: A barra de ação, pronta para a ação, com vários itens no submenu adicional.

Como desenvolvedor, você tem que escolher quais ações são as mais importantes e o Android tentará mantê-las na tela. Você pode adicionar três ações diferentes aos seus itens de menu:

- `android:showAsAction="ifRoom"`: Este XML é para os itens pouco importantes que você prefere ter na barra de ação, caso tenha espaço, mas o Android pode mover para o menu adicional.
- `android:showAsAction="always"`: Este XML é para os primeiros dois itens que você sabe que deseja sempre ter na tela e para os itens que você sempre deseja mostrar na barra de ação.
- `android:showAsAction="never"`: Use este XML para os itens que você pode posicionar com segurança no menu adicional.

É tentador listar todas as suas ações como `android:showAsAction="ifRoom"` ou `"always"`, mas não encha a barra de ação com ações demais. Considere colocar muitas ações, tais como, `"Settings"`, no submenu adicional usando `"never"`.

A melhor parte da barra de ação é que você não tem que fazer nada para adicioná-la ao seu aplicativo. Todos os menus se transformam, como mágica, nos itens da barra de ação quando são executados nos dispositivos Android 3.x ou posterior.

Usando a Biblioteca de Suporte e a ActionBarSherlock

Naturalmente, nem todos compram rapidamente itens de última geração. Quando este livro foi escrito, um grande número de dispositivos ainda estava sendo executado nas versões mais antigas do Android (anteriores a 4.0). Você deve levar isso em conta.

Conheça seu público! Visite `http://developer.android.com/about/dashboards` (conteúdo em inglês) para ver as últimas estatísticas sobre quais versões do Android seus usuários provavelmente estão executando. Quando este livro foi escrito, o Android 2.3.3 (Gingerbread) tinha a maioria dos usuários, mas sua liderança está diminuindo.

Você deseja usar os fragmentos e a barra de ação, mas se a maioria dos dispositivos de seus usuários estiver executando o Android 2.x, esses dispositivos mais antigos não suportarão os recursos modernos. O que um desenvolvedor deve fazer?

Felizmente, estas ferramentas podem ajudar:

- **Biblioteca de Suporte Android (da Google):** Traz os fragmentos para seus aplicativos anteriores a 4.0.
- **ActionBarSherlock de Jake Wharton:** Permite usar a barra de ação para os aplicativos anteriores a 4.0. Vá para `http://actionbarsherlock.com` (conteúdo em inglês) para obter mais detalhes.

Capítulo 17

Portando Seu Aplicativo para os Tablets Android

Neste Capítulo

Fazendo o download de um emulador tablet

Usando atividades

Criando fragmentos separados

Seu aplicativo de telefone está pronto para conquistar o tablet e a situação está ficando animada. Este capítulo mostra como modificar o aplicativo Task Reminder (desenvolvido na Parte lll) para trabalhar em um tablet Android.

Configurando um Emulador Tablet

Primeiro, as coisas mais importantes — você precisa de um tablet no qual testar seu aplicativo. Se você já tem um tablet, estará bem; se não, precisará de um emulador para simular um dispositivo Android em seu computador. A Google os chama de Dispositivos Virtuais Android (AVDs). Siga estas etapas para obter o AVD Nexus 7 da Google:

1. **Clique no ícone Android AVD Manager (Gerenciador AVD Android) na barra de ferramentas do Eclipse.**

2. **Clique em New (Novo).**

   A caixa de diálogo Create New Android Virtual Device (Criar Novo Dispositivo Virtual Android) será aberta, como mostrado na Figura 17-1.

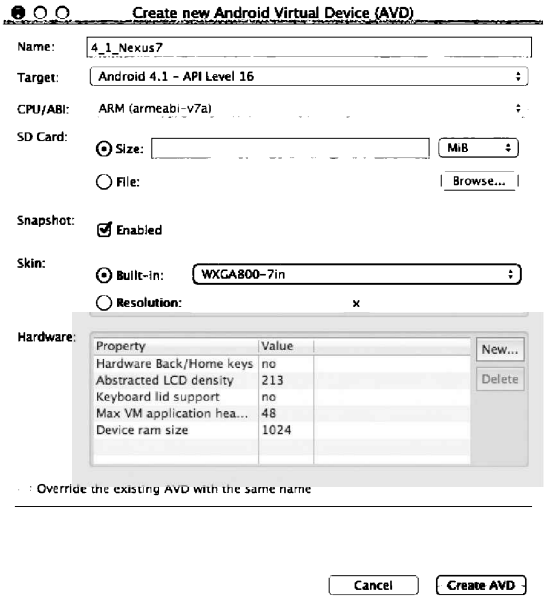


Figura 17-1: Crie um emulador tablet Nexus 7.

3. **Escolha Android 4.1 – APl Level 16 (Nível 16) na lista suspensa Target (Destino).**

   O Nexus 7 usa o Android 4.1.

4. **Na seção Skin (Aparência), selecione o botão de rádio Built-ln (Predefinida) e escolha WXGA800-7 no menu suspenso.**

5. **Clique no botão Create AVD (Criar AVD).**

6. **Escolha o AVD que você acabou de criar na lista de AVDs e clique no botão Start (Iniciar) para inicializá-lo.**

Atualizando o arquivo AndroidManifest

Você precisa informar ao Android que seu aplicativo funciona nas telas dos tablets com tamanhos grande e extra grande. Sem isso, seu aplicativo será executado no modo de Compatibilidade como se estivesse em uma tela menor e o Android expandirá seu aplicativo para que ele preencha a tela. Ele poderá parecer "recortado" e feio.

Edite o arquivo `AndroidManifest.xml` para adicionar as seguintes linhas ao elemento `supports-screens` dentro do elemento `<manifest>` antes do elemento `<application>`:

```
<supports-screens
      android:largeScreens="true"
      android:normalScreens="true"
      android:smallScreens="true"
      android:xlargeScreens="true" />
```

Você pode achar que a definição de um desses atributos impede incorretamente que seu aplicativo seja baixado por dispositivo com esse tamanho, mas este não é necessariamente o caso. Se `xlargeScreens="false"`, por exemplo, seu aplicativo ainda será baixado nos dispositivos com tamanho extra grande, mas será executado no modo de Compatibilidade e o Android irá aumentá-lo para caber na tela. Por outro lado, embora seja fácil expandir um aplicativo, é difícil diminui-lo, portanto, se `smallScreens="false"`, seu aplicativo não será baixado pelos dispositivos com tela pequena.

Programando Atividades para os Tablets

Depois de atualizar seu arquivo `AndroidManifest`, a próxima etapa será começar a codificar as novas atividades que são únicas para versão tablet de seu aplicativo.

Criando ReminderListAndEditorActivity

A versão tablet de seu aplicativo precisa de uma nova atividade principal. A que funciona para os telefones não funciona com os tablets porque você deseja manter essa atividade para que seu aplicativo continue a funcionar nos telefones. Crie um arquivo `ReminderListAndEditorActivity.java` e adicione este código:

```
package com.dummies.android.taskreminder;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;            →4

public class ReminderListAndEditorActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.reminder_list_and_editor);        →10
    }
}
```

Eis como o código funciona:

→ 4 Esta linha assegura que você usará `FragmentActivity` a partir da biblioteca de suporte, ao invés de uma predefinida no Android 3.x e posterior. Do contrário, seu código não funcionará nas versões anteriores do Android.

→ 10 O layout `R.layout.reminder_list_and_editor`, que define o layout desta atividade, não existe ainda, mas você irá criá-lo posteriormente neste capítulo.

Adicione a nova atividade ao arquivo `AndroidManifest.xml` abaixo de `ReminderEditActivity` que já existe para os telefones:

```
<activity android:name=".ReminderListAndEditorActivity" android:label="@
string/edit_reminder_title"/>
```

Forneça um título à atividade. Abra `strings.xml` e adicione uma string para `edit_reminder_title`, tal como, `Task Reminder - Edit`.

Escolhendo a atividade certa

O aplicativo Task Reminder agora tem duas atividades principais diferentes — uma para telefones e outra para tablets. Quando o usuário inicializar seu aplicativo, qual o Android escolherá?

No momento, o Android escolhe a atividade do telefone porque é a única associada ao filtro da intenção `android.intent.action.MAIN`. Como não há nenhum modo do aplicativo escolher automaticamente a atividade correta para você, você precisará detectá-la manualmente quando o aplicativo iniciar, e então mudar para versão tablet, se necessário.

Adicione o seguinte método à sua classe ReminderListActivity:

```
private Boolean isTablet() {
    int sizeMask = getResources().getConfiguration().screenLayout &
              Configuration.SCREENLAYOUT_SIZE_MASK;
    boolean large = (sizeMask == Configuration.SCREENLAYOUT_SIZE_LARGE);
    boolean xlarge = (sizeMask == 4);
    return large || xlarge;
}
```

Este código detecta se o dispositivo tem uma tela no formato grande ou extra grande. Se tiver, retornará true.

Por que `sizeMask == SCREENLAYOUT_SIZE_LARGE` para o booleano `large`, mas `sizeMask == 4` para o booleano `xlarge`? Nas versões posteriores do Android, `SCREENLAYOUT_SIZE_XLARGE` é igual a 4, portanto, você pode achar que poderia substituir 4 por `SCREENLAYOUT_SIZE_XLARGE`. Contudo, `SCREENLAYOUT_SIZE_XLARGE` não apareceu no Android até a API nível 9, portanto, se você tentar usar esta constante em um dispositivo mais antigo, ele irá paralisar. Como o aplicativo Task Reminder suporta dispositivos mais antigos que a API nível 9, você não poderá usar a constante `SCREENLAYOUT_SIZE_XLARGE` diretamente. Veja `http://d.android.com/reference/android/ content/res/Configuration.html` para obter mais informações sobre `SCREENLAYOUT_SIZE_XLARGE`.

Começando com o Android 3.2, `SCREENLAYOUT_SIZE_MASK` ficou obsoleta em favor de um novo sistema de qualificadores de tamanho do dispositivo. O sistema é muito mais confiável para escolher se é para mostrar um layout de telefone ou tablet, mas como não está disponível antes da versão 3.2 (mesmo com a biblioteca de suporte), está além do escopo deste livro. Visite `http://developer.android.com/guide/practices/screens_support.html` (conteúdo em inglês) para obter detalhes sobre os novos qualificadores de tamanho.

Em seu método `onCreate()`, adicione o código mostrado em negrito antes da chamada para `setContentView()`:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Troque para a atividade Tablet e termine esta se estiver em um tablet.
    if (isTablet()) {
        startActivity(new Intent(this, ReminderListAndEditorActivity.class));
        finish();
        return;
    }
    setContentView(R.layout.reminder_list);
}
```

A ideia é que se este código for executado em um tablet, `ReminderListActivity` sairá imediatamente e `ReminderListAndEditorActivity` será iniciado. Assim, se um usuário abrir seu aplicativo em um telefone, o aplicativo funcionará adequadamente para o tamanho do telefone. Mas, se um usuário abrir seu aplicativo em um tablet, ele também será executado adequadamente. Um pequeno *overhead* está envolvido ao fazer este tipo de troca, mas passará despercebido pelos seus usuários.

Criando o layout da atividade

Depois de ter a atividade, você precisará criar seu layout.

Neste ponto, você já tem a ideia básica de como será este layout: um fragmento de lista e um fragmento de edição colocados lado a lado.

Crie um arquivo `reminder_list_and_editor.xml` em seu diretório res/layout e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >                    →5

    <fragment
        android:id="@+id/list_fragment"                   →8
        android:name="com.dummies.android.taskreminder.ReminderListFragment"→9
        android:layout_width="0dp"                        →10
        android:layout_height="fill_parent"               →11
        android:layout_weight="1" />                      →12

    <FrameLayout
        android:id="@+id/edit_container"                  →15
        android:layout_width="0dp"
        android:layout_height="fill_parent"
        android:layout_weight="1" />                      →18

</LinearLayout>
```

Eis como funciona o código:

→ 5 A linha integra seus dois fragmentos em um LinearLayout horizontal em tela cheia para que apareçam lado a lado.

→ 8-9 Estas linhas designam o primeiro fragmento como o fragmento de lista, chamado "`@+id/list_fragment`" e especificam seu nome de classe completo como `com.dummies.android.taskreminder.ReminderListFragment`.

→ **10** Esta linha fornece o layout com uma largura `0dp`. Veja a linha 12 para descobrir por que a largura zero é vantajosa.

→ **11** Esta linha informa ao fragmento de lista para ocupar metade da tela na horizontal e a tela completa na vertical.

→ **12-18** `LinearLayouts` suporta o parâmetro especial `"layout_weight"`, que pode ser usado para dividir de modo flexível a tela entre duas ou mais exibições. Para usá-lo, defina as larguras ou as alturas dos filhos de `LinearLayout` para `0dp`, e então defina um peso para cada um. `LinearLayout` soma todos os valores de peso e atribui a cada filho a proporção que o peso do filho individual representa.

A linha 12 designa que as exibições `list_fragment` e `edit_container` tenham um peso 1 para que cada exibição ocupe metade da tela.

Você também pode fazer com que `list_container` tenha um valor 1 e `edit_container` tenha um valor 2 e neste caso, `list_fragment` ocupará um terço da tela e `edit_container` ficará com os dois terços restantes.

Note que esta estratégia funciona apenas para os `LinearLayouts`, embora possa ser bem útil.

→ **15** Esta linha designa um espaço reservado (placeholder — `"@+id/edit_container"`) para `ReminderEditFragment`. Um espaço reservado é preferível porque, mesmo que a exibição de lista esteja sempre na tela, a exibição de edição poderá ir e vir, dependendo se um usuário está editando um item ou não. Se um item não estiver sendo editado, seria confuso para o usuário ver um fragmento de edição vazio na tela.

Execute seu aplicativo em ambos os emuladores, telefone e tablet. Embora a nova `ReminderListAndEditorActivity` apareça no emulador tablet, não estará no emulador telefone.

Trabalhando com Fragmentos nos Aplicativos do Tablet

Você criou uma nova atividade para seu aplicativo tablet e agora tem que adicionar fragmentos a ele. Não há necessidade de escrever novos fragmentos para usar com a atividade do tablet. Porém, você precisa fazer algumas mudanças em seus fragmentos para assegurar que eles funcionarão corretamente neste novo ambiente.

Comunicando-se com fragmentos

Permitir que os fragmentos se comuniquem usando atividades como intermediárias é considerada uma boa prática ao usar os fragmentos. Visite `http://developer.android.com/guide/components/fragments.html` (conteúdo em inglês) para obter mais detalhes sobre este padrão importante. Sempre que você precisar interagir com outro fragmento, deverá usar um método na atividade do fragmento, ao invés de acessar outro fragmento diretamente. A única situação em que faz sentido acessar um fragmento a partir de outro é quando você sabe que não precisará reutilizar seu fragmento em outra atividade. Contudo, a vida é sempre cheia de surpresas — você sempre deve escrever fragmentos supondo que irá reutilizá-los, ao invés de codificá-los especificamente entre si.

Comunicando-se entre os fragmentos

Se você tentar adicionar uma tarefa com seu aplicativo tablet, ele não mostrará o fragmento de edição perto do fragmento de lista como o esperado, mas abrirá uma nova atividade para adicionar uma tarefa.

O motivo para seu aplicativo abrir o fragmento de edição em uma nova atividade, ao invés de próximo do fragmento de lista, é que ele ainda está fazendo exatamente o que foi informado. Ele ainda está executando o antigo comportamento do telefone, que é iniciar uma nova atividade para cada fragmento.

O código que faz isso é esta linha em `ReminderListFragment`:

```
startActivity(new Intent(this, ReminderEditActivity.class).putExtra(
ReminderProvider.COLUMN_ROWID, id));
```

Isso deve ser fácil de alterar, certo? Não tão rapidamente. Se você mudar essa linha para fazê-la funcionar para os tablets, "quebrará" seu aplicativo para telefones. O problema é que você deseja um comportamento (o existente) para os telefones e outro comportamento para os tablets.

Para resolver este problema, você cria um método abstrato chamado `editReminder()` que faz uma coisa para os telefones e outra para os tablets. Então, você substitui a chamada existente para `startActivity()` por esse método atualizado.

Não seja tão rápido ao colocar o novo método `editReminder()` em `ReminderListFragment`. O fragmento não sabe se está sendo executado em um telefone ou um tablet. Ele tem um método que pode chamar — `editReminder()` — que permite ao usuário editar um lembrete.

O segredo é perceber que como você precisa de uma versão de `editReminder()` para telefones e outra para tablets, precisa colocar `editReminder()` em *dois* lugares — um que é executado nos telefones e outro que é executado nos tablets. Onde uma classe é executada nos telefones e outra é executada nos tablets? Na atividade, claro. Portanto, você colocará `editReminder()` nas classes `Activity` do telefone e do tablet.

Para criar o callback `editReminder()` para seu fragmento chamar, siga estas etapas:

1. **Crie uma nova interface `OnEditReminder.java` e coloque o método `editReminder()` nela, assim:**

```
package com.dummies.android.taskreminder;

public interface OnEditReminder {
   public void editReminder(long id);
}
```

2. **Implemente este método em `ReminderListActivity` para os telefones.**

   Modifique `ReminderListActivity` adicionando as partes em negrito:

```
package com.dummies.android.taskreminder;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class ReminderListActivity extends FragmentActivity implements
OnEditReminder {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.reminder_list);

      // Switch to tablet activity and finish this one if user is on a
         tablet.
      if (isTablet()) {
        startActivity(new Intent(this, ReminderListAndEditorActivity.
        class));
        finish();
        return;
      }
   }

   @Override
   public void editReminder(long id) {
      startActivity(new Intent(this, ReminderEditActivity.class).
         putExtra(
            ReminderProvider.COLUMN_ROWID, id));
   }
}
```

A chamada `editReminder()` é idêntica, linha por linha, àquilo que você tinha em `ReminderListFragment`, exceto que, agora, está em `ReminderListActivity`.

3. **Chame este método a partir do fragmento:**

   Visite `RemiderListFragment` e modifique os métodos `onOptionsItemSelected()` e `onListItemClick()` como a seguir:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    caseR.id.menu_insert:
        ((OnEditReminder) getActivity()).editReminder(0);
        return true;
    caseR.id.menu_settings:
startActivity(new Intent(getActivity(), TaskPreferences.class));
return true;
      }

return super.onOptionsItemSelected(item);
    }

@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    super.onListItemClick(l, v, position, id);
    ((OnEditReminder) getActivity()).editReminder(id);
   }
```

Agora, ao invés de chamar `startActivity()` diretamente em cada método, o aplicativo chamará `getActivity()` para obter a atividade, fazendo sua conversão para um `OnEditReminder`, e então chamando `editReminder()`.

Casting (conversão de tipos) é normalmente visto com desagrado no Java porque, em geral, não é seguro. Contudo, é seguro fazer a conversão do resultado de `getActivity()` para um `OnEditReminder` porque `ReminderListFragment` sempre estará em `ReminderListActivity` ou `ReminderListAndEditorActivity` e ambos implementam `OnEditReminder`. Se você quiser adicionar o fragmento a outra atividade, também certifique-se de que ele implementa `OnEditReminder`.

4. **Implemente a mesma interface `OnEditReminder` em `ReminderListAndEditorActivity` para os tablets.**

   Adicione o código em negrito a `ReminderListAndEditorActivity`.

```
public class ReminderListAndEditorActivity extends FragmentActivity
            implements OnEditReminder {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.reminder_list_and_editor);
    }

    @Override
    public void editReminder(long id) {
        // TBD
    }
}
```

Verifique a próxima seção para ver exatamente como implementar `editReminder()` em seu aplicativo tablet.

Antes de fazer isso, você precisa conhecer mais uma interação sutil entre os fragmentos. Observe `OnClickListener` para `mConfirmButton` em `ReminderEditFragment` e verá esta linha:

```
getActivity().finish();
```

Nada bom. Chamar `getActivity().finish()` a partir do aplicativo telefone retorna o usuário para a atividade de lista, mas chamá-lo a partir do aplicativo tablet fecha o aplicativo. Claramente, você deseja remover do aplicativo tablet o fragmento de edição, mas não terminar a atividade inteira.

Para corrigir este problema, siga estas etapas:

1. **Crie uma nova interface chamada `OnFinishEditor` e faça com que fique assim:**

   ```
   package com.dummies.android.taskreminder;

   public interface OnFinishEditor {
       public void finishEditor();
   }
   ```

2. **Modifique `ReminderEditFragment` para chamar esta interface, ao invés de chamar `finish()` diretamente.**

   Substitua ambas as instâncias de `getActivity().finish()` pelo seguinte (deve haver duas):

   ```
   ((OnFinishEditor) getActivity()).finishEditor();
   ```

3. **Modifique as duas atividades de** `ReminderEditFragment` **para adicionar esta interface e implemente o método** `finishEditor()`**.**

   Adicione o seguinte a `ReminderEditActivity`:

```
public class ReminderEditActivity extends FragmentActivity implements
                OnFinishEditor {

    @Override
    public void finishEditor() {
        finish();
    }
}
```

Este é o mesmo código que costumava ser chamado em `OnClickListener` e `onLoadFinished()`.

4. **Adicione o seguinte código a** `ReminderListAndEditorActivity`**:**

```
public class ReminderListAndEditorActivity extends FragmentActivity
                implements
            OnEditReminder, OnFinishEditor {

    @Override
    public void finishEditor() {
        FragmentManager fragmentManager = getSupportFragmentManager();
                →6
        FragmentTransaction transaction = fragmentManager.
                beginTransaction(); →7
        Fragment previousFragment = fragmentManager
                .findFragmentByTag(ReminderEditFragment.DEFAULT_EDIT_
                FRAGMENT_TAG); →9
      transaction.remove(previousFragment); →10
      transaction.commit(); →11
    }
}
```

Como antes, você está usando `FragmentManager` e `FragmentTransaction` para gerenciar a inclusão e a remoção dos fragmentos da atividade. Eis o que o código faz:

→ 6 Solicita `FragmentManager` a `FragmentActivity`.

→ 7 Inicia `FragmentTransaction` chamando `beginTransaction()`.

→ 9 Pede a `FragmentManager` para encontrar o fragmento anterior chamando `DEFAULT_EDIT_FRAGMENT_TAG`, se houver. Este nome deve estar de acordo com o nome usado quando você adicionou inicialmente o fragmento em `editReminder()`.

→ **10** Remove o fragmento. Se `previousFragment` for `null`, esta linha não fará nada.

→ **11** Aceita a transação. Lembre-se que toda chamada para `beginTransaction()` deve terminar com uma chamada para `commit()`.

Adicionando transações de fragmento

Basicamente, `editReminder()` deve mostrar um fragmento de edição para a tarefa na qual o usuário tocou. Ou deve mostrar um fragmento de edição vazio se o usuário tocar no botão Add (Adicionar) na barra de edição para adicionar uma nota tarefa. Se um usuário tocar em várias tarefas seguidas, `editReminder()` deverá substituir o fragmento de edição existente por um novo, representando o último item.

Você já colocou os fragmentos nas atividades usando o XML em seu layout `reminder_list.xml`. Como você deseja adicionar e remover dinamicamente os fragmentos, desta vez usará o Java, ao invés do XML. O processo não é difícil; é apenas um pouco diferente do XML.

Quando você quiser adicionar ou remover os fragmentos no Java, precisará usar `FragmentManager` para iniciar uma `FragmentTransaction`. Então, você fará suas alterações e chamará `commit()` em `FragmentTransaction`, muito parecido como faria ao interagir com uma transação do banco de dados.

Dentro de `editReminder()`, adicione o seguinte código:

```
/**
* Defina o fragmento de edição, substituindo o fragmento existente, se já
* houver algum um no local.
*/
@Override
public void editReminder(long id) {
    ReminderEditFragment fragment = new ReminderEditFragment();     →7
    Bundle arguments = new Bundle();                    →8
    arguments.putLong(ReminderProvider.COLUMN_ROWID, id);        →9
    fragment.setArguments(arguments);                   →10

    FragmentTransaction transaction = getSupportFragmentManager()
            .beginTransaction();                    →13
    transaction.replace(R.id.edit_container, fragment,
            ReminderEditFragment.DEFAULT_EDIT_FRAGMENT_TAG);      →15
    transaction.addToBackStack(null);                   →16
    transaction.commit();                       →17
}
```

Eis como o código funciona:

→ **7** Cria um novo fragmento `ReminderEditFragment`.

Os fragmentos devem ter construtores sem argumentos. Todos os argumentos ficam em um objeto bundle.

→ **8-10** Informa ao fragmento qual tarefa está sendo editada. Um ID 0 indica que um novo fragmento está sendo criado.

→ **13** Obtém `FragmentManager` chamando `FragmentActivity.getSupportFragmentManager()`, e então chama `beginTransaction()` para iniciar uma nova transação do fragmento.

→ **15** Chama `FragmentTransaction.replace()` para substituir o fragmento existente pelo novo fragmento. A exibição `edit_container` informa onde colocar o fragmento, `fragment` informa qual fragmento usar e `ReminderEditFragment.DEFAULT_EDIT_FRAGMENT_TAG` mostra o nome do fragmento.

→ **16** Chama `addToBackStack()` e transmite `null` para o nome opcional do estado.

Esta linha requer uma pequena explicação. Pense no que acontece sempre que você inicia uma nova atividade — ela é adicionada à pilha de atividades e, se os usuários tocarem no botão Back (Voltar), serão retornados para a atividade anterior. Esta interação padrão é esperada pelos usuários para quase todas as atividades.

O comportamento padrão dos fragmentos, porém, é o oposto. Por padrão, quando você adiciona um fragmento à uma atividade, ele não entra na pilha de apoio. Portanto, um usuário que toca no botão Back sai da atividade, ao invés de remover o fragmento recém-adicionado. Isto pode não ser o que você deseja que aconteça. Se você quiser que o botão Back remova o fragmento, precisará chamar `addToBackStack()`.

Ao adicionar fragmentos dinamicamente, pense no que seus usuários provavelmente irão esperar que o botão Back faça. Neste caso, quando o usuário tocar em um item da lista para exibir um fragmento de edição, será razoável que ele espere ser capaz de tocar no botão Back para fechar o fragmento de edição.

→ **17** Toda chamada para `beginTransaction()` deve vir acompanhada de uma chamada para `commit()`. É onde o Android faz o trabalho de adicionar fragmentos ou removê-los de sua atividade.

Parabéns – agora, você deve ter uma versão totalmente implementada de seu aplicativo de telefone sendo executada em seu tablet.

Capítulo 18

Indo além do Google

Neste Capítulo

Descobrindo quais recursos não funcionam com o Kindle

Configurando e testando com um emulador

Fazendo upload de seu aplicativo para a loja Amazon

Para o Android, a Google pode ser a maior na área — mas não é a única. Como a Google torna todo lançamento do Android aberto para o público via Android Open Source Project (Projeto de Fonte Aberta Android), muitas empresas produzem suas próprias versões personalizadas do código-fonte Android.

Uma versão com a qual você pode estar familiarizado, a Amazon, escolheu o Android para rodar em seu tablet, o Kindle Fire.

O Kindle baseado no Android pode executar aplicativos com poucas ou nenhuma modificação. Porém, ele não tem acesso à Google Play Store, significando que se você quiser que os usuários Kindle Fire sejam capazes de fazer download de seu aplicativo, terá que publicá-lo na Amazon Appstore para Android. Neste capítulo, você descobrirá como portar seu aplicativo para o Kindle Fire, e então publicá-lo via Amazon.

DICA

Um motivo para querer portar para o Kindle Fire é atingir mais usuários. Mas, apenas você pode decidir se os usuários adicionais valerão a pena o esforço extra necessário. Faça seu trabalho de casa e leia as estatísticas relevantes sobre quantos usuários cada plataforma nova tem antes de se comprometer com o esforço.

Trabalhando com os recursos do Google

Como o Kindle File não é um "verdadeiro" dispositivo Android (ele não usa o código-fonte Android oficial da Google, mas sua versão personalizada), não tem acesso a nenhum serviço Google de fonte fechada que você já pode estar

usando. E mais, o próprio dispositivo pode não ter certos recursos com os quais você está acostumado:

- **Google Maps:** Se você estiver usando a biblioteca Google Maps para levar mapas para o aplicativo Android, não poderá usar essa biblioteca no Kindle Fire.
- **Serviços de localização:** Você não pode usar o Maps, nem tem acesso aos serviços de localização no Kindle Fire. Ele não tem GPS ou serviços de localização baseado em Wi-Fi, portanto, você não tem como informar onde o dispositivo está localizado fisicamente.
- **Compra de aplicativos na Google Play Store:** Se seu aplicativo usa o recurso de compra de aplicativos para permitir que os usuários comprem de dentro dele, não poderá usar a mesma API para seu aplicativo Kindle Fire. Felizmente, a Amazon tem uma versão do recurso de compra no aplicativo que você pode usar no Kindle Fire.
- **Câmera, microfone, Bluetooth, 3G, armazenamento externo:** O Kindle Fire não tem nenhum destes itens, portanto, se seu aplicativo usá-los, encontre um modo de solucionar a limitação ou considere não lançar seu aplicativo no Kindle Fire.
- **Honeycomb, Ice Cream Sandwich, Jelly Bean:** A Amazon usa a versão do código-fonte Android antes do Honeycomb ter sido lançado, portanto, o Kindle Fire não tem acesso a nenhum recurso nestas três versões. Em particular, você notará que o Kindle Fire tem uma aparência única que é diferente de qualquer outro tablet Android. Veja o Capítulo 1 para obter os recursos que vêm com estas três versões do Android.

Mesmo sem estes recursos e serviços, muitos aplicativos Android funcionam no Kindle Fire com pouca ou nenhuma modificação. Se isto incluir seu aplicativo, leia.

Configurando Seu Kindle Fire ou Emulador

Se você deseja desenvolver para o Kindle Fire, precisará do próprio Kindle Fire para testar seu aplicativo ou um emulador que possa agir como um substituto. Como o Kindle Fire é uma implementação especial do Android, você não pode usar o mesmo ADB utilizado com os outros dispositivos Android, a menos que faça algumas alterações na configuração.

Criando um emulador do tipo Kindle

Se você não tiver acesso a um Kindle Fire, precisará criar um emulador para ele. O processo para fazer isso é um pouco diferente do que é para um emulador Android normal porque o Kindle Fire executa sua própria versão do Android. Siga estas etapas:

1. **Instale o SDK do Android 2.3.3 (API Nível 10) usando o SDK Manager (Gerenciador SDK).**

   Verifique o Capítulo 2 para obter mais detalhes sobre como usar o SDK Manager.

2. **No SDK Manager, escolha Tools (Ferramentas) → Manage Add-on Sites (Gerenciar Sites Complementares).**

   A caixa de diálogo Add-on Sites (Sites Complementares) será aberta, como mostrado na Figura 18-1.

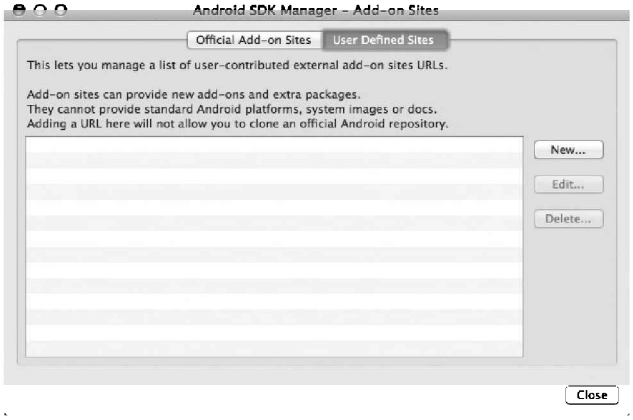


Figura 18-1: Adicione um SDK.

3. **Na guia User Defined Sites (Sites Definidos pelo Usuário), clique no botão New (Novo).**

4. **Insira a URL** `http://kindle-sdk.s3.amazonaws.com /addon.xml` **e clique em Close (Fechar).**

   A caixa de diálogo Android SDK Manager (Gerenciador SDK do Android) será aberta, como mostrado na Figura 18-2.

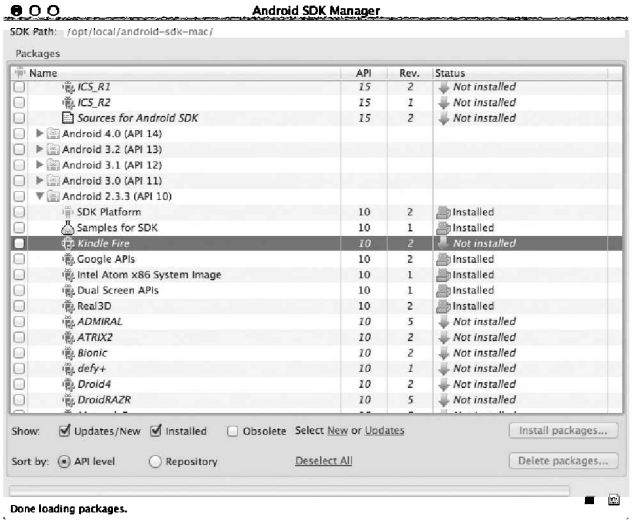


Figura 18-2: Instale o emulador Kindle Fire.

5. **Pagine para Android 2.3.3 (API 10), selecione Kindle Fire, e então clique no botão Install Packages (Instalar Pacotes).**

6. **Aceite o acordo de licença e clique em Install (Instalar) novamente.**

7. **Abra o AVD Manager (Gerenciador AVD) e clique no botão New (Novo).**

8. **Selecione Target Kindle Fire (Amazon) – API Level 10 (Visar Kindle Fire (Amazon) – API Nível 10).**

   A caixa de diálogo Create New Android Virtual Device (Criar Novo Dispositivo Virtual Android) será aberta, como mostrado na Figura 18-3.

9. **Digite um nome para seu emulador e clique em Create AVD (Criar AVD).**

10. **Selecione o AVD 2_3_3_Kindle_Fire que você acabou de criar, clique em Start (iniciar), e então clique no botão Launch (Inicializar) para executar seu novo emulador.**

    O emulador Kindle Fire agora está sendo executado em uma janela. Veja Figura 18-4.

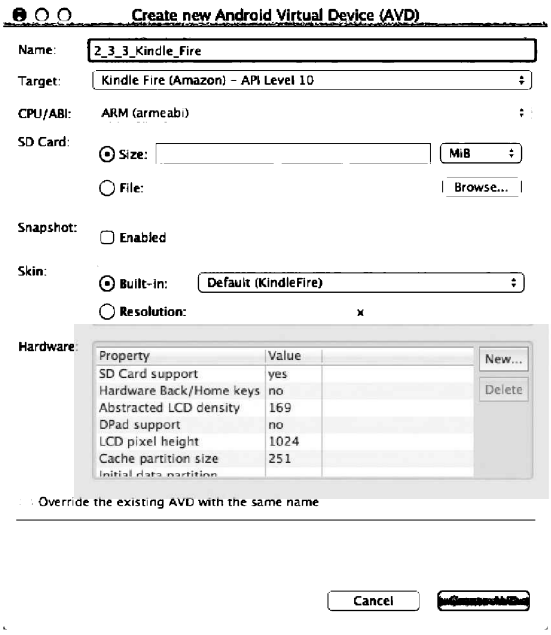


Figura 18-3: Nomeie seu emulador.

Figura 18-4: O emulador executando o Kindle Fire.

Agora, você pode executar seu aplicativo Android a partir do Eclipse em seu emulador Kindle Fire.

Configurando o ADB (Mac)

Se você estiver usando um dispositivo Kindle Fire real (ao contrário de um emulador), não poderá conectar um Kindle Fire direta e prontamente usando o ADB sem fazer algumas modificações primeiro. Siga estas etapas:

1. **Em seu Mac, edite o arquivo** `adb_usb.ini` **na pasta** `.android` **de seu diretório principal.**

   Adicione as seguintes linhas à parte inferior do arquivo:

   ```
   0x1949
   0x0006
   ```

2. **Conecte seu Kindle Fire e reinicie o ADB:**

   ```
   adb kill-server
   adb devices
   ```

Agora, você verá seu Kindle Fire listado na saída do comando adb devices.

Configurando o ADB (Windows)

Como em um Mac, você não pode conectar um dispositivo Kindle Fire direta e prontamente usando o ADB sem fazer algumas modificações primeiro.

1. **Em sua máquina Windows, edite o arquivo** `adb_usb.ini` **na pasta** `.android` **de seu diretório principal.**

   Adicione a seguinte linha à parte inferior do arquivo:

   ```
   0x1949
   ```

2. **Edite o arquivo** `android_winusb.inf`**.**

   Adicione as duas linhas a seguir às seções Google.NTx86 e Google.NTamd64:

   ```
   ;Kindle Fire
   %SingleAdbInterface% = USB_Install, USB\VID_1949&PID_0006
   %CompositeAdbInterface% = USB_Install, USB\VID_1949&PID_0006&MI_01
   ```

3. **Conecte seu Kindle Fire e, quando solicitado, escolha instalar o driver manualmente.**

4. **Navegue para o arquivo** `android_winusb.inf` **que acabou de editar e instale-o.**

5. **Reinicie o ADB executando os seguintes comandos em uma janela de comando:**

```
adb kill-server
adb devices
```

Agora, você verá seu Kindle Fire na saída do comando adb devices.

Publicando na Amazon Appstore para Android

Publicar na Amazon Appstore para Android é parecido com publicar na Google Play Store: Você cria uma conta, e então talvez precise pagar a taxa de desenvolvedor.

Diferentemente da Google Play Store, os aplicativos devem ser verificados na Amazon Appstore para Android, portanto, planeje com antecedência a data de envio do aplicativo pois levará alguns dias para que fique disponível na loja.

Siga estas etapas:

1. **Vá para** `https://developer.amazon.com/welcome.html` **(conteúdo em inglês).**

2. **Conecte-se usando seu login Amazon ou crie uma nova conta.**

3. **Clique no botão Accept and Continue (Aceitar e Continuar) para aceitar o acordo de licença do desenvolvedor.**

4. **Preencha as informações de monetização se você pretende cobrar por seu aplicativo ou por compras feitas através do aplicativo. Então, clique em Save (Salvar).**

   Veja a Figura 18-5.

5. **Clique no botão Add a New App (Adicionar um Novo Aplicativo).**

6. **Digite o título de seu aplicativo, o formato (telefone, tablet ou ambos) e informações de contato na guia General Information (Informações Gerais). Clique em Save (Salvar) quando terminar.**

   Sinta-se à vontade para preencher outros campos opcionais, tais como SKU, se for útil. Veja a Figura 18-6.

amazon mobile app distribution

Mike Burton — Android Development for Dummies (Sign Out)

Registration

1. Profile Information | 2. Developer License Agreement | 3. Royalty Payments

* indicates a required field.

Do you plan to monetize apps? * — No / Yes

Methods may include charging for the apps themselves or the sale of in-app items.

Where is your bank/financial institution? * — United States

Payment method — Electronic funds transfer (Issued in USD)

Bank/financial institution name *

Account holder *

Account type * — Individual Checking

Figura 18-5: Escolha se cobrará por seus aplicativos.

New App Submission

General Information | Availability & Pricing | Description | Images & Multimedia | Content Rating | Binary File

* indicates a required field.

App title *

App SKU

Form Factor * — -Select one-

Customer support contact — Use my default support information

Customer support email address *

Customer support phone *

Customer support website *

Cancel | Save

Figura 18-6: Forneça à Amazon os detalhes de seu aplicativo.

7. **Escolha em quais países tornará disponível seu aplicativo — e seu preço.**

8. **Escolha uma categoria para seu aplicativo, o idioma usado, o título e as descrições curta e longa. Então, clique em Save (Salvar).**

   Veja a Figura 18-7.

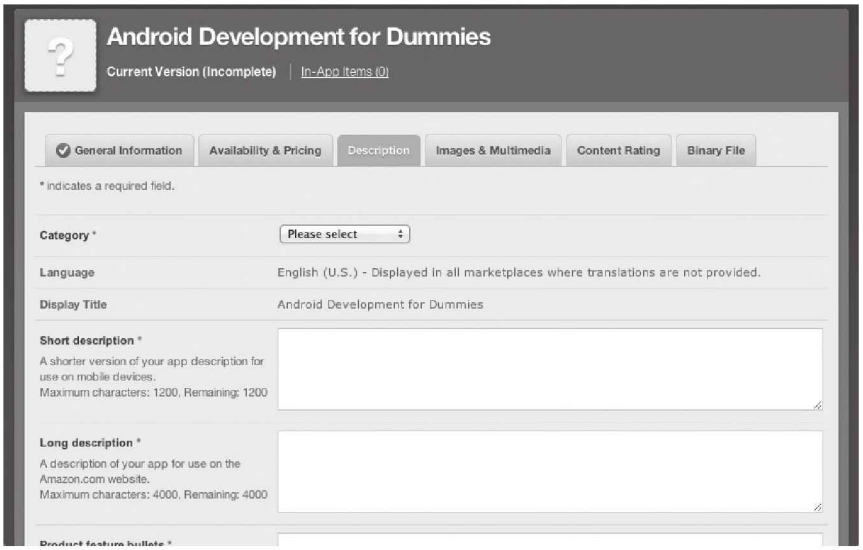


Figura 18-7: Selecione uma categoria e idioma para seu aplicativo.

9. **Faça upload de um ícone pequeno, um ícone grande, capturas de tela para incluir em sua descrição do aplicativo e uma captura de tela promocional. Então, clique em Save (Salvar).**

10. **Escolha a avaliação de conteúdo de seu aplicativo e as restrições de idade, clicando nos devidos botões de rádio. Então, clique em Save (Salvar).**

11. **Faça upload do código binário de seu aplicativo. Então, clique em Save (Salvar).**

    Veja o Capítulo 8 para obter mais informações sobre como construir e fazer upload do arquivo APK de seu aplicativo.

12. **Clique no botão Submit App (Enviar Aplicativo).**

O processo de revisão pode levar desde horas até dias ou semanas. Porém, quando seu aplicativo for lançado na loja de aplicativos, você poderá encontrá-lo na Amazon Appstore para Android junto com outros aplicativos, como mostrado na Figura 18-8.

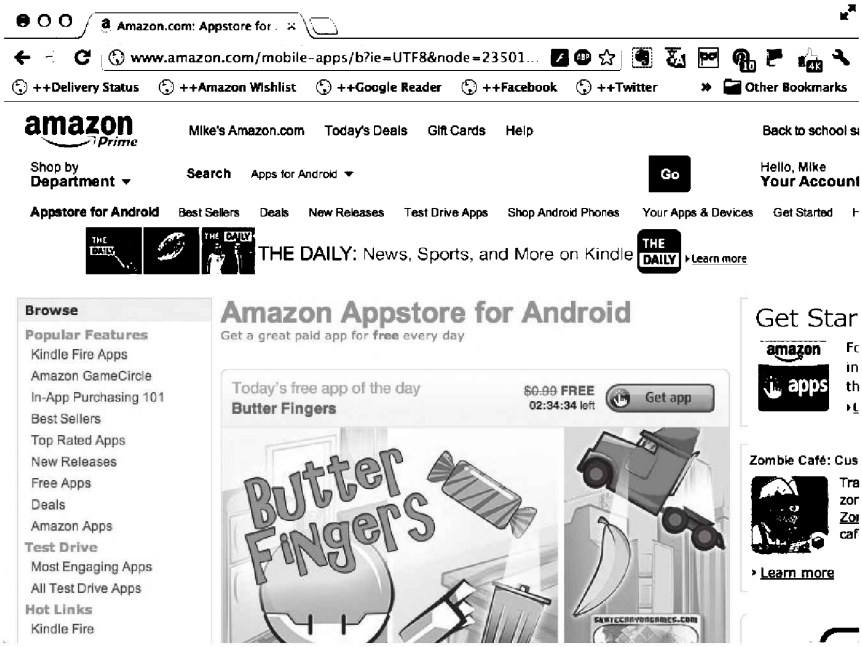


Figura 18-8: A Amazon Appstore para Android.

Parte V

A Parte dos Dez

"Esta é a conta de uma gravadora, Deb. Diga que você não contratou o Sr. Elton John para compor uma música para seu aplicativo."

Nesta parte...

A Parte V consiste em algumas das melhores novidades secretas do Android que você adquire apenas após ter estado nas trincheiras de desenvolvimento por algum tempo. O Capítulo 19 lista alguns dos melhores aplicativos de amostra que podem ajudá-lo na criação do próximo aplicativo bem-sucedido. Esses aplicativos variam desde aplicativos orientados a bancos de dados e jogos interativos até os aplicativos que interagem com as interfaces de programação de aplicativos (APIs) web de terceiros.

A Parte V fecha com uma lista das ferramentas e bibliotecas profissionais que podem ajudar a aperfeiçoar e melhorar a produtividade de seu processo de desenvolvimento de aplicativos e facilitar muita sua vida como desenvolvedor.

Capítulo 19

Dez Aplicativos de Amostra Gratuitos e SDKs

Quando você desenvolve aplicativos Android, pode encontrar vários obstáculos no código. Talvez você queira que um aplicativo comunique-se com uma API de terceiros que retorna o JSON ou execute uma detecção de colisão em um jogo. Em geral, você pode pesquisar a web para obter um código de amostra porque outra pessoa provavelmente já o escreveu. Então, tudo que tem a fazer é rever o código, alterá-lo segundo suas necessidades e continuar com o desenvolvimento.

Revisar um código de amostra aumenta seu conhecimento mesmo que você não precise do código em seu aplicativo. Na verdade, uma boa maneira de descobrir como programar para o Android é ver o código de amostra. Certamente, ele vem com o SDK do Android — em Demos API, por exemplo (veja Capítulo 2) — mas muito código real de aplicativos interessantes está disponível gratuitamente na web. Você pode encontrar na Internet muitos aplicativos com fonte aberta com alta qualidade para servir como exemplos, graças à natureza de fonte aberta do Android.

Grande parte dos dez excelentes aplicativos com fonte aberta e amostras neste capítulo são aplicativos Android reais que você pode instalar a partir da Google Play Store. Experimente um aplicativo em seu dispositivo, e então abra seu código-fonte para ver como é internamente.

Aplicativo Google I/O 2012

`http://code.google.com/p/iosched`

Todo ano, uma multidão de fãs do desenvolvimento Google invade o Moscone Center em São Francisco para uma conferência de vários dias para discutir sobre todas as coisas relacionadas à Google. O aplicativo oficial da conferência é escrito para os dispositivos Android e serve como um exemplo de como escrever bons aplicativos para a plataforma.

LOLcat Builder

http://code.google.com/p/apps-for-android

O LOLcat mostra como manipular imagens no Android — como tirar uma foto usando a câmera do dispositivo, adicionar títulos às imagens, e então salvar o arquivo resultante no cartão SD. Você pode ver como criar várias intenções, que permitem enviar a imagem no formato de mensagens multimídia (MMS) ou como um anexo de e-mail.

Amazed

http://code.google.com/p/apps-for-android

O divertido jogo Amazed demonstra o uso do acelerômetro embutido em um dispositivo para navegar uma bola de gude em 2D através de vários obstáculos em níveis de labirinto cada vez mais difíceis. Se você estiver interessado em aplicativos baseados em acelerômetros, revisar o código-fonte Amazed poderá ajudar muito. O aplicativo não só mostra como usar o acelerômetro, como também demonstra outros fundamentos de desenvolvimento de jogos, tais como a detecção de colisão e o princípio de loop do jogo.

API Demos

A pasta samples do SDK do Android contém o código-fonte para o aplicativo API Demos, que demonstra como usar várias APIs do Android através de pequenos exemplos curtos e funcionais. Você pode encontrar milhares de exemplos simples e diretos no código-fonte API Demos. Incorporar animação em seu projeto ou reproduzir um arquivo de áudio em seu aplicativo é fácil porque o API Demos fornece exemplos de ambos. Se você tem muitas ideias, mas não tem muito tempo, deverá instalar esse aplicativo demo em seu dispositivo e lidar com seus vários exemplos para ver exatamente o que eles podem fazer.

HoneycombGallery

Se você quiser que seu aplicativo seja bem executado em telefones e tablets, verifique o exemplo HoneycombGallery no SDK do Android porque ele pode evitar horas de depuração e posicionamento das exibições na interface do usuário. O aplicativo de amostra funcional mostra como suportar vários tamanhos de tela e resoluções, e demonstra o modo correto de usar os fragmentos em dispositivos com diferentes tamanhos. Você pode encontrar o código-fonte na pasta samples do SDK de seu Android.

K-9 Mail

`http://code.google.com/p/k9mail/`

O K-9 Mail é um cliente de e-mail popular para Android que costumava vir com o Android antes de se tornar um aplicativo separado. É um aplicativo de fonte aberta extraordinariamente cheio de recursos, com funcionalidades tais como pesquisa, transferência, sincronização, flag, assinaturas e mais.

Agit

`https://github.com/rtyley/agit`

O Git é um Sistema Distribuído de Controle de Versões (DVCS) de fonte aberta e popular. O Agit permite exibir, na palma de sua mão, todos os seus repositórios Git favoritos localizados no GitHub.com. O aplicativo demonstra como usar a API GitHub, assim como a estrutura RoboGuice.

SDK do Facebook para Android

`http://github.com/facebook/facebook-android-sdk`

Se você for ambicioso, poderá cuidar da tarefa de criar o próximo aplicativo popular Facebook, mesmo que não saiba por onde começar. Use o SDK do Facebook Android para integrar facilmente a funcionalidade Facebook em seu aplicativo — autenticar usuários, fazer solicitações da API e muito mais. Integre todas as coisas boas do Facebook sem esforço.

Replica Island

http://code.google.com/p/replicaisland

Talvez você queira criar um jogo de rolagem lateral em 2D para a plataforma Android, mas não tem ideia de como iniciar. Bem, é seu dia de sorte porque o jogo legal de rolagem lateral Replica Island apresenta nada menos do que o pequeno android verde (BugDroid) que os desenvolvedores conhecem e adoram. O Replica Island não é apenas um jogo gratuito e popular na Google Play Store, é também uma ferramenta de aprendizagem com fonte aberta para os desenvolvedores de jogos.

Notepad Tutorial

http://d.android.com/guide/tutorials/notepad/index.html

Se você estiver interessado em compreender os princípios básicos do SQLite sem focar em recursos como tarefas em segundo plano e outros conceitos técnicos, o Notepad Tutorial é para você. Embora seja simples em sua execução e uso, o código-fonte e o tutorial que vem com ele são úteis.

Capítulo 20

Dez Ferramentas para Simplificar Sua Vida no Desenvolvimento

Como desenvolvedor, você naturalmente constrói ferramentas para ser mais produtivo — por exemplo, para ajudar na comunicação assíncrona, análise XML e JSON, utilitários de data e hora, e muito mais. Antes de escrever milhares de estruturas ou classes auxiliares para lidar com tais itens, procure as ferramentas que já existem. Este capítulo lista as dez ferramentas e utilitários que podem simplificar sua vida no desenvolvimento, aumentando sua produtividade e assegurando que seu aplicativo tenha um bom nível.

droid-fu e ignition

```
http://github.com/kaeppler/droid-fu
https://github.com/kaeppler/ignition
```

A biblioteca de fonte aberta droid-fu tem muitos métodos para ajudar a aplicar um golpe de karatê para diminuir drasticamente seu tempo de desenvolvimento. A biblioteca é composta de classes utilitárias que fazem todo o trabalho pesado para você, tal como, lidar com as solicitações assíncronas em segundo plano, recuperar imagens na web e o mais surpreendente, melhorar o ciclo de vida do aplicativo.

Também verifique o ignition, um substituto para o droid-fu dos mesmos autores, mas atualmente em estágio alfa.

RoboGuice

```
http://code.google.com/p/roboguice
```

Não, o RoboGuice não é a bebida energética mais recente e direcionada para os desenvolvedores — é uma estrutura que usa a biblioteca Google Guice

para aperfeiçoar a injeção de dependência. A *injeção de dependência* lida com a inicialização de variáveis na hora certa, para que você não tenha que inicializá-las. Este conceito reduz a quantidade de código que você tem que escrever e facilita a manutenção de seu aplicativo.

Kit de Ferramentas do Tradutor

`http://translate.google.com/toolkit`

Se você quiser aumentar o número de pessoas que podem usar seu aplicativo, não haverá melhor modo do que traduzir seu aplicativo para outros idiomas. A solução é usar a Google para encontrar auxiliares para traduzir seu aplicativo para você. As traduções não serão tão claras quanto as traduções feitas por um falante nativo, mas serão um bom ponto de partida gastando muito pouco. Você pode considerar obter as traduções iniciais feitas pela Google, e então entrar em contato com sua comunidade de usuários para encontrar voluntários para editar as traduções para você ou usar um site web de terceiros, tal como o ODesk, para encontrar tradutores. Até os classificados podem ser um ótimo recurso!

Draw 9-patch

`http://developer.android.com/tools/help/draw9patch.html`

O utilitário Draw 9-patch permite criar facilmente imagens dimensionáveis para o Android. Você usa esse utilitário para incorporar instruções em uma imagem para informar ao sistema operacional onde estendê-las para que elas sejam exibidas o mais nítidas e claras possível, independentemente do tamanho ou da resolução da tela do dispositivo.

Hierarchy Viewer

`http://developer.android.com/tools/help/hierarchy-viewer.html`

Trabalhar com várias exibições dentro do arquivo de layout para criar uma interface de usuário nem sempre é um processo simples. O Hierarchy Viewer, localizado no diretório `tools` SDK do Android, permite ver exatamente como seus componentes ficam graficamente na tela. Este formato permite ver claramente os limites de um componente para que você possa determinar o que está acontecendo dentro do layout. O Hierarchy Viewer, a ferramenta definitiva para criar uma interface de usuário com pixels perfeitos, também permite aumentar a tela na exibição com pixels perfeitos para assegurar que as imagens e as IUs pareçam uniformes em todos os tamanhos de tela em todas as densidades.

UI/Application Exerciser Monkey

http://developer.android.com/tools/help/monkey.html

Não se preocupe: O UI/Application Exerciser Monkey não precisa ser alimentado com bananas para ficar feliz! Você usa o Exerciser Monkey para realizar testes de stress em seu aplicativo. Ele simula toques aleatórios, cliques e outros eventos do usuário para assegurar que o uso anormal não faça o aplicativo explodir. O Exerciser Monkey pode ser usado para testar os aplicativos em seu emulador ou em seu próprio dispositivo.

zipalign

http://developer.android.com/tools/help/zipalign.html

A ferramenta zipalign alinha todos os dados descompactados em um APK. Executar o zipalign minimiza o consumo de memória durante a execução. Se você estiver usando o ADT no Eclipse, seu aplicativo sempre será alinhado com o zip quando você exportar um aplicativo assinado, como demonstrado no Capítulo 8.

layoutopt

http://developer.android.com/tools/help/layoutopt.html

A ferramenta de linha de comando layoutopt analisa os layouts e informa qualquer problema ou ineficiência. É uma ferramenta útil para executar em todos os seus layouts e diretórios de recursos porque identifica os problemas que podem diminuir a velocidade de seu aplicativo e causar problemas posteriormente.

Git

http://git-scm.com

O Git — um sistema distribuído de controle de versão super-rápido, gratuito e de fonte aberta — gerencia os repositórios com rapidez e eficiência, facilitando o backup do trabalho. Não deixe uma paralisação do sistema arruinar seu dia por não ter um sistema de controle de versão para seu próximo grande aplicativo. O Git torna o trabalho com os branches simples e eficiente, e integra-se facilmente com seu fluxo de trabalho. Os plug-ins do Eclipse existem para ajudar a gerenciar seu repositório Git de dentro do IDE do Eclipse. Embora o Git seja distribuído, provavelmente você desejará um local remoto para colocar o repositório Git. Você pode obter um repositório

Git gratuito e privado no Projectlocker (`http://projectlocker.com`) ou no Unfuddle (`https://unfuddle.com`). Se seu código for de fonte aberta, você poderá criar repositórios gratuitos em Github.com.

Paint.NET e GIMP

```
www.getpaint.net
www.gimp.org
```

Você irá trabalhar com imagens em algum momento durante o desenvolvimento Android. A maioria dos profissionais usa o Adobe Photoshop, mas você pode não ser capaz de pagar muito por um programa de edição de imagens. Portanto, você tem duas alternativas gratuitas: Paint.NET e GIMP.

O programa de manipulação de imagens Paint.NET foi escrito com o .NET Framework. O Paint.NET, que opera na plataforma Windows, funciona muito bem e é usado por muitos desenvolvedores no mundo inteiro.

O programa de fonte aberta GIMP é parecido com o Photoshop. O GIMP pode ser instalado no Windows, Linux ou Mac.

Índice

• Símbolos •

@ (símbolo), 241

• A •

aberta, plataforma, 10
aberto, código-fonte, 10
ação
  barra de ação, 20
  intenção pendente, 169
  intenções, 13
ação, barra
  ações, 19–20
  botão Back, 19
  botão Up, 19
  contextual, 20
  guias, 19
  logotipo do aplicativo, 19
  página, 19
  tablet, 337
ação, menu da barra, 232
acelerômetro, 23
acesso, permissões do local, 88
`acquire()`, método, 298
`ActionBarSherlock`, 336
`ACTION_DIAL`, ação, 167
`ACTION_EDIT`, ação, 167
`ACTION_MAIN`, ação, 167
`ACTION_VIEW`, ação, 167
`Activity`, classe, 113–119
`AdapterContextMenuInfo`, objeto, 286
`adb devices`, comando, 356–357
`adb` (Ponte de Depuração Android), ferramenta, 132
Add/Edit Task Reminder, tela, 220
`add( )`, método, 325
`Add-on` Sites, caixa de diálogo, 353
`add-ons`, pasta, 48
`addPreferencesFromResource( )`, método, 321
`Add Reminder`, ícone do menu, 234
`Add Repository`, janela, 44
`addToBackStack( )`, método, 349
AdMod, empresa de publicidade, 197
ADT (Ferramentas de Desenvolvimento Android)
  Assistente para Exportar, 188
  configurar Eclipse com, 44–46
  designer visual, 112
Agit, aplicativo, 365
`AlarmManager`, classe
  ativar processos com, 292
  criar classe `OnAlarmReceiver`, 294–296
  criar classe `ReminderIntentService`, 296–298
  criar classe `ReminderManager`, 293–294
  criar classe `ReminderService`, 298–299
  erro de execução, 139–140
  motivos para usar, 289–290
  reinicializar serviços, 299–300
`ALARM_SERVICE`, tipo de serviço do sistema, 125
alerta, caixa de diálogo
  confirmação da janela `AlertDialog`, 256
  criar, 255–257
  descrição básica, 252
  escolher para tarefa, 252
  motivos para usar, 252
`AlertDialog`.Builder, classe, 255
`AlertDialog`, classe, 259
`AlertDialogFragment`, fragmento, 255
`ALTER`, instruções, 268
alternância, exibição do botão, 108–109
amarelo, ícone de aviso, 61
Amazed, aplicativo, 364
Amazon Appstore
  publicar em, 357–360
  recursos, 203

amostra, aplicativo, 363–366
Android 3.0 (Honeycomb), 17, 21
Android 4.0 (Ice Cream Sandwich), 17, 21
Android 4.1 (Jelly Bean), 17, 21
Android ADT
  Assistente para Exportar, 188
  configurar Eclipse com, 44–46
  descrição básica, 34
  designer visual, 112
Android Beam, 58
Android, Biblioteca de Suporte, 18, 336
Android, caixa de diálogo SDK Manager, 353–354
Android, código-fonte, 30
Android, Dispositivo Virtual (AVD), 63–65, 337–338
`android:icon`, valor, 233
`android.intent.action.MAIN`, filtro de intenção, 340
`android.jar`, arquivo, 80
Android, Kit de Desenvolvimento Nativo (NDK), 39
`android:layout_height="match_parent"`, atributo, 101
`android:layout_width="match_parent"`, atributo, 101
`AndroidManifest.xml`, arquivo, 78, 181–182, 291, 339
Android Open Source Project, 30
`android:orientation="vertical"`, atributo, 101
Android Package, arquivo. *See* Veja APK
`android.R`, classe, 233
Android, recursos da estrutura, 29–30
Android, SDK
  aplicativo Silent Mode Toggle, 96–97
  caminho das ferramentas, definir, 38–40
  demos API, 52
  descrição básica, 34
  download, 36–37
  ferramenta adb, 38
  navegação, 47–48
  pacotes, 49–50
  pastas, 47–48
`android.util.Log`, pacote, 136
Android, versão, 48
`anim/`, diretório, 82
ANR (Aplicativo Não Respondendo), caixa de diálogo, 16
apagar tarefa, 287–288
API, demos, 52
API Demos, aplicativo, 364
APK (Pacote Android), arquivo
  arquivo manifest, 186–187
  assinar aplicativos, 187–188
  criar, 189–191
  descrição, 43
  ferramentas para construir, 187
aplicativo
  adicionar imagens a, 102–106
  amostra, 363–366
  arquivo manifest, 86–88
  assinar, 187–188
  atividade, 13
  capacidade mashup, 11–12
  classificar, 203
  colocar preço, 195–197
  comercializar, 10
  compatibilidade cruzada, 11
  criar ícone de inicialização para, 107–109
  demos API, 52
  estrutura, 30–31
  executar no emulador, 131
  fazer upload para Google Play Store, 198–202
  global, 157–159
  Hello Android, 70–77
  intenções, 13–14
  Java, 13
  Keytool, 188
  motivos para desenvolver para Android, 9–12
  navegação de pastas, 77–86
  nome, 56
  participação no mercado, 10
  plataforma aberta, 10
  publicar, 198–202
  Silent Mode Toggle, 92–99
  testar, 149
  título, 200
  visualizar no designer visual, 110–111
aplicativo, logotipo (barra de ação), 19
`AppWidgetProvider`, classe
  classe `IntentService`, 175

componente da tela inicial, 165–166
implementação, 170–171
layout do componente do aplicativo, 173–174
layout `widget.xml`, 173
listagem do código, 176–177
mensagem de transmissão, 170–171
metadados, 179–181
registrar novos componentes no arquivo manifest, 181–182
responder à intenção, 172
armazenamento. *Veja* dados, armazenamento
armazenamento de chaves, 188
array
com tipos, 153
convenções do nome de arquivo `arrays.xml`, 83
inteiro, 153
`assets`, pasta, 81
assinar aplicativo, 187–188
assíncrono, processamento, 15
`AsyncTask`, classe, 15
ativa/em execução, estado da atividade, 114
atividade
adicionar fragmento, 217–218
alteração da configuração, 118
básico do desenvolvimento, 13
ciclo de vida, 115–118
criar, 118–123
estados, 114
grupo, 113
importância, 113
indicador destrutível, 118
métodos, 114, 116
pilha, 114
quando usar, 213
seguir caminho, 117
tratamento da entrada do usuário, 119–120
usar para criar e editar lembretes, 214–217
atualização, entrada da tarefa de operação, 281–282
atualizar notificações, 311
áudio e vídeo, suporte, 24
`AudioManager`, variável
obter bom serviço, 124–125
trocar modo silencioso com, 125–128
`AUDIO_SERVICE`, tipo de serviço do sistema, 125
automática, recompilação, 61
automático, teste, 148
avaliação, intenção pendente, 169
AVD (Dispositivo Virtual Android), 63–66, 337–338
AVD Manager, caixa de diálogo, 66

• B •

Back, botão (barra de ação), 19
banco de dados, tabela, 267, 268
Barnes & Noble Nook, 32
`beginTransaction( )`, método, 349
`bindService( )`, método, 167
`bin`, pasta, 78, 83–84
bloqueio, processos, 253
Bluetooth, 49
Bluetooth, rádio, 22
booleano, valor, 153
botão
data e hora, 243
exibição do botão de alternância, 109–110
seletor, 242–243
`BroadcastReceiver`, objeto, 166, 294
brutos, arquivo de componentes, 81
Build Target, definições, 62
`buildUpdate( )`, método, 178
bússola, 22

• C •

cache, 262
caixa de diálogo
alerta, 252–257
personalizada, 253
Progresso, 253
`Calendar`, objeto, 249
callbacks, seletor de data e hora, 248–249
camadas, imagem, 157
câmera
descrição básica, 22

permissões, 88
cancelAll( ), método, 311
cancel( ), método, 311
carregador
banco de dados SQLite, 283
CursorLoader, 283
descrição básica, 18
Cascata, Folhas de Estilo (CSS), 153
categoria, intenção pendente, 168
certificado, assinar aplicativo para processo, 188
CheckBoxPreference, preferência, 314
checkIfPhoneIsSilent( ), método, 128
Choose Packages to Install, caixa de diálogo, 38, 50
ciclo de vida, atividade
alteração da configuração, 118
caminhos da atividade, 117–118
métodos da atividade, 116
métodos de callback, 115
principais loops, 116
cinco estrelas, sistema de classificação, 203
classe
Activity, 113–119
AlarmManager, 289–290, 292–299
AlertDialog.Builder, 255
android.R, 233
AppWidgetProvider, 165
AsyncTasks, 15
DatePicker, 242
DialogFragment, 242, 244
DialogPreference, 314
estrutura, 123–127
ImageView, 105
Intent, 166
IntentService, 175
ListFragment, 223–228
LoaderManager, 286
MainActivity, 119
MenuInflater, 234
NotificationManager, 209
Object, 125
OnAlarmReceiver.java, 292
OnClickListener, 121
OnDateSetListener, 248–250
OnTimeSetListener, 248
ProgressBar, 253
ProgressDialog, 253
ReminderDbAdapter, 268
ReminderEditActivity, 214
ReminderEditFragment, 249
ReminderListActivity, 211–212
ReminderManager, 293–294
ReminderManager.java, 292
ReminderProvider, 265
ReminderService, 298–299
ReminderService.java, 292
RemoteViews, 164
SharedPreferences, 315–316
SQLiteOpenHelper, 268
TimePicker, 242
View, 96, 121
WakeReminderIntentService, 296–297
WakeReminderIntentService.java, 292
classificar aplicativos, 203
cliente, computação do servidor, 24
clique, eventos
curto, 225–226
longo, 226–227
Cloud Messaging, estrutura, 26
codificar aplicativos
atividades, 113–122
classes da estrutura, 123–127
erros, 134–140
instalar aplicativo, 131–135
interação, 148
reconhecer todas as soluções possíveis, 147–149
testar aplicativos, 149
teste automático, 148
código
comentar, 141
digitar, 121–122
extrair para método, 122–123
linhas irregulares e vermelhas em, 121
color/, diretório, 82
colors.xml, convenções do nome de arquivo, 83
COLUMN_BODY, propriedade, 267
COLUMN_DATE_TIME, propriedade, 267
COLUMN_ROWID, propriedade, 267

columns, parâmetro, 280
COLUMN_TITLE, propriedade, 267
colunas e linhas, 266–267
comentar código, 141
comentário, 203
comercializar, 10
commit( ), método, 247, 326, 349
compartilhadas, preferências, 262, 315–316
compatibilidade, 63
  código da versão, 63
  compatibilidade cruzada, 11
componente. *See* Veja tela inicial, componente
componente, intenção pendente, 168
compressão de imagem, 156
computador, hardware, 33–34
configuração
  depuração, 67
  Eclipse, 43–45
  execução, 66–69
configuração, alteração, 118
Console, exibição (Eclipse), 75–76
consultar (ler), operação, 280–281
contatos
  capacidade mashup, 12
  modos de usar, 24
ContentResolver, método delete( ), 287
ContentUris.parseId( ), método, 279
conteúdo, provedor
  banco de dados SQLite, 264–272
  constantes, campos e construtores, 265–266
  criar para manter código do banco de dados, 264
  descrição, 263
  URLs, 269–272
context, parâmetro, 170
contextual, barra de ação, 20
contextual, menu
  arquivo XML do menu, 236–237
  carregar o menu, 237
  descrição básica, 233
  tratamento da seleção do usuário, 237–238
cópia, proteção contra, 200
corpo, campo, 220
cor, recurso, 154
Create New Android Virtual Device, caixa de diálogo, 65, 337–338
create table DATABASE_TABLE, propriedade, 267
criada, estado da atividade, 114
cron, serviço, 208
CRUD (Criar, leR, atUalizar e Deletar), tarefa, 209–210
cruzada, compatibilidade, 11
CSS (Folhas de Estilo em Cascata), 153
cursor
  controles sem cursor, 15
  retornar toda tarefa com, 283–287
CursorLoader, carregador, 282–283
Cursor, objeto, 278–279
curto, eventos de clique, 225–226
Customer, objetos, 79

• D •

dados
  intenção pendente, 167–169
  intenções, 13
  motivos para desenvolver para Android, 9–12
dados, armazenamento. Veja também SQLite, banco de dados
  armazenamento externo, 263
  armazenamento interno, 262
  armazenamento remoto, 262
  arquivos públicos, 263
  cache local, 262
  conexão da rede, 263
  criação do provedor de conteúdo SQLite, 264–272
  descrição básica, 261–262
  preferências compartilhadas, 262
  provedor de conteúdo, 263
  selecionar opções de armazenamento, 262–263
Dalvik Debug Monitor Server. *See* Veja DDMS
data e hora, botões, 242
data, seletor
  arquivo DatePickerDialogFragment, 244–245

atendente de clique do botão Date, 243–244
callbacks, 248–249
método `showDatePicker( )`, 246–247
Date, atendente de clique do botão, 243–244
`DATE_FORMAT`, 250
`DatePicker`, classe, 242
`DatePickerDialogFragment`, arquivo, 244
DDMS (Dalvik Debug Monitor Server)
depurar, 49–50
exibir mensagens, 136–137
exibir mensagens de registro em, 135–136
mensagem de erro de fechamento forçado, 134–135
recursos, 134
visor LogCat, 134–135, 138
`delete( )`, método, 279, 282, 287–288
densidade, pasta, 103
densidade, pixel independente (dp), 152
dependência, injeção, 368
depuração, configuração, 67
desenho, recursos, 106
desenvolvedor
considerações para se tornar, 21–22
taxa de registro, 28
desenvolvedor, perfil (Google Play Store), 192–195
desenvolvimento
atividade, 13
bibliotecas Open Handset Alliance (OHA), 31–32
código-fonte Android, 30
controles sem cursor, 15
emulador, 49
estrutura Android, 29–30
estrutura do aplicativo, 30–31
exibições, 15
hardware, 33–34
instalar e configurar ferramentas de suporte, 34
intenção, 13–14
interface do usuário, 100–102
kernel Linux 2.6, 28–29
linguagem de programação Java, 13, 32–33
plataforma Android, 48
processamento assíncrono, 15–16
recursos, 12
serviços em segundo plano, 16
`DialogFragment`, classe, 242, 244
`DialogPreference`, classe, 314
dimensão, 152
`dimens.xml`, convenções do nome de arquivo, 83
dispositivo
fazer download do driver USB do Windows, 50
hardware, 21–22
instalar aplicativo em, 133–135
reinicializar, 299–300
`docs`, pasta, 48
`doReminderWork( )`, método, 298, 307–308
download
Eclipse, 41
SDK do Android, 36–37
dpi (pontos por polegada), 152
dp (pixel independente da densidade), 152
Draw 9-patch, utilitário, 368
`drawable/`, diretório, 82
`drawable-hdpi/`, diretório, 82
`drawable-ldpi/`, diretório, 82
`drawable-mdpi/`, diretório, 82
`drawable-xhdpi/`, diretório, 82
driver, modelo (kernel Linux 2.6), 29
droid-fu, ferramenta, 367

• E •

Eclipse
aplicativo Silent Mode Toggle, 92
botão Open Perspective, 136–137
configuração, 43–45
definições do espaço de trabalho, 41–42
download, 41
executar, 41
exibição Console, 75–76
IDE, 34
iniciar projeto em, 55–60

instalação, 41
tela de boas-vindas, 42–43
Eclipse, depurador
configurar aplicativo como depurável, 142–143
erro de execução, 139
erro lógico, 145–146
iniciar, 142–145
navegação da depuração, 145
perspectiva Debug, 144–145
pontos de interrupção, 140–142
propriedade depurável, 142–143
editar preferências, 326
`edit( )`, método, 326
`editReminder( )`, método, 236, 344–346, 348
`EditText`, exibição, 239–242
`EditTextPreferences`, preferência, 314
`EditText.setError( )`, método, 258
eliminação, entrada da tarefa para operação, 282
e-mail, 203
emulador
configurar, 63–66
execução do aplicativo Hello Android, 49
executar aplicativo em, 131
limitações, 49
número da porta, 71
tela de inicialização, 71
tela Inicial, 72–75
vantagens, 49
entrada, destaque do campo, 259
entrada, validação
destaque do campo de entrada, 259
instância `AlertDialog`, 259
mensagem Toast, 258
métodos para, 259
validação personalizada, 259
Environment Variables, caixa de diálogo, 39–40
erro
arquivo de recursos, 83
codificar aplicativo, 134–136
comentar código para gerar, 141
Dalvik Debug Monitor Server (DDMS), 134–139
Eclipse Debugger, 139–146
execução, 139–140
lógico, 145–146
mensagens de fechamento forçado, 134–135
método `setError( )`, 258
erro, mensagem, 60–62
erro, relatório, 203
escala, pixel independente (sp), 152
espaço de trabalho do Eclipse, 41–42
estados, atividade, 114
estilo, 153
estrangeiro, tradução de idioma, 158
estrutura
Android, 29–30
aplicativo, 30–31
mídia, 29
estrutura, classe
alternar modo silencioso com AudioManager, 127
descrição básica, 123–124
obter bom serviço, 124–125
verificar estado da campainha do telefone, 124–125
evento, atendente
digitar código, 121–122
evento do teclado, 120
eventos de clique, 119–121
execução
estrutura Android, 29
trabalhar com preferências em atividades durante, 323–326
execução, configuração, 67–70
execução, erro, 139–140
exibição
aplicativo Silent Mode Toggle, 96
básico do desenvolvimento, 15
exibir propriedades, 98–99
interface do usuário, 101–102
expandido, menu, 232
expiração, assinar aplicativo com data, 188
explícito, intenção pendente do componente, 169
Export Android Application Wizard, 189–191
Exportar, Assistente, 189
externo, armazenamento, 263

Extract Android String, caixa de diálogo, 155

• F •

Facebook, SDK para Android, 365
ferramenta
  droid-fu, 367
  Git, 369
  Hierarchy Viewer, 368
  ignition, 367
  Kit de Ferramentas do Tradutor, 368
  layoutopt, 369
  programa de fonte aberta GIMP, 370
  programa de manipulação de imagens Paint.Net, 370
  RoboGuice, 367–368
  UI/Application Exerciser Monkey, 369
  utilitário Draw 9-patch, 368
  zipalign, 369
Ferramentas de Desenvolvimento Android. *See* Veja ADT
`fill_parent`, valor, 102, 331
`findPreference( )`, método, 321
`findViewById( )`, método, 121
`finishEditor( )`, método, 348
`finish( )`, método, 117, 347
físico, dispositivo, 133–136
`FLAG_ONE_SHOT`, flag, 308
flags, parâmetro, 170
fonte, pasta (`src`), 78–80
forçado, mensagem de erro de fechamento, 134–135
Foursquare, aplicativo de rede social, 23
fragmento
  adicionar a atividade, 217–218
  ciclo de vida, 219
  classe `ListFragment`, 223–228
  criar layout para adicionar/editar, 220–223
  descrição básica, 17
  quando usar, 213
  tablet, 333
`FrameLayout`, layout, 97

• G •

`gen`, pasta, 61–62, 84–86
geolocalização
  capacidade mashup, 11
  jogo baseado no local, 11
  rede social, 11
`getActivity( )`, método, 287
`getArguments( )`, método, 218
`getCacheDir( )`, método, 262
`getDefaultSharedPreferences( )`, método, 324
`getEditText( )`, método, 321
`getExternalFilesDir( )`, método, 263
`getExternalStorageState( )`, método, 263
`getListView( )`, método, 226
`getLoaderManager( )`, método, 283
`getLock( )`, método, 297
`getMenuInfo( )`, método, 286
`getString( )`, método, 325
`getSupportFragmentManager( )`, método, 349
`getSystemService( )`, método, 125, 294
`getType( )`, método, 271
GIMP, programa de fonte aberta, 370
Git, ferramenta, 369
global, recurso, 157–159
Gmail, aplicativo, 14
Google
  API Maps, 25–26
  diretrizes da marca, 31
  estrutura Cloud Messaging, 26
Google AdSense, empresa de publicidade, 197
Google Checkout, conta comercial, 195
Google I/O 2012, aplicativo, 363
Google Nexus 7, AVD, 337
Google Play Store
  aplicativo pago em, 196
  capturas de tela, 197–198
  descrição básica, 185
  perfil do desenvolvedor, 192–195
  preço do aplicativo, 195–197
  upload do aplicativo para, 198–202
GPS, recurso, 23
GPX (Formato de Troca GPS), 135
gráficos, biblioteca (Open GL), 29
Graphical Layout, guia, 110

gravar armazenamento externo, permissões, 88
`gravity`, propriedade, 240
`GridLayout`, layout, 18
`groupBy`, parâmetro, 281
grupo, atividade, 113
guias, barra de ação, 19

• H •

hardware
  acelerômetro, 23
  cartão SD, 23
  computador, 33–34
  descrição, 21
  recurso GPS, 23
  recursos, 22
  sistema operacional, 33
  tela de toque, 22–23
`having`, parâmetro, 281
`hdpi`, pasta, 108
Hello Android, aplicativo
  executar em emulador, 70–75
  verificar status da implementação, 76–77
Hierarchy Viewer, ferramenta, 368
Holo, temas, 20–21
Honeycomb (Android 3.0), 17, 21
HoneycombGallery, aplicativo, 365
hora e data, botões, 242
hora, seletor
  atendente de clique do botão Time, 247
  callbacks, 248–249
  método `showTimePicker( )`, 247–248
  `TimePickerDialogFragment`, 247–248

• I •

Ice Cream Sandwich (Android 4.0), 17, 21
ícone
  menu de opções, 232
  modelo, 107
`id`, atributo, 105
idioma, tradução, 157
`id`, parâmetro, 226
`if`, instrução, 324
ignition, ferramenta, 367
imagem
  adicionar a aplicativo, 102–106
  adicionar a layout, 104–105
  camadas, 157
  colocar na tela, 102–104
  compressão, 156
  definir propriedades, 104–105
  pixels, 156
  recursos de desenho, 106
  resolução, 156
`ImageView`, classe, 105
implantação, aplicativo Hello Android para status, 76–77
implícito, intenção pendente do componente, 168
inicial, componente da tela
  adicionar componente a, 183–184
  aplicativo Silent Mode Toggle, 163
  classe `AppWidgetProvider`, 164–165, 170–179
  classe `RemoteViews`, 164
  classes associadas, 162
  criar, 170–179
  descrição básica, 162–163
  estados, 163
  intenção pendente, 165–168
inicialização, configuração, 66–69
inicialização, ícone
  colocar no projeto, 108
  combinar tamanhos com densidade da tela, 108
  modelos, 107
  padrão, 107
inicialização, receptor
  criar, 299–301
  verificar, 302
Inicial, tela, 72–75
`initialLayout`, propriedade, 180
`initLoader( )`, método, 283, 286
in (polegada), 152
inserir, operação, 280
`insert( )`, método, 280
`insertOfThrow( )`, método, 279
instalação
  aplicativo no dispositivo físico, 133–136
  Eclipse, 41
  executar aplicativo no emulador, 131

JDK do Java, 35–36
instalações versus instalações ativas, 203
Install Details, caixa de diálogo, 44
inteira, duração (ciclo de vida da atividade), 116
inteiros, array, 153
inteiro, valor, 153
intenção, resolução, 169
intenção. Veja também pendente, intenção
básico do desenvolvimento, 13–14
elementos, 13
enviar mensagens com, 14
iniciar nova atividade com, 227–228
registrar receptor, 14
seletor, 228–230
transmitir, 14
`Intent`, classe, 166
`Intent`, objeto, 166
`Intent`, parâmetro, 170
`IntentService`, classe, 175
interação, aplicativos para codificar, 148
Internet
capacidade mashup, 11–12
computação do cliente-servidor, 24
permissões, 88, 291
interno, armazenamento, 262
`isFinishing( )`, método, 117
IU (interface do usuário)
adicionar imagens a aplicativos, 102–106
aplicativo Silent Mode Toggle, 92–99
atividades, 113, 119
atributos do layout XML, 101
carregadores, 18
código de fonte aberta, 10
criar ícones de inicialização para aplicativos, 107–109
descrição básica, 91
desenvolvimento, 100–102
designer visual, 97–99
exibição do botão de alternância, 109–110
exibições, 101–102
sistema de exibição, 30
visualizar aplicativos no designer visual, 110–111

• J •

J2EE (Java Platform, Enterprise Edition), 13
Java
classe `Object`, 125
pacote, 57
Java, JDK
descrição básica, 34
instalação, 35–36
Java, linguagem de programação
básico do desenvolvimento, 32–33
básico do desenvolvimento Android, 13
letras maiúsculas e minúsculas, 2
Java Platform, Enterprise Edition (J2EE), 13
Jelly Bean (Android 4.1), 17, 21
jogar, 11
`jtwitter.jar`, biblioteca, 83–84
jUnit, estrutura de teste da unidade, 148
JVM (Máquina Virtual Java), 32

• K •

K-9 Mail, aplicativo, 365
Key Creation, tela, 190
Keyhole Markup Language (KML), arquivo, 135
Keystore Selection, tela, 189
Keytool, aplicativo, 188
Kindle Fire
configurar ADB, 356–357
criar emulador do tipo Kindle, 352–355
descrição básica, 351–352
KISS, princípio (Keep It Simple, Stupid), 26
KML (Keyhole Markup Language), arquivo, 135

• L •

layout
aplicativo Silent Mode Toggle, 93–96
componente do aplicativo, 173–174
fragmento, 220–223
recursos de desenho, 106
situação de conteúdo estático, 99
tablet, 334–335
`layout/`, diretório, 82
`layout_height`, atributo, 101
`layoutopt`, ferramenta, 369

layout_width, atributo, 101
LED, luz, 305
lembrete, campo da data, 220
lembrete, campo da hora, 220
lembrete, script, 208
lembrete, tarefa, 236
ler (consultar), operação, 280–281
ler permissões do estado do telefone, 88
letras maiúsculas e minúsculas, 2
libs, pasta, 78, 83–84
limit, parâmetro, 281
limpar notificações, 311
LinearLayout, layout, 97
Linguagem de Consulta Estruturada (SQL), 261
Linux 2.6, kernel, 28–29
lista, exibição, 21
ListFragment, classe
  dados falsos, 224
  descrição, 223
  eventos de clique do usuário, 225–227
  exibição de lista, 223
list_menu_item_longpress.xml, arquivo, 238
ListPreference, preferência, 314
ListView, exibição, 332
LoaderCallback, interface, 283, 285
LoaderManager, classe, 286
Locais, gerenciador, 30
localização, camadas de imagem, 157
local, spoofing de dados, 134
LOCATION_SERVICE, tipo de serviço do sistema, 125
LogCat, visor, 134–135, 138
lógico, erro, 145–146
LOLcat Builder, aplicativo, 364
longo, evento de clique, 226–227
l, parâmetro, 226
lpdi, pasta, 108
luz, notificação, 305

• M •

MainActivity, classe, 119
MainActivity.java, arquivo, 79, 118
manifest, arquivo
  arquivo do pacote Android, 186–187
  código da versão, 86–87
  conteúdo, 86
  nome da versão, 87–88
  permissões, 88
Maps, API, 25–26
Maps, aplicativo, 23
marca, diretrizes, 31
mashup, capacidade, 11–12
match_parent, valor, 102, 332
mAudioManager, variável, 124
mCalendar, objeto, 250–251
mdpi, pasta, 103
medida, unidades, 152
memória, componente do aplicativo, 175
memória, gerenciamento (kernel Linux 2.6), 29
menu
  ações do usuário, 235
  barra de ação, 232
  contextual, 233, 236–238
  exemplos bons/ruins, 231
  expandido, 232
  opções, 232–233
  submenu, 233
  tarefa de lembrete, 236
  XML, 233–234
Menu, botão, 18–19
menu_delete, valor, 237
menu/, diretório, 82
MenuInflater, classe, 234
menuInfo, parâmetro, 227
menu, parâmetro, 227
menu, recurso, 154
mercado, participação, 10
Messaging, aplicativo, 14
metadados, 179–181
método
  atividade, 114
  extrair código, 122–123
mídia, estruturas, 29
milímetro (mm), 152
MIME, tipo, 168
minHeight, propriedade, 180
minLines, propriedade, 240
Min SDK Version, definições, 62
minSdkVersion, valor, 186
minWidth, propriedade, 180
mm (milímetro), 152

MMS (serviço de mensagens multimídia), 364
modelo gratuito, preço do aplicativo, 196
modelo, ícone, 107
Monkey, interface do usuário e aplicativo, 148
moveToFirst( ), método, 278

• N •

New Android App Wizard, 56–57
New Blank Activity, tela, 59
New Java Class, caixa de diálogo, 170
newWakeLock( ), método, 297
Nexus One, dispositivo, 112
no aplicativo, compras, 197
nome
  aplicativo, 56
  Dispositivo Virtual Android, 64
  pacote Java, 57
  perfil do desenvolvedor Google Play, 192
  recursos de nomenclatura do diretório values, 83
normal, imagem no modo, 104
Notepad Tutorial, 366
Notification Manager
  adicionar recursos de string, 310
  criar notificação, 307–310
  exibir fluxo de trabalho, 310
  limpar notificação, 311
  método doReminderWork( ), 307–308
NotificationManager, classe, 209
Notification, objeto, 311
notify( ), método, 310–311
notify_new_task_message, recurso de string, 310
notify_new_task_title, recurso de string, 310
nullColumnHack, parâmetro, 280

• O •

Object, classe, 125
OHA (Open Handset Alliance), 31–32
onActivityCreated( ), método, 219
onAlarmReceiver, classe, 294–296
onAlarmReceiver.java, classe, 292
OnBootReceiver, 300
onBootReceiver, classe, 300–301
onClickListener, classe, 121
onClick( ), método, 121
onContextMenuItemSelected( ), método, 238
onContextSelectedItem( ), método, 287
onCreateContextMenu( ), método, 226, 237
onCreateLoader( ), método, 283, 286
onCreate( ), método, 114, 116–119, 121, 219
onCreateView( ), método, 219
OnDateSetListener, classe, 248–250
onDateSet( ), método, 250
onDestroy( ), método, 116–118
OnEditReminder.java, interface, 345
onFocusChanged( ), método, 258
onHandleIntent( ), método, 177, 297
onKeyDown( ), método, 120
onListItemClick( ), método, 226–228, 346
onLoaderReset( ), método, 283, 287
onLoadFinished( ), método, 283, 287, 348
onOptionsItemSelected( ), método, 322, 346
onPause( ), método, 114, 116–118
onReceive( ), método, 177, 291, 300
onRestart( ), método, 117
onResume( ), método, 116–118, 129
onSaveInstanceState( ), método, 252
On Save, método, 258
onStart( ), método, 116–118
onStop( ), método, 116–118
onTimeSetListener, classe, 248–250
onTimeSet( ), método, 250
onUpdate( ), método, 177
onUpgrade( ), método, 268
onViewCreated( ), método, 226, 234
opções, menu, 232–233
Open GL (biblioteca de gráficos), 29
Open Handset Alliance (OHA), 31–32
Open Perspective, botão, 136–137
operacional, sistema
  compatibilidade, 63

destino da construção, 62
plataformas, 33
raízes, 11
`orderBy`, parâmetro, 281

• p •

pacote Java, 57
padrão, recurso, 83
página (barra de ação), 19
pago, preço do aplicativo do modelo, 196
Paint.Net, programa de manipulação de imagens, 370
paisagem, modo, 112
parada, estado da atividade, 114
parado, estado do fragmento, 219
pasta
`assets`, 81
`bin`, 78, 83–84
densidade, 103
fonte, 78–80
`gen`, 61–62, 84–86
`hdpi`, 108
`ldpi`, 108
`libs`, 78, 83–84
`mdpi`, 103
navegação, 77–86
recursos, 81, 83
SDK do Android, 47–48
Target Android, Library, 80
`xhdpi`, 108
pausada, estado da atividade, 114
pausado, estado do fragmento, 219
pendente, intenção
ações, 167
avaliação, 168
categoria, 168
como usar, 169–170
componente, 168
componente explícito, 168
componente implícito, 168
dados, 167–169
parâmetros, 169
resolução da intenção, 169
sistema de intenções Android, 165–166
tipo, 168
permissões
comumente solicitadas, 88
definir no arquivo `AndroidManifest.xml`, 291
experiência do usuário afeta, 290
Internet, 291
segurança, 25
personalizada, caixa de diálogo, 253
personalizada, validação, 259
pilha, atividade, 114
pixel (px), 152, 331
pixels, 156
`platforms`, pasta, 48
`platform-tools`, pasta, 48
Play Store. *See* Veja Google Play Store
Ponte de Depuração Android (adb), ferramenta, 132
ponto (pt), 152
pontos de interrupção, 140–142
pontos por polegada (dpi), 152
porta, número, 71
portas, redirecionamento, 134
`position`, parâmetro, 226
preço, aplicativo, 195–197
`PreferenceActivity`, classe
abrir, 322
arquivo `TaskPreferences`, 320–321
descrição, 315
objetos `Preference`, 315
seleções de menu, 322–323
tela de preferências, 320
`PreferenceCategory`, preferência, 317
Preferences, caixa de diálogo, 46–47
`PreferenceScreen`, preferência, 316
preferência
adicionar valores, 325–326
arquivo, 318–319
`CheckBoxPreference`, 314
classe `DialogPreferences`, 314
compartilhada, 315–316
definir, 316
editar, 326
`EditTextPreference`, 314
em atividades na execução, 323–326
layout, 316–317
`ListPreference`, 314
pares de chave-valor, 313
persistir valores, 316
recuperar valores, 323–325
recursos de string, 319–320

RingtonePreference, 314
primeiro plano, duração (ciclo de vida da atividade), 116
processos, gerenciamento (kernel Linux 2.6), 29
ProgressBar, classe, 253
ProgressDialog, classe, 253
progresso, caixa de diálogo, 253
progresso, classes, 253
projection, variável, 278
project.properties, arquivo, 78, 88
Project Selection, caixa de diálogo, 68
projeto
aplicativo Hello Android, 70–77
aplicativo Task Reminder, 210
arquivo de propriedades, 88
arquivo manifest do aplicativo, 86–88
colocar ícone de inicialização, 108
configuração da inicialização, 66–69
configuração do emulador, 63–66
definições Build Target e Min SDK Version, 62–63
Eclipse, 55–60
estrutura, 77–88
fechar, 89
mensagem de erro, 60–62
navegação de pastas, 77–86
promocional, captura, 200
promocional, texto, 200
proximidade, sensor, 22
pt (ponto), 152
publicar
na Amazon Appstore, 357–360
na Google Play Store, 198–202
publicidade, 196
públicos, arquivos, 263
putString( ), método, 326
px (pixel), 152, 331

• Q •

query( ), método, 278, 280

• R •

rádio, 22
raw/, diretório, 82
real, captura da tela em tempo, 197
RECEIVE_BOOT_COMPLETED, permissão, 299
recurso
cor, 154
dimensões, 152
estilo, 153
global, 157–159
imagens, 156
menu, 154
mover strings para, 154–155
padrão, 83
temas, 153
tipos, 151
valor, 153
recurso, detecção, 11
recursos, erro do arquivo, 83
recursos (res), pasta, 81, 83
redes, pilha (kernel Linux 2.6), 29
registerForContextMenu( ), método, 226, 234, 236
registro
receptor da intenção, 14
taxa de registro do desenvolvedor, 28
registro, mensagem DDMS, 135
reinicializar dispositivos, 299–300
RelativeLayout, tipo de layout, 95–96
release( ), método, 298
ReminderDbAdapter, classe, 267
ReminderEditActivity, classe, 214
ReminderEditFragment, classe, 249
reminder_edit.xml, arquivo, 220–222
ReminderListActivity, classe, 211
ReminderListAndEditorActivity.java, arquivo, 339
reminder_list_and_editor.xml, arquivo, 342
ReminderListFragment, fragmento, 212–213, 283–287
ReminderManager, classe, 293–294
ReminderManager.java, classe, 292
ReminderProvider, classe, 265, 274–279
ReminderService, classe, 298–299
ReminderService.java, classe, 292
RemoteViews, classe, 164
Replica Island, aplicativo, 366
Reply, botão, 21
RequestCode, parâmetro, 170

res, diretório, 82
res/menu, diretório, 236
resolução, imagem, 156
res (recursos), pasta, 81, 83
res/values/strings.xml, arquivo, 319–320
retomada, estado da atividade, 114
retomado, estado do fragmento, 219
retrato, modo, 112
RingtonePreference, preferência, 314
R.java, arquivo, 84–86
R.layout.activity_main, arquivo, 119
R.layout.reminder_edit_activity, arquivo, 215
RoboGuice, ferramenta, 367–368
Run As, caixa de diálogo, 70
Run Configurations, caixa de diálogo, 67–69

• S •

salvar nomes de arquivo, 252
samples, pasta, 47
savedInstanceState( ), método, 245, 252
SCREENSIZE, valor do emulador, 64
scrollbars, propriedade, 240
ScrollView, exibição, 332
SD, cartão (Cartão Digital Seguro)
  armazenamento removível, 262
  descrição básica, 23–24
  permissões, 291
SDK (kit de desenvolvimento do software)
  aplicativo Silent Mode Toggle, 96–97
  definições Build Target e Min SDK Version, 62–63
  definir local, 46
  JDK do Java, 34
  navegação, 47–48
  pacotes, 49–50
  pastas, 47–48
  SDK do Android, 36–40
SDK Manager, 36–37, 50
Security Warning, caixa de diálogo, 41
segundo plano, serviços, 16
segurança
  modelo de segurança (kernel Linux 2.6), 29
  permissões, 25
Seguro, Cartão Digital. *See* Veja SD, cartão
selectionArgs, parâmetro, 281
selection, parâmetro, 281
seletor, botões, 242–243
seletor, intenção, 228–230
sendBroadcast( ), método, 167
serviço em segundo plano, 16
setButtonClickListener( ), método, 122–123
setClickable( ), método, 120
setContentView( ), método, 119, 212
setError( ), método, 258
setHasOptionsMenus( ), método, 234
setKeyListener( ), método, 321
setLastestInfo( ), método, 308
setLatestEventInfo( ), método, 309
setListAdapter( ), método, 286
set( ), método, 294
setNotificationUri( ), método, 278
setReminder( ), método, 294
ShareCompat, 18
SharedPreferences, classe, 315–316
showDatePicker( ), método, 244, 246–247
show( ), método, 247
showTimePicker( ), método, 247–248
silencioso, imagem no modo, 104
Silent Mode Toggle, aplicativo
  arquivo de layout XML, 94–96
  componente da tela inicial, 162
  declaração XML padrão, 95
  definições do projeto, 92
  exibições, 96
  ferramentas do layout SDK, 96–97
  imagens de telefone, 104
  layout, 93–96
  no modo de campainha normal, 93
  no modo de campainha silencioso, 93–94
  tipo de layout, 95
SimpleCursorAdapter, adaptador, 286
SimpleDateFormat, 250
SKIN, valor do emulador, 64
social, rede, 11
Socket Seguro, Camada (SSL), 29

software, ferramentas
APIs Google, 25–26
contatos, 24–25
Internet, 24
segurança, 25
suporte de áudio e vídeo, 24
software, kit de desenvolvimento. *See* Veja SDK
som, opção, 305
sp (pixel independente da escala), 152
SQLite, banco de dados. Veja também dados, armazenamento
carregadores, 282–283
criação da tabela do banco de dados, 268
criação da tabela do banco de dados,, 266
descrição básica, 29
inserir tarefa em, 272–281
objeto da tabela, 266–267
retornar toda tarefa com cursor, 283–287
SQL (Linguagem de Consulta Estruturada), 261
`SQLLiteOpenHelpder`, classe, 267–268
`src` (fonte), pasta, 78–80
SSL (Camada de Socket Seguro), 29
`startActivity( )`, método, 166, 230, 236, 344
`startService( )`, método, 167
status, barra
aumentar uma notificação, 305
carregador de progresso, 305
descrição básica, 303–304
notificação de luz, 305
notificação de som, 305
notificação de vibração, 305
notificação de visualização expansível, 306
string
arquivo `strings.xml`, 310
convenções do nome de arquivo `strings.xml`, 83
mover para recursos, 154–155
`styles.xml`, convenções do nome de arquivo, 83
submenu, 233

• T •

tabela, banco de dados, 267, 268
tabela, objeto, 266–267
`TabHost`, layout, 97
`table`, parâmetro
operação de atualização, 281
operação de consulta (leitura), 280
operação de eliminação, 282
operação de inserção, 280
tablet
adicionar transações de fragmento, 349–350
barra de ação, 334–336
comunicação entre fragmentos, 344–349
configurar emulador, 337–338
criar arquivo `ReminderListAndEditorActivity.java`, 339–340
criar layout da atividade, 342–343
desenvolver para, 329–336
escolher atividade, 340–342
fragmentos, 333
layout, 334
portar aplicativo para, 337–350
tamanho da tela, 331
versus telefone, 330
`TAG`, constante, 136
tarefa
apagar, 287–288
escolher caixa de diálogo de alerta, 253
lembrete, 236
tarefa, entrada
implementação da classe `ReminderProvider`, 274–279
operação de atualização, 281–282
operação de consulta (leitura), 280–281
operação de eliminação, 282
operação de inserção, 280
salvar valores da tela no banco de dados, 273–274
Target Android Library, pasta, 80
`TARGET_VERSION`, valor, 64
`TaskPreferences`, arquivo, 320
`task_preferences.xml`, arquivo, 318–319
Task Reminder, aplicativo
adicionar fragmento a atividade, 217

armazenar dados, 208
arquivo `reminder_edit.xml`, 220–222
arquivo `R.layout.reminder_edit_activity`, 215
atividade, usar para criar e editar lembretes, 214–216
ciclo de vida do fragmento, 219
classe `ListFragment`, 221–226
classe `NotificationManager`, 209
classe `ReminderEditActivity`, 214
classe `ReminderListActivity`, 211–212
criar layout para adicionar/editar fragmento, 220–223
distrair usuário, 208–209
exigências básicas, 207–208
fragmento `ReminderListFragment`, 212–213
fragmento versus atividade, 213
iniciar novos projetos, 209–210
intenções, 227–230
para tablets, 330–336
script do lembrete, 208
tarefa Criar, leR, atUalizar e Deletar (CRUD), 209
telas do aplicativo, 209–212
teclado
evento do teclado, 120
exibição na tela, 241–242
tela, captura, 197–198
tela, capturar, 134
tela, tamanho no tablet, 331–332
telefone, estado da campainha, 124–127
telefone versus tablet, 329–330
Telefonia, gerenciador, 30
tema, 153
temp, pasta, 47
testar
aplicativo, 149
automático, 148
`TextView`, exibição, 96, 332
`TextWatcher( )`, método, 258
thread, 15
`TIME_FORMAT`, 250
`TimePicker`, classe, 242
`TimePickerDialogFragment`, 247
tipos, array, 153
título, 200
título, campo, 220
`Toast`, mensagem, 258
`toggleUI( )`, método, 128–129
tom da chamada, 305
tools, pasta, 47
toque, evento, 120
toque, tela
capacidade multitoque, 22–23
descrição básica, 22
Tradutor, Kit de Ferramentas, 368
transmissão, mensagem, 170–171
transmitir intenções, 14
`try-finally`, bloco, 298

• U •

UI/Application Exerciser Monkey, ferramenta, 369
unidades de medida, 152
Up, botão (barra de ação), 19
`updateAppWidget( )`, método, 178–179
`update( )`, método, 279
`updatePeriodMillis`, propriedade, 180
upload do aplicativo, 198–203
`URIMatcher.match( )`, método, 278
URLs, 269–272
`usb_driver`, pasta, 47
usuário, entrada
atendente de eventos, 119–120
botões do seletor, 242–243
caixa de diálogo deu alerta, 252–257
criar interface, 239–242
evento de toque, 120
evento do teclado, 120
exibição do teclado na tela, 241–242
exibição `EditText`, 239–242
monitorar experiência do usuário, 203
nomes de arquivo, 252
seletor da hora, 247–251
seletor de data, 243–247
validação da entrada, 258
usuário, interface.. *See* Veja IU
usuário, menus de ações, 235

• V •

validar entrada do usuário, 259
valor, recurso, 153
`values/`, diretório, 82
`values`, parâmetro, 280
vermelho, ícone (mensagem de erro), 60
versão, código, 58, 63, 86–87
versão, nome, 87–88
VGA (Video Graphics Array), 64
vibração, opção, 305
vídeo e áudio, suporte, 24
`View`, classe, 96, 121
`ViewPager`, 18
visível, duração (ciclo de vida da atividade), 116
visual, designer
  abrir, 97–98
  dispositivo Nexus One, 112
  propriedades da exibição, 98–99
  visualizar aplicativos em, 110–111
`void`, método, 122
`v`, parâmetro, 226–227

• W •

`WakeLock`, objeto, 297
`WakeReminderIntentService`, classe, 296–298
`WakeReminderIntentService.java`, classe, 292
WebKit, motor do navegador Web, 29
`whereArgs`, parâmetro, 282
`whereClause`, parâmetro, 282
Wi-Fi Direct, API, 62
`wrap_content`, valor, 102, 331

• X •

`xhdpi`, pasta, 108, 156
XML, arquivo de layout do aplicativo Silent Mode Toggle, 94–96
XML, atributos do layout, 101
XML, menu, 233–234
`xmls:android="..."`, atributo, 101

• Y •

YouTube, aplicativo, 19

• Z •

zipalign, ferramenta, 369